Legitimate Peripheral Participation in Hybrid FOSS Community Innovation

Susan Gasson
Drexel University
sgasson@drexel.edu

Michelle Purcelle
Drexel University
mjw23@drexel.edu

Abstract

FOSS communities are increasingly employing a hybrid model where free, open source software development is combined with commercial customer support to ensure community sustainability. This makes it difficult for peripheral users, who are not part of the core administrative or sponsoring organization to participate meaningfully. The paper presents a study of modes of Legitimate Peripheral Participation by users who attempt to introduce product feature innovations to hybrid FOSS communities. We identify eight modes of virtual peripheral participation by users, exploring the technology and social/community affordances, and the performativity and participation effects that these engender to move peripheral users towards core membership.

1. Introduction

The emergence of hybrid Free, Open Source Software (FOSS) communities with commercial and/or institutionally-funded involvement has enabled for-profit firms to exploit the potential of open-source innovation. Most research into FOSS communities assumes that individuals participate in "bazaar" model software development, where software features are developed in full view of the community, and where product priorities are determined through a bottom-up, democratic community process. This model guides FOSS community organizing structures [3] and leads communities to adopt a one-size-fits-all technology platform to support FOSS community participation [7].

Hybrid FOSS communities combine free, open source software development with commercial customer support and sponsorship of product features, to ensure community funding and sustainability. These hybrid FOSS organizations engage users in a variety of ways that do not involve software development [1]. In this paper, we explore how the *participation architecture*, defined as "the socio-technical framework that extends participation opportunities to external parties and integrates contributions" [14, p.146], affects user participation and innovation in a hybrid FOSS community. We develop a framework for a participation architecture by considering how the theory of Legitimate Peripheral Participation (LPP) [10], a conceptual framework based on experiential learning through joint participation in co-located practice, can apply to participation in an online (not co-located) community. We end by considering the mediating effect of virtual participation by peripheral users on FOSS social structures, technology-in-practice [12], and community membership [10].

2. Conceptual underpinnings

2.1 Legitimate peripheral participation

Recent organizational studies have emphasized the situated and intertwined nature of both learning and practice, in the context of work. Lave & Wenger's theory of Legitimate Peripheral Participation (LPP) provides a framework for situated learning in Communities of Practice (CoPs). Expertise and knowledge are situated in (located in a specific, situational context) the internal logic, structural roles, cultural values, norms and meanings of *how-we-do-things-here*. LPP posits that community identity, values, and expertise are propagated through sustained, situated, joint practice. Newcomers start as peripheral group members, who lack the contextual understanding to interpret the meanings ascribed to work practices and values. Community-related expertise is acquired through a form of socially-situated apprenticeship, where individuals participate in "legitimate" practices (those that conform to the norms and values of the community), under the guidance of experienced community members. Through situated participation, they internalize the cultural meaning of social roles and norms, and the symbolic meaning of shared representations of identity, such as a preference for specific genres of communication. This allows them to move towards core community membership. In becoming a core member, they demonstrate not only expertise, but also their participation in a shared community identity [10].

URI: http://hdl.handle.net/10125/50464
ISBN: 978-0-9981331-1-9

So how might we apply LPP, a theory of action predicated on shared participation in situated, co-located practice to online community participation? Lave argues that situatedness embodies the shared understandings and practices that provide a community with its unique identity. She observes that the achievement of core CoP membership can be constrained by limiting access to the full range of activities available to participants in the community – and that this limitation on access is common in the division of labor involved in most organizational forms [9]. To understand these constraints on peripheral participation we need to explore the problems of access to the practices and legitimacy of participation that translate into core membership [9, 10].

2.2 A framework to evaluate participation architecture

West and O'Mahony originated the term *participation architecture* to denote the role played by technical platforms in supporting crowdsourced, open source software communities [14].Figure 1 shows our conceptual framework, which is based on an analysis of socio-technical affordances for online community participation. We apply this framework to explore the socio-technical affordances and effects underpinning online legitimate peripheral participation.

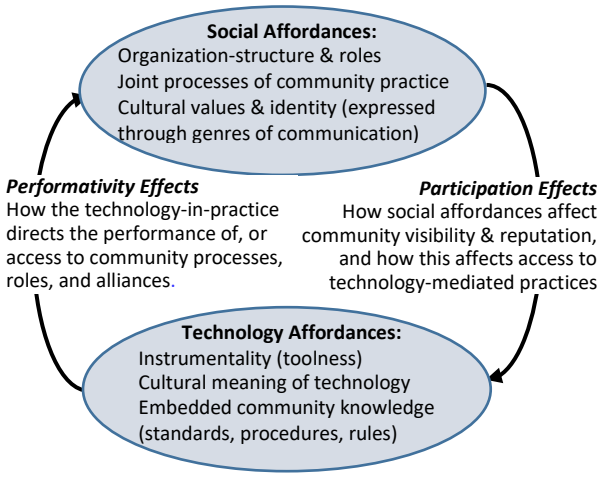

Figure 1. A socio-technical framework for online community participation architecture

2.2.1 Technical affordances. Affordances are the possibilities for action offered by an object or environment, for individuals with specific attributes [6]. In their analysis of FOSS community technology platforms, West and O'Mahony compared design features of platforms to support sponsored vs. autonomous (bazaar-model) FOSS communities. They identified features that afforded community transparency (the ability to follow and understand production efforts) and accessibility (the degree to which participants can influence production strategies) to external community members [14]. As we wish to study a different focus of collaboration (legitimate peripheral participation), we sought dimensions of technology affordance that would permit actors who derive from various backgrounds, with different experience, and who inhabit different social worlds to collaborate [10]. We adopted the three dimensions of technology affordance developed by Sun [13] to evaluate the differential effects of technology across national cultures: (i) instrumentality (toolness), (ii) cultural meaning, and (iii) embedded community knowledge (standards, procedures, rules). An online technology platform provides different affordances to its users, depending on their experience with similar platforms and their expectations of how technology should behave. Users occupying different community roles will experience a different form of *technology-in-practice* [12]. For example, an experienced product user will search the feature-repository to see if a new feature has been proposed previously (and why it was rejected), whereas an inexperienced user will just post a new feature request, receiving criticism for replicating a prior request. So the technology-in-practice affords a higher ease of feature submission to experienced users than peripheral users. The different affordances perceived by peripheral community members create *performativity effects* that direct the performance of, or access to community processes, roles, and alliances as a result of the limited technology-in-practice available to the participant [12]. The selection of FOSS technology platforms and their configuration is targeted at experienced software developers. The learning-curve means that they do not have access to a technology-mediated processes used in common by more experienced users. This presents barriers to sustained participation and membership.

2.2.2 Socio-cultural affordances. We defined three socio-cultural dimensions of an online participation architecture from the LPP literature [9, 10]:
(i) Organizational structure and roles;
(ii) Joint processes of community practice, and
(iii) Cultural values and identity, as expressed through shared genres of communication.

FOSS community roles are defined around an "onion-model" layers of expertise, with core technical community administrators planning software release priorities, the most experienced technical developers prioritizing new features because they understand existing product capabilities and constraints, sponsoring users and external developers in the next

layer influencing and feeding into these decisions, and less established (more peripheral) developers and end-users involved as volunteers to write code or explore the feasibility of new features [2]. Users of the product and recently-recruited software developers tend to be assigned less central roles, simply because of the technical knowledge required to understand product change implications, although some privileged users, who have become enculturated in community software release practices and values, participate in the inner layers. In addition to community administration roles (which are fluid, as community members spend more or less time participating over time), we also need to understand how ideas for new features are adopted, to become a priority for software development [3, 7]. A peripheral user, paired with an experienced community member can become knowledgeable about the product, and enculturated in community roles, structures, and practices, to the extent that they can participate legitimately in the community [11]. But users need to acquire situated knowledge of how to participate in the processes of creating and diffusing ideas for innovation. Whelan et al. found that two types of intermediary role are involved when traditional firms attempt to import ideas for innovation: *idea scouts*, who connect external sources of ideas for innovation with the internal network, and *idea-connectors*, who can implement those ideas [15]. It may be that similar roles are involved in achieving LPP in online innovation communities.

Finally, the social roles, influence, and alignments between various community players that result from socio-cultural participation by a peripheral user result in *participation effects*, that affect community visibility & reputation, which in turn enable or constrain access to technology-mediated practices, for example belonging to a low-status group in the community division of labor precludes access to some uses of the technology platform, which constrains LPP [9].

3. Research site and method

Evergreen is an open library system community that develops and maintains an open-source software product to support libraries, mainly in the USA and Canada. The software system helps library patrons find materials, and helps libraries manage, catalog, and circulate those materials. It is developed to be scalable and robust across any size or complexity of collections. The project uses Launchpad.net, an open software platform that allows open source software communities to manage bug reports, wishlist ideas, translations, and blueprints for the future development of their products. The Evergreen community open source project employs a hybrid, deployment business model meaning that while the code is free, users can pay for support and professional services to maintain and customize the software. The majority of development is done by paid developers at software companies and some of the user organizations. The Evergreen project would be considered relatively small, compared to other FOSS communities, based on lines of code and users. The product is in use nationally across the USA [4].

The exploratory study reported here attempts to understand how the socio-technical affordances of a participation architecture for a hybrid-FOSS community affects the ability of non-technical, peripheral users to move towards core community membership and participation. The study was performed longitudinally, over the period Fall 2014 – Fall 2017. We engaged in three types of data collection and analysis, employed simultaneously:

Community ethnography. We engaged in participant observation by engaging with various community processes and groups. We conducted frequent interviews with the core community administrators, "bug wranglers" (non-peripheral software developers or technical users), members of the Advisory Board, end-users of the software product, and non-core software developers. This allowed us to become enculturated in community roles, practices, and structures.

Analysis of technology platform affordances. We categorized the technology affordances provided by a wide range of online resources for community support, including community web-pages, the community WIKI, IRC access, online resources created by special interest groups, change request submission forms, idea tracking interfaces, code repository interfaces, and request or activity monitoring reports.

Trace ethnography. We categorized 3243 community bug reports to identify user-generated feature or change requests. We performed a content analysis on the activity logs associated with user requests, analyzing 343 critical submissions and 183 high importance submissions. We performed a content analysis on IRC log data for a period of eighteen months, from Jan 2014 to June 2015, with more recent record collection to identify patterns of user participation over time. We followed user-generated feature or change requests across channels and technology records, then related the patterns to interview and affordance analysis findings, to understand how technology-mediated peripheral participation was enacted.

Findings were synthesized across the three analyses to reassemble the "vapor trails" of participation activity [5, 8] and to develop the analytical framework shown in Figure 1, which provided an epistemic object that was clarified and elaborated as our findings emerged.

4. Findings

Our analysis identified five major enablers or constraints on Legitimate Peripheral Participation (LPP) as members in online FOSS communities. These are discussed individually in this section.

4.1 Technology-mediated participation

New peripheral community members, both users and developers, initially tended to engage with the community by posting new feature requests using a product "bug report." Participants were instructed (in the community WIKI) to propose their idea on the mailing list or IRC, before posting a feature request, to identify whether it had been discussed in the past, whether it was being considered currently, and to gauge support for the idea if neither was the case. Feature changes were assigned a priority and a status by one of 27 "bug wranglers," the core development community members. If priority was assigned as Critical or High, the feature was likely to be discussed and developed further; if it was assigned a priority of Wishlist, it was likely to remain in limbo unless a core developer took an interest in the feature.

While users were recommended to address their request to the developer mailing list or IRC first, they could submit the idea directly to the bug report tracker for inclusion on the community wishlist. We found very few instances of innovative feature requests submitted in this way. The majority of new feature-request bug reports were submitted by developers, often following discussions with users. Because developers were familiar with the software product, they often defined new features as enhancements or changes to existing features, rather than innovation. As a result of this developer intermediation, it was difficult for a user community member to establish community visibility and the commensurate legitimacy of participation for innovation that software developers earned. Success in idea selection (for product development) tended to occur when a user-idea was discussed on the technical community IRC channel or email list. These channels were more immediate than the formal bug reports interface: ideas could be elaborated, popular support garnered, and technical developer support elicited. But peripheral users needed to learn the formalisms and genres of discussion employed in these channels. For example, IRC involves a unique form of turn-taking and abbreviated meanings, as shown in the following snippet, where a peripheral participant needs to follow a discussion that is taking place between four software developers and four experienced organizational users, all diving into a rapid-fire discussion of the potential for implementing an "Awesome Box" in the software. (An Awesome Box is an alternate returns box at your library. If you checked out an item and you thought it was awesome, you return it to the Awesome Box instead of the regular returns box). The discussion veers from the potential value to library users, to ways of implementing this in the software product, to mechanisms for implementing this in a physical library, back to the software potential, then to how it might be presented in the software user-interface.

C1: Some of our members have talked about doing the awesome box.
I should find out if any have.
C2 : I know our libraries' patrons would love it
C4 : Actually, I could really see the value of doing that and then having the catalog able to flag "awesome" items in opac searches.
C2 : seems like a simple change from my naive perspective
C3 : C4, how about an optional relevance component? that'd be easy-peasy
C4 : especially if they could choose to do that for "locally awesome" items.
I would think so too.
C3 : (relatively speaking, obv ... it's still search)
C4 : right
C5 : We're going to do it. We just need a box.
Or...some place with a sign...or yeah. But we're going to do it.
C6 : ditto C5
C1 : At least one of our members is doing the awesome box already. <Link>
C5 : C11, that's where I saw it.
C10: Looks great. I may have to get to my workbench this weekend.
C11: I've registered for an awesome box (the web page), now I'm just thinking about ways to do the patron "this is awesome" both in-library and in-catalog. As well as treat awesomebox as a form of added content in the catalog.

To participate in this discussion, a new user needs to understand who is speaking (participant roles in the community), acceptable ways to take turns in IRC chats in this community channel, plus the esoteric form of shorthand terminology in use, combining library and software terms in a community-situated set of meanings. That requires experiential learning.

4.2 Participation through experiential learning

The co-construction of knowledge, where product users sought out alliances with technical developers, to explore and develop ideas prior to formal submission was key to user success in having features or feature-changes accepted. A confluence of influence is required for a new feature request to succeed. It was clear from our analysis that developers maintained a clear mental model of system priorities and make

decisions based on the synergies between new feature requests and planned work, as it was common to see a new feature request related to other bug reports (feature requests or product issues). Development priorities were driven by the request importance (a categorization applied by community administrators). Because technical developers and product users did not work together by default in developing ideas for new or changed features, they had few opportunities to develop a common language or shared priorities for change. The onus was on users to acquire an understanding of software conventions and terminology, in order to present a persuasive use-case. Socially-connected users were able to leverage their connections via email, or – better, as this was the regular hangout of core developers – via IRC discussions to engage with intermediaries who could assist them with reframing their request to attract support. Engaging with a core developer via interactive discussions in online discussion boards or an IRC chatroom could lead to the feature or issue being presented in a way that made it more relevant to core administrators, as technical developers assisted the user in framing a use-case that would communicate the critical nature of a change request, in words a software developer would understand.

If a persuasive case was made, change requests would often be implemented as a result of an administrator intervening because they agreed that the feature was important. The key skill here was for the user to frame a use-case that would be recognized as of interest to the wider community, as shown in this example, where a user convinces a community admin that the product lacks a key feature by reframing the purpose of an interface (the "aka" comment in the following discussion):

User: The Verify Credentials Screen (aka Test Password) should automatically check the password when the Enter key is pressed.

Admin: [Explanation of how the software has been fixed]. … This commit checks when the enter key is pressed, and then blanks the password on success. … Thanks for this useful usability enhancement <User>!

In contrast, when an experienced cataloging domain expert requests an innovative feature request, the developer (a core committer), attempts to enroll other cataloging experts to justify the request by tagging it with "cataloging":

User: It has always seemed like a good idea to me to include the ISBN when it is available, in the copy/item record. Often there are multiple ISBNs on a bibliographic record; for hard copy, paperback, electronic, Volume number, Part number, etc. … Having this information could be helpful in establishing replacement costs. It might be used as part of the order process to tell a vendor exactly what format and volume number you wish to purchase. If indexed, it could be used run precise reports of the volumes/copies were owned by an Org Unit.

Developer: Not sure I entirely understand what's being wished for here. I always thought ISBN was specific to the record and all related copies attached and not different per copy... Marking incomplete wishlist pending further discussion by catalogers who can help explain this procedure and wishlist item."

Tags: added: cataloging

The technology affordance of tagging does not provide any performative effect, as a user has to proactively search for and respond to feature requests that are tagged as relevant to them. So no-one responds. The user then add an explanation of how this ISBN would help in library operations, but without the social affordance of additional user support, the developer does not accept the value of the use-case and the feature request remains assigned a "wishlist" priority, making it unlikely to be implemented.

4.3 Social apprenticeships and enculturation

When a peripheral user wanted to submit a new feature or change request, they needed to learn that their request was likely to languish with no fix in sight, unless the change-request communicated what needed to be done to a technical developer, who could identify what needed to change, exploring how that section of the code worked, and involving other developers to understand problems, as shown in this feature-change request log:

User: It appears in 2.0 and 2.1, the shelving locations are not sorted alphabetically by default when you go to Advanced Search - Search Filters - Shelving Location.

Developer 1: I have a hunch this is related to <bug-report-link> which discusses the sort order of a few other interfaces [discusses evidence]…

Developer 2: I think you're right about it being id-based by default. Something else to consider is that you're supposed to be able to explicitly determine the order of shelving locations. The search filter doesn't appear to be aware of this (which should get a separate bug report).

Developer 1: I did some more digging at this today. Findings are frustrating! [discusses conflicting evidence]

Developer 2: Hello <User>, My quick testing shows that these results are simply coming out in "database order", which can be considered more or less undefined from our perspective. [detail deleted] I would try adding: <patch code> in <filename>.

User: <Developer 2> - Thanks for the suggestion. It's working perfectly! I've pushed it up to a Working repo for further evaluation.

An experienced user understands that the activity logs record interactions that took place because other users and software developers proactively sought out and supported a feature that they thought would be a positive innovation, participated in developer discussions of how the code worked over IRC, collaborated in exploring and evaluating code changes administered via github, and responded to administrative calls for additional developers to get involved in code exploration. Peripheral users could only see the start and end of this process in the log-records. They needed to learn how the need for change was evaluated, how a status was assigned to a change, how to involve developers to explore the change, who would be able to understand and work on various areas of the software, and how to ensure that the investigation continued, rather than simply being forgotten as other priorities arose. This presented a steep learning curve that was most often overcome by users participating in product evaluation, or engaging in online discussions that led to them learning how the software worked. Developers would invite users to participate in software evaluation or exploration – but for this, they needed to know that peripheral users experienced in the application domain they were attempting to explore were available for collaboration. Users needed to make themselves visible.

New organizational users not only had to learn *who-does-what* in the community, but also had to learn a new language, in order to describe operational problems with the software product in terms of how the software was designed. A core developer noted:

"For an organization considering the software or somebody who has been using the software but isn't deeply tied into some of the existing communication channels or who doesn't know some of the individuals who've spearheaded a development, rather who they are, my perception is that it could be much more of a challenge for them to figure out how to get started with, you know, with taking their idea and getting somebody to write the code for it, to write the documentation for it, and to get it folded it into the software."

A major way of providing a community social apprenticeship was via the special interest groups formed around specialist areas of library administration, including Cataloging, Acquisitions, Reports, and Academic Library management. These user-groups met regularly for the purpose of sharing information, discussing bugs and identifying ways to improve software. Special interest groups therefore played an important role in enculturating new peripheral participants, as they offer users the opportunity to engage in technical as well as social learning. There were no formal membership requirements for participation, but as these groups met virtually, except for the yearly conference, there was *both* technical and social learning involved in participating. Users needed to be informed when and on which IRC channel the meeting will occur – finding this information required local knowledge and a technical understanding of how the technology worked. But users reported that these meetings helped a lot, in obtaining advice on who to discuss issues with, or how to defined and frame a feature change request.

4.4 Establishing social capital

Obtaining social capital required community legitimacy, the ability to be recognized as possessing expertize in a valued knowledge domain. The legitimacy of peripheral user participation was undermined when decision-making focused on technical, rather than user issues. We found this to be frequently the case, as technical developers outnumbered product users in IRC discussions. The user justification of a new feature – even when supported by multiple users - was frequently subsumed to developers' interest in the difficulty of changing specific areas of code. One user, when asked about difficulties participating in online debate, discussed how he had trouble responding to a developer asking why he wanted a feature, *"It's been a month - I haven't given feedback because I didn't know how to put in words."*

Some peripheral users did appear to establish a legitimate (valued) reputation for expertise in the community, by two socially-afforded mechanisms: (i) they led a special interest group or community outreach group that core admins and developers recognized as central to the product user-base, or (ii) they proactively sought out and partnered with core technical developers. Technical developers in particular noticed – and built social networks with – users who were interested and available for collaboration in exploring software operation.

In moving towards the core, the highest-status, most trusted community members attain release commit privileges. A commit, or product-revision, is a finalized change to a software code file (or set of files). This allows them to assign software code for a new feature, feature-enhancement, or bug-fix to be integrated into a specific release of the software. To be assigned commit privileges is the ultimate in legitimate peripheral participation: the peripheral member becomes a core community member. Technical developers could build social capital by obtaining a reputation for expertise in other communities, then collaborating with core commit developers in the Evergreen community to enhance their reputation. But the hurdles that users needed to surmount appeared much higher, as their legitimacy was difficult to establish in a community focused on demonstrated

technical expertise. We only encountered one community member who had joined the community as a peripheral user and advanced to having release commit privileges. This individual had some software development experience when they joined, had been proactive in special interest group coordination, and had collaborated widely with technical developers. The announcement that they had achieved the role of core committer emphasized their technical contributions, discussing their community and user coordination work as secondary to their technical expertise.

4.5 Establishing social network support

Users perceived a need to develop a social network of technical developers who would advocate for their ideas and provide support for idea selection. Technical developers also saw the benefit in establishing social network connections with users to discuss ideas for innovation and extended their network with product users who were invited to comment on, and evaluate feature changes, even when they had not originated those changes. Experienced community user-participants described the exchange of favors in as a way of enrolling software developers in collaborative idea exploration, although they noted its limits:

"Going behind the scenes and working to get things done on a favor basis - that happens a fair bit and it is usually for fairly small things. I have some good friends among the developers in the community. I've actually gotten them to work on a number of things for me over the years on you know a quid pro quo basis. Well, we weren't doing actual exchanges one for one, but you know, I help them out they help me out, but that's not going to get me 40 hours of coding time."

Generating popular support was an important factor in having a change request implemented, so special interest groups were important in ensuring that issues relating to various library specializations were coordinated and legitimated within the developer community. But interest group members remained "outsiders" to the core technical developer community, who spoke a different language and employed different values in evaluating outcomes. They and their concerns were easily delegitimized as community influencers.

Special interest groups learned to mobilize a critical mass of social network support in order for a change request to be implemented. We identified three types of user role coming into play in this mobilization: *idea improvers* explore and add detail to feature ideas, *idea-supporters* provide support for an idea in community voting and idea discussions (important when many technical developers lack the context to understand how important a proposed change is, to the user community), and *group memory managers* provide insights into the rationale underpinning core product features and prior changes implemented. But the ability for special interest groups to mobilize support relied on their having an organizing platform. This was recognized by the Acquisitions interest group, who wished to establish a separate mailing list as their members felt uncomfortable discussing product suggestions and ideas on the general community mailing list. These were often poorly understood when initially proposed – interest group members wanted the opportunity to discuss, explore, and develop new ideas with other Acquisitions Librarians, before subjecting these to scrutiny by technical developers. But the core technical developers did not want the additional administrative load of monitoring a separate mailing list. It took many months of behind-the-scenes negotiation for a SIG mailing list to finally be configured and made available to them.

4.6 Providing funding and product code effort

The availability of resources for technical (code) development is a key consideration in explaining which features are selected for implementation. Table 1 summarizes our analysis of a sample (approx. half) of change requests submitted during a twelve month time period. In our sample period, 79 ideas were presented with code and 89 were presented without code. 77% of the ideas submitted with code were implemented and planned for a fixed product release. By contrast, only 9% of the ideas submitted without code were implemented. In a hybrid community that combines volunteer effort with the work of developers paid by member library organizations, those users or organizations who could fund or otherwise ensure effort for technical code development experienced a disproportionate influence in determining feature selection priorities for release.

Table 1. Request success with/without code

	Not implemented	Implemented in fixed release
With code	18 (23%)	61 (77%)
Without code	81 (91%)	8 (9%)

As a result of the central role played by code submission, many technology developers spent part of their working hours developing code for change requests, or worked on these in their spare time, even when this was not authorized by their employer. The software support company formalized this, to ensure innovation, rather than just bug fixing:

"Every month … we have something we call community day where all the employees of XYZ are meant to work on projects that are just simply purely for the benefit of the community … an opportunity to look into tackling the

some of the wishlist items or long standing bugs that ordinarily wouldn't be on our radar either because they don't directly affect our customers or because the thing in question is big enough where we would be looking for development funding to do it."

5. Discussion of findings

The findings above explore a variety of enablers and constraints on Legitimate Peripheral Participation (LPP) in a hybrid-FOSS community. Lave & Wenger note that we can identify some common processes inherent in the evolution of structures, roles, and processes that underpin the enactment of a community of practice [10]. Employing the conceptual model of Figure 1, we identified eight modes of LPP. Table 2 explores the following elements of each mode:
Mode of participation: how the peripheral user engages with the online community participation architecture;
Technology affordances: how the user experiences the technology-in-practice, in this mode of participation;
Performativity effects: How the technology-in-practice directs the performance of, or access to community processes, roles, and alliances;
Social affordances: Roles, processes, and interactions user can access in this mode of participation;
Participation effects: How social alliances affect community reputation & idea visibility, and how this affects access to technology-mediated joint practices;
LPP outcome: effect on legitimacy and peripherality of member participation.

Modes 1 and 2 of socio-technical participation demonstrate the social mechanism that differentiates success from failure in participating at the periphery, to submit a product innovation request. Participating in informal discussions with other users and with software developers results in the inherent performativity of using these channels for this purpose: the idea is explored in detail resulting in a persuasive use-case that sensitizes software developers to the value of making this change. These effects allows the user to make progress in moving towards the core of the community: their idea is legitimized and they gain community visibility. As a result of this form of participation the socio-technical platform presents a different form of *technology-in-practice* [12] to users who simply submit a formal change request than users who first explore the change implications through community discussions.

Modes 3 and 4 summarize ways of participating in social community engagement, by joining or leading a special interest group (or other user group). Three roles for innovation brokering were identified, in contrast to the two roles identified in prior studies [15]: *Idea improvers, who* develop and explore suggestions for innovation to expand and improve on these, *Idea-supporters* who provide social and community support to increase community awareness of an innovation, and *Group memory managers,* who provide an application domain specific repository that allows the group to understand change rationale against prior changes.

Modes 5 and 6 contrast modes of experiential learning. The first allows the co-construction of knowledge by means of discussions that explore ways of altering the product to meet a need for change. This allows a peripheral user to become enculturated in software development practices and provides them with access to a social network of technical developer contacts, with whom they can exchange favors and explore how to implement ideas. The second co-constructs knowledge by the user participating in software development, under the guidance of a more experienced technical developer. This provides the user with software expert status, gaining social capital that moves them towards the core and legitimizing their interactions with core technical developers.

Modes 7 and 8 provide the means for a peripheral participant user to exert power and influence in this form of hybrid community. Providing funding or effort for software development influences the likelihood that a user innovation will be selected for implementation. Demonstrating software development expertise, combined with application domain expertise demonstrated in technology-mediated interpersonal interactions over a sustained period of time leads to the peripheral participant being accepted into the core of community members responsible for strategic decision-making and community administration.

We conclude that a peripheral community member experiences a different technology-in-practice [12] to that experienced by core technical developers, by and for whom the technology was originally selected and configured. The technical affordances offered to peripheral users as a result constrain their access to practices that accomplish LPP in community practices. This in turn introduces *performativity effects* that direct the activity and impact of individuals, groups, or alliances. To break this "vicious cycle" of constraints, peripheral users need to be proactive in seeking out more experienced users or technical software developers, who will collaborate in shared practices to enculturate the user, to co-develop ideas and frame persuasive use-cases, and to engage the peripheral user in experiential learning, creating *participation effects* that raise the reputation of a peripheral user and improve visibility of their ideas, enabling them to engage in the co-development of software features, that advance them towards core community membership.

Table 2. Effects of various forms of participation on socio-technical participation architecture-in-practice and on outcomes affecting legitimate peripheral participation

Modes of peripheral participation	**Tech. affordances (*technology-in-practice)***	**Performativity effects of technology use**	**Socio-cultural affordances**	**Participation effects of social apprenticeship**	**LPP outcome**
1. New peripheral user does not follow advice to discuss idea before submission	Bug reporting tool offers structured idea submission and evaluation (status & priority)	None for user – idea disappears into technology black box that communicates no progress	Admin reviews change requests - only selected if user aligns feature with comm. interests	None for user, as admin selects and monitors implementation independent of user.	Little or no user learning or community visibility
2. User discusses idea on community IRC channel or email list	Steep learning curve on IRC as user understands turn-taking & genre of text communication	Idea is explored across software developers. Discussion sensitizes others to value of change	Need s/w dev. collaborator to co-create persuasive use-case	Aligns a network-of-practice: a set of developers who coordinate development activity via online tech. platform	Idea is legitimized, and user gains some visibility in community (social capital)
3. Users collaborate around shared interests in special interest group (SIG)	Collaboration with other users in virtual meetings. Coordination power depends on ability to legitimize access to exclusive space	Ideas evaluated by application domain experts; Ideas gain comm. support; Users develop community social network	*Idea improvers* develop ideas *Idea-supporters* vote for idea; *Group memory managers* recall rationale of changes	Persuasive use-case is developed for idea; Influence & social support provided SIG retains memory of change rationale	User enculturated in socio-cultural community norms & practices; User gains social network of application domain experts
4. User leads or takes prominent role in SIG or outreach	Persistence of trace records indicate role in SIG	Provides user visibility to tech. developers	Provides social capital to user with s/w developers	Legitimates user as influential decision-maker in community	Moves user away from periphery towards core
5. User engages in experiential learning by collaborating with software dev. discussions	IRC permits rapid feedback; email list posts provide diffusion of ideas	User is exposed to suggestions & questions that develop/clarify persuasive use-case	Need s/w dev. or admin sponsor to allocate effort to idea implementation	Allows user to participate in s/w devt. practices Progresses change request towards implementation	User enculturated in s/w development, practices; User gains social network of tech. developer contacts
6. User acquires software development expertise through experiential learning	Steep learning curve as user must engage via github & code tools to participate	Tech. tools for s/w development become ready-to-hand; automatic in use	User allowed to participate in near-to-core activities, (code development. & testing)	User gains access to s/w development tech. platform; allows user to modify software code	User acquires software expert status & engages with core developers
7. User provides funding or effort for software code development	Attracts attention, improving chance that idea will be selected for release	Raises status & priority of feature request or change idea	Admins prioritize changes that already have devt. effort	Feature is more likely to progress rapidly & be scheduled for release	Raises legitimacy of user as someone who can provide funding or effort
8. User demon-strates software expertise & prod. knowledge via interactions	User gains code access and modification privileges	User can influence feature adoption by providing sample code	User recognized as application domain and software domain expert	User participates in organizing product releases; Ultimately user awarded core commit privileges	User gains social capital and ultimately gains core community membership

6. Conclusions

In this study, we explored the inclusivity of the socio-technical participation architecture underpinning a hybrid-FOSS community. Our conceptual framework and its application may be distinguished from the majority of FOSS community studies, as it is analyzed from the perspective of how a peripheral, non-technical product user can participate. The majority of studies in the FOSS literature adopt the perspective of software developers – which is rational, considering that these are software communities. But this literature tells us little about how innovation may be encouraged by supporting the participation of peripheral users. Our study attempts to accomplish that aim, within the space limitations imposed by a conference paper. We presented a socio-technical framework for online community participation architecture in Figure 1 and demonstrated the application of the framework underpinning the model in exploring eight modes of virtual LPP in Table 2.

We conclude that LPP in hybrid-FOSS communities involves engagement with socio-technical enculturation, social community engagement, experiential learning that involves the co-construction of knowledge, and social legitimation. These processes culminate in the participant's assimilation of the community identity, enacting community forms, roles, and procedures as part of their membership [10], but also impacting these through a sequence of *participation effects* and *performativity effects*.

Our framework for analysis, developed partly from the synthesis of findings, has implications for research and practice. Clarifying the affordances that must be supported by the combination of technical and social online community participation architectures – and understanding their impact on the performativity of technology-in-practice and the participation effects achieved through providing access to mechanisms for social participation – are key to successful community participation by peripheral users.

7. References

[1] Carillo, K. and Bernard, J.-G., "How Many Penguins Can Hide Under an Umbrella? An Examination of How Lay Conceptions Conceal the Contexts of Free/Open Source Software", Thirty Sixth International Conference on Information Systems, 2015

[2] Crowston, K., Wei, K., Howison, J., and Wiggins, A., "Free/Libre open-source software development: What we know and what we do not know", ACM Computing Surveys (CSUR), 44(2), 2012, pp. 7:1 - 7:35.

[3] Crowston, K. and Shamshurin, I., "Core-Periphery Communication and the Success of Free/Libre Open Source Software Projects", Journal of Internet Services and Applications, 8(10), 2017, pp. 1-11.

[4] Evergreen-Project, "About Us",https://evergreen-ils.org/about-us/, accessed Sept. 4, 2017.

[5] Geiger, R.S. and Ribes, D., "Trace Ethnography: Following Coordination through Documentary Practices", Proceedings of the 44th Hawaii International Conference on System Sciences, 2011

[6] Gibson, J.J., "The Theory of Affordances", in R. Shaw and J. Bransford, (eds.): Perceiving, Acting, and Knowing, Lawrence Erlsbaum Associates, Hillsdale, NJ, 1977

[7] Howison, J. and Crowston, K., "Collaboration Through Open Superposition: A Theory Of The Open Source Way", MIS Quarterly, 38(1), 2014, pp. 29-50.

[8] Latour, B., Reassembling the Social, Oxford University Press, Oxford UK, 2005.

[9] Lave, J., "Situating Learning In Communities of Practice", in L.B. Resnick, J.M. Levine, and S.D. Teasley, (eds.): Perspectives on Socially Shared Cognition, American Psychological Association, Washington DC, 1991, pp. 63-82.

[10] Lave, J. and Wenger, E., Situated Learning: Legitimate Peripheral Participation, Cambridge University Press, Cambridge UK, 1991.

[11] Majchrzak, A. and Malhotra, A., "Towards an information systems perspective and research agenda on crowdsourcing for innovation", Journal of Strategic Information Systems 22 (2013), 22(4), 2013, pp. 257–268.

[12] Orlikowski, W., "Using Technology and Constituting Structures: A Practice Lens For Studying Technology In Organizations", Organization Science, 11(4), 2000, pp. 404-428.

[13] Sun, H., Cross-Cultural Technology Design, Oxford University Press, Oxford UK, 2012.

[14] West, J. and O'Mahony, S., "The role of participation architecture in growing sponsored open source communities", Industry and Innovation, 15(2), 2008, pp. 145–168.

[15] Whelan, E., Golden, W., and Donnellan, B., "Digitising the R&D social network: revisiting the technological gatekeeper", Information Systems Journal, 23(3), 2013, pp. 197-218.