Learning from open source software projects to improve scientific review

Citation: Ghosh, Satrajit S. et al. "Learning from Open Source Software Projects to Improve Scientific Review." Frontiers in Computational Neuroscience 6 (2012).

As Published: http://dx.doi.org/10.3389/fncom.2012.00018

Publisher: Frontiers Research Foundation

Persistent URL: http://hdl.handle.net/1721.1/73564

Version: Final published version: final published article, as it appeared in a journal, conference proceedings, or other formally published context

frontiers in
COMPUTATIONAL NEUROSCIENCE

ORIGINAL RESEARCH ARTICLE
published: 18 April 2012
doi: 10.3389/fncom.2012.00018

Learning from open source software projects to improve scientific review

***Satrajit S. Ghosh*[1]*, *Arno Klein*[2], *Brian Avants*[3] *and K. Jarrod Millman*[4]**

[1] *McGovern Institute for Brain Research, Massachusetts Institute of Technology, Cambridge, MA, USA*
[2] *New York State Psychiatric Institute, Columbia University, New York, NY, USA*
[3] *Department of Radiology, PICSL, University of Pennsylvania School of Medicine, Philadelphia, PA, USA*
[4] *Helen Wills Neuroscience Institute, University of California Berkeley, Berkeley, CA, USA*

Edited by:
Nikolaus Kriegeskorte, Medical Research Council Cognition and Brain Sciences Unit, UK

Reviewed by:
Harel Z. Shouval, University of Texas Medical School at Houston, USA
Nikolaus Kriegeskorte, Medical Research Council Cognition and Brain Sciences Unit, UK

***Correspondence:**
Satrajit S. Ghosh, McGovern Institute for Brain Research, Massachusetts Institute of Technology, 43 Vassar St., 46-4033F MIT, Cambridge, MA 02139, USA.
e-mail: satra@mit.edu

Peer-reviewed publications are the primary mechanism for sharing scientific results. The current peer-review process is, however, fraught with many problems that undermine the pace, validity, and credibility of science. We highlight five salient problems: (1) reviewers are expected to have comprehensive expertise; (2) reviewers do not have sufficient access to methods and materials to evaluate a study; (3) reviewers are neither identified nor acknowledged; (4) there is no measure of the quality of a review; and (5) reviews take a lot of time, and once submitted cannot evolve. We propose that these problems can be resolved by making the following changes to the review process. *Distributing reviews to many reviewers* would allow each reviewer to focus on portions of the article that reflect the reviewer's specialty or area of interest and place less of a burden on any one reviewer. *Providing reviewers materials and methods to perform comprehensive evaluation* would facilitate transparency, greater scrutiny, and replication of results. *Acknowledging reviewers* makes it possible to quantitatively assess reviewer contributions, which could be used to establish the impact of the reviewer in the scientific community. *Quantifying review quality* could help establish the importance of individual reviews and reviewers as well as the submitted article. Finally, we recommend *expediting post-publication reviews* and *allowing for the dialog to continue and flourish* in a dynamic and interactive manner. We argue that these solutions can be implemented by adapting existing features from open-source software management and social networking technologies. We propose a model of an open, interactive review system that quantifies the significance of articles, the quality of reviews, and the reputation of reviewers.

Keywords: distributed peer review, code review systems, open source software development, post-publication peer review, reputation assessment, review quality

INTRODUCTION

Scientific publications continue to be the primary mechanism for disseminating systematically gathered information about the natural world and for establishing precedence and credit for this research. In the current atmosphere of highly competitive and uncertain research funding, publications are instrumental in determining how resources are distributed, who gets promoted, and in which directions research advances. This has cultivated a publish-or-perish mentality where the focus is on maximizing the number of publications rather than on the validity and reproducibility of research findings, and a decrease in the amount of information apportioned to each article. Peer review is the primary means of filtering this rapidly growing literature prior to publication in an effort to ensure quality and validity.

Currently the typical review process for an article involves a preliminary screening by a journal editor followed by an anonymous and private review by a very small number of individuals (2–5, but often just 2) presumed to have expertise in the research topic (**Figure 1A**)[1]. The editor takes into consideration the reviewers' recommendations to either publish, reject, or request revisions of the article. If published, the public only sees the final version of the article without any of the reviews (however, see, BioMed Central). After publication, problems such as fraud or mistakes are addressed via retraction after disclosure or exposure by countering articles or letters to the editor

[1]Currently, reviewers are solicited by the editors of journals based on either names recommended by the authors who submitted the article, the editors' knowledge of the domain or from an internal journal reviewer database. This selection process results in a very narrow and biased selection of reviewers. An alternative way to solicit reviewers is to broadcast an article to a larger pool of reviewers and to let reviewers choose articles and components of the article they want to review. These are ideas that have already been implemented in scientific publishing. The Frontiers system (frontiersin.org) solicits reviews from a select group of review editors and the Brain and Behavioral Sciences publication (http://journals.cambridge.org/action/displayJournal?jid=BBS) solicits commentary from the community.

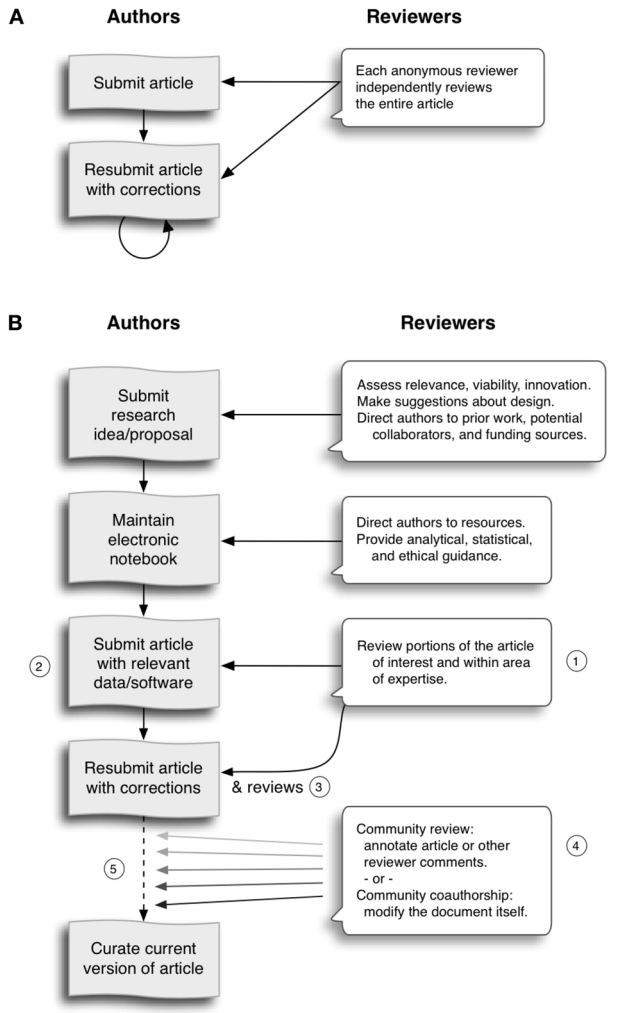

FIGURE 1 | (A) The top flowchart illustrates a standard scientific article review process before acceptance or rejection. **(B)** The bottom flowchart illustrates our model of open scientific review, from inception of an idea through its execution and documentation. This article focuses on the review of the documentation, and in particular on post-publication review. The numbers in the figure refer to the five solutions we propose to improve the review process: (1) *distribute reviews to many reviewers,* (2) *provide reviewers materials and methods to perform comprehensive evaluation,* (3) *acknowledge reviewers,* (4) *quantify review quality,* and (5) *expedite the post-publication review process and allow it to continue indefinitely.*

(e.g., Chang et al., 2006; http://retractionwatch.wordpress.com). Through peer review and the scientific community's history of policing itself, scientists hope to achieve a self-correcting process. However, this self-correction is currently impeded by slow, private, and incremental reviews without objective standards and limited post-publication feedback. Without a transparent and objective framework, journals have gained a hierarchical stature, with some attracting the best authors, articles, and reviewers. These journals have been quantified by impact factors (Garfield, 1955), and as such, have overtaken the review process as arbiters of quality and significance of research. With the difficulty for individual reviewers to review the increasing number and complexity of articles, and the use of journal impact factors as proxies for evaluations of individual articles, the integrity of the review process and, indeed, of science suffers (Smith, 2006; Poschl and Koop, 2008).

In contrast to peer review of scientific articles, when software programmers develop open source software and review their code, the process is open, collaborative, and interactive, and engages many participants with varying levels of expertise. There is a clear process by which comments get addressed and new code gets integrated into the main project. Since computer programs are much more structured and objective than prose, it is more amenable to standardization and, therefore, to review. These code review systems also take advantage of some of the latest technologies that have the potential to be used for publication review. Despite all of these differences, the purpose of code review systems mirror the purpose of publication review to increase the clarity, reproducibility, and correctness of contributions.

The most prominent example of a post-publication review system, arxiv.org, comes from the field of high energy particle physics. It has transformed the way results are disseminated, reviewed, and debated. Authors submit articles to arXiv even before they are submitted or appear in a traditional journal. Often, discussion and responses take place before the article appears in print. Interesting findings and the scientific discourse related to these findings are thus brought to the immediate attention of the community and the public. This process of rapid, fully open debate based on the exchange of technical preprints takes place even for major new results that in other fields would typically be shrouded in secrecy. A recent example was the open discussion of the possible discovery of a new particle at Fermilab's Tevatron accelerator that did not fit the Standard Model of particle physics[2]. However, this system has been applied to narrow domains of expertise, does not have a rating mechanism and its scalability in the context of increasingly interdisciplinary domains remains untested.

The advent of social networking technology has altered the traditional mechanisms of discourse, but the ease of adding to online discussions has also resulted in increasingly redundant and voluminous information. Blogs (e.g., polymathprojects.org), social network sites (e.g., Facebook, Google+) and scientific discussion forums (e.g., metaoptimize.com, mathoverflow.net, and researchgate.net) are redefining the technologies that extract, organize, and prioritize relevant, interesting and constructive information and criticism. In the scientific world, new discoveries and technologies make rapid dissemination and continued reappraisal of research an imperative. However, the scientific establishment has been slow to adopt these social technologies. The peer review system is one area where the scientific community may benefit from adopting such technologies.

For the publication review process to continue to play a critical role in science, there are a number of problems that need to be addressed. In this article, we list five problems and potential solutions that derive from distributed code review in open source software development.

[2]http://arstechnica.com/science/news/2011/05/evidence-for-a-new-particle-gets-stronger.ars

PROBLEMS WITH THE CURRENT PEER-REVIEW PROCESS

REVIEWERS ARE EXPECTED TO HAVE COMPREHENSIVE EXPERTISE

Reviewers are expected to work in isolation, unable to discuss the content of an article with the authors or other reviewers. When faced with an article that may be authored by half a dozen or more experts in their respective disciplines, how could a few reviewers be expected to have the range of expertise necessary to adequately understand and gauge the significance (or insignificance) of all aspects of a given article? Why are the different components of an article, including the background, experimental design, methods, analysis of results, and interpretations handed over as a package to each reviewer, rather than delegated to many experts in each domain? Realistically, it is common practice for a reviewer to criticize portions of an article that he or she understands, is interested in, has time to read, and takes issue with, while falling silent on the rest of the article. This leads an editor to assume these silences are indicators of tacit approval. The unrealistic expectations placed on each of the reviewers, coupled with the delayed and sequential interactions they have with the authors and editors, have made the review process inefficient.

REVIEWERS DO NOT HAVE SUFFICIENT ACCESS TO METHODS AND MATERIALS TO EVALUATE A STUDY

The typical review process does not require submission of data or software associated with an article (Association for Computing Machinery Transactions on Mathematical Software was an early exception), and the descriptions provided in methods sections are often inadequate for replication. This makes it impossible for a reviewer, if so inclined, to fully evaluate an article's methods, data quality, or software, let alone to replicate the results of the study. Failing to expose the methods, data, and software underlying a study can lead to needless misdirection and inefficiency, and even loss of scientific credibility (Ioannidis, 2005). One example is the case of Geoffrey Chang, whose rigorous and correct experimental work was later retracted due to a software bug that undermined the paper's conclusions (Chang et al., 2006).

REVIEWERS ARE NEITHER IDENTIFIED NOR ACKNOWLEDGED

Review is currently considered one's unpaid "duty" to maintain the standards and credibility of scientific research. There is little motivation for potential reviewers to participate in the review process; some motivation comes from the knowledge gained from as yet unpublished results. However, the current system does not acknowledge their services in a manner that could factor into their evaluations for promotion and funding opportunities. In addition to acknowledging a reviewer's contributions for the benefit of the reviewer, identifying a reviewer has many benefits to science and scientific discourse, including transparency of the review process and proper attribution of ideas.

THERE IS NO MEASURE OF THE QUALITY OF A REVIEW

Currently there is no way to objectively quantify the quality, strength, impartiality, or expertise of the reviews or reviewers. Without measures associated with the quality of any portion of a review, the community is forced to trust the qualitative assessment of the editor and the journal's impact factor as proxies for quality. This prevents external scrutiny and makes it impossible to evaluate or standardize the review process.

REVIEWS TAKE A LOT OF TIME AND ONCE SUBMITTED CANNOT EVOLVE

A lengthy review process holds up grant submissions, funding of research programs, and the progress of science itself. And even after this process, for the vast majority of articles none of the information (criticism or feedback) generated during the review is made publicly available (BioMed Central is one counterexample). Furthermore, after an article has been published, the review process simply ends even for those who participated, as if the work and interpretations of the results are sealed in a time capsule. Data, methods, analysis, and interpretations of the results are all a product of their time and context, and at a later time may not stand up to scrutiny or may yield new insights.

PROPOSED RE-DESIGN OF THE PEER REVIEW PROCESS

There are notable examples of journals (e.g., Frontiers—frontiersin.org, BioMedCentral—biomedcentral.com, PLoS One—plosone.org) that address one or another of the above problems, but the vast majority of journals do not address any of the above problems. We propose an open post-publication review system for scientific publishing that draws on the ideas, experience, and technologies recently developed to support community code review in open source software projects.

Figure 1B illustrates this model of open scientific review, from inception of an idea through its execution and documentation. The numbers in the figure refer to the five solutions we propose to improve the review process that addresses each of the problems listed in the prior section: (1) distribute reviews to many reviewers, (2) provide reviewers materials and methods to perform comprehensive evaluation, (3) acknowledge reviewers, (4) quantify review quality, and (5) expedite the post-publication review process and allow it to continue indefinitely. With the continued inclusion of new comments, the concept of a "publication" itself gives way to a forum or an evolving dialogue. In this sense, review can be seen as a form of co-authorship. The end-to-end review process in **Figure 1B** would integrate collaborative authoring and editing (e.g., Google docs; annotum.org—Leubsdorf, 2011), reviewing and discussion of scientific ideas and investigations. This article focuses on the review of the documentation, and in particular on post-publication review.

In this section, we describe our proposed solutions, then highlight the relevance of current code review systems in addressing the problem and finally describe enhancements to the current systems to support our proposed solution.

DISTRIBUTE REVIEWS TO MANY REVIEWERS

Reviewers would no longer work in isolation or necessarily in anonymity, benefiting from direct, dynamic, and interactive communication with the authors and the world of potential reviewers. This would help reviewers to clarify points, resolve ambiguities, receive open collegial advice, attract feedback from people well outside of the authors' disciplines, and situate the discussion in the larger scientific community. Reviewers could also focus on portions of the article that reflect their expertise and interests;

but they would, of course, have the opportunity to provide feedback on an entire article. Furthermore, they would not be held responsible for every aspect of the article, leaving portions that they are not qualified or interested in for others and their silence would not be mistaken for tacit approval. This will lessen burden placed on any one reviewer, enabling a more comprehensive, timely and scientifically rigorous review. This would also expose which portions of an article were not reviewed.

In case there is a fear of disclosure prior to publication[3], of an overwhelming amount of participation in a review where anyone could be a reviewer, or of a lack of consensus across reviewers, there are at least three types of alternatives available. One would be to assign certain reviewers as moderators for different components of the article, to lessen the burden on the editor. A second would be to restrict the number of reviewers to those solicited from a pool of experts. This would still improve scientific rigor while lessening the burden on each individual reviewer, as long as they review specific components of the article they are knowledgeable about. A third would be to conduct a preliminary review consisting of a limited, possibly anonymous and expedited review process *prior to the full and open review* as we propose. At different stages of such a tiered review, reviewers might be assigned different roles, such as mediator, editor, or commenter.

Relevance of code review systems

In the same manner that articles are submitted for review and publication in journals, code in collaborative software projects is submitted for review and integration into a codebase. In both scientific research and in complex software projects, specialists focus on specific components of the problem. However, unlike scientific review, code review is not limited to specialists. When multiple pairs of eyes look at code, the code improves, bugs are caught, and all participants are encouraged to write better code. Existing code review systems such as Gerrit (http://code.google.com/p/gerrit) as well as the collaborative development and code review functionality provided by hosting services like GitHub (http://github.com) are built for a distributed review process and provide reviewers the ability to interact, modify, annotate and discuss the contents of submitted code changes.

Indeed, the purpose of these systems mirror the purpose of scientific review—to increase the clarity, reproducibility and correctness of works that enter the canon. While no journals provide a platform for performing such open and distributed review, the Frontiers journals do provide an interactive, but non-public discussion forum for authors and reviewers to improve the quality of a submission after an initial closed review. In GitHub, code is available for everyone to view and for registered GitHub members to comment on and report issues on through an interactive web interface. The interface combines a discussion forum that allows inserting comments on any given line of code together with a mechanism for accepting new updates to the code that fix unresolved issues or address reviewer comments (an example is shown in Appendix **Figure A1**). These interactive discussions become part of a permanent and open log of the project.

Enhancing code review systems for article review

These existing code review systems, while suitable for code, have certain drawbacks for reviewing scientific articles. For example, the GitHub interface allows line-by-line commenting which reflects the structure of code. But commenting on an article's text should follow the loose structure of prose with comments referring to multiple words, phrases, sentences or paragraphs rather than whole lines. These comments should also be able to refer to different parts of an article. For example, a reviewer might come across a sentence in the discussion section of an article that contradicts two sentences in different parts of the results section. The interface should allow reviewers to expose contradictions, unsubstantiated assumptions, and other inconsistencies across the body of an article or across others' comments on the article. This system can be used in both a traditional review-and-revise model as well as a collaborative Wikipedia-style revision model that allows collaborative revision of the article. Since metrics keep track of both quality and quantity of contributions (discussed later), such an approach encourages revisions to an article that improve its scientific validity instead of a new article. A mock-up of such a review system is shown in **Figure 2**.

PROVIDE REVIEWERS MATERIALS AND METHODS TO PERFORM COMPREHENSIVE EVALUATION

In a wide-scale, open review, descriptions of experimental designs and methods would come under greater scrutiny by people from different fields using different nomenclature, leading to greater clarity and cross-fertilization of ideas. Software and data quality would also come under greater scrutiny by people interested in their use for unexpected applications, pressuring authors to make them available for review as well, and potentially leading to collaborations, which would not be possible in a closed review process.

We propose that data and software (including scripts containing parameters) be submitted together with the article. This not only facilitates transparency for all readers including reviewers but also facilitates reproducibility and encourages method reuse. Furthermore, several journals (e.g., Science—sciencemag.org, Proceedings of the National Academy of Sciences—pnas.org) are now mandating availability of all components necessary to reproduce the results (Drummond, 2009) of a study as part of article submission. The journal Biostatistics marks papers as providing code [C], data [D], or both [R] (Peng, 2009).

While rerunning an entire study's analysis might not currently be feasible as part of a review, simply exposing code can often help reviewers follow what was done and provides the possibility to reproduce the results in the future. In the long run, virtual machines or servers may indeed allow standardization of analysis environments and replication of analyses for every publication. Furthermore, including data with an article enables readers and reviewers to not only evaluate the quality and relevance of the data used by the authors of a study, but also to determine if the results generalize to other data. Providing the data necessary to reproduce the findings allows reviewers to

[3]To allay concerns over worldwide pre-publication exposure, precedence could be documented by submission and revision timestamps acknowledging who performed the research.

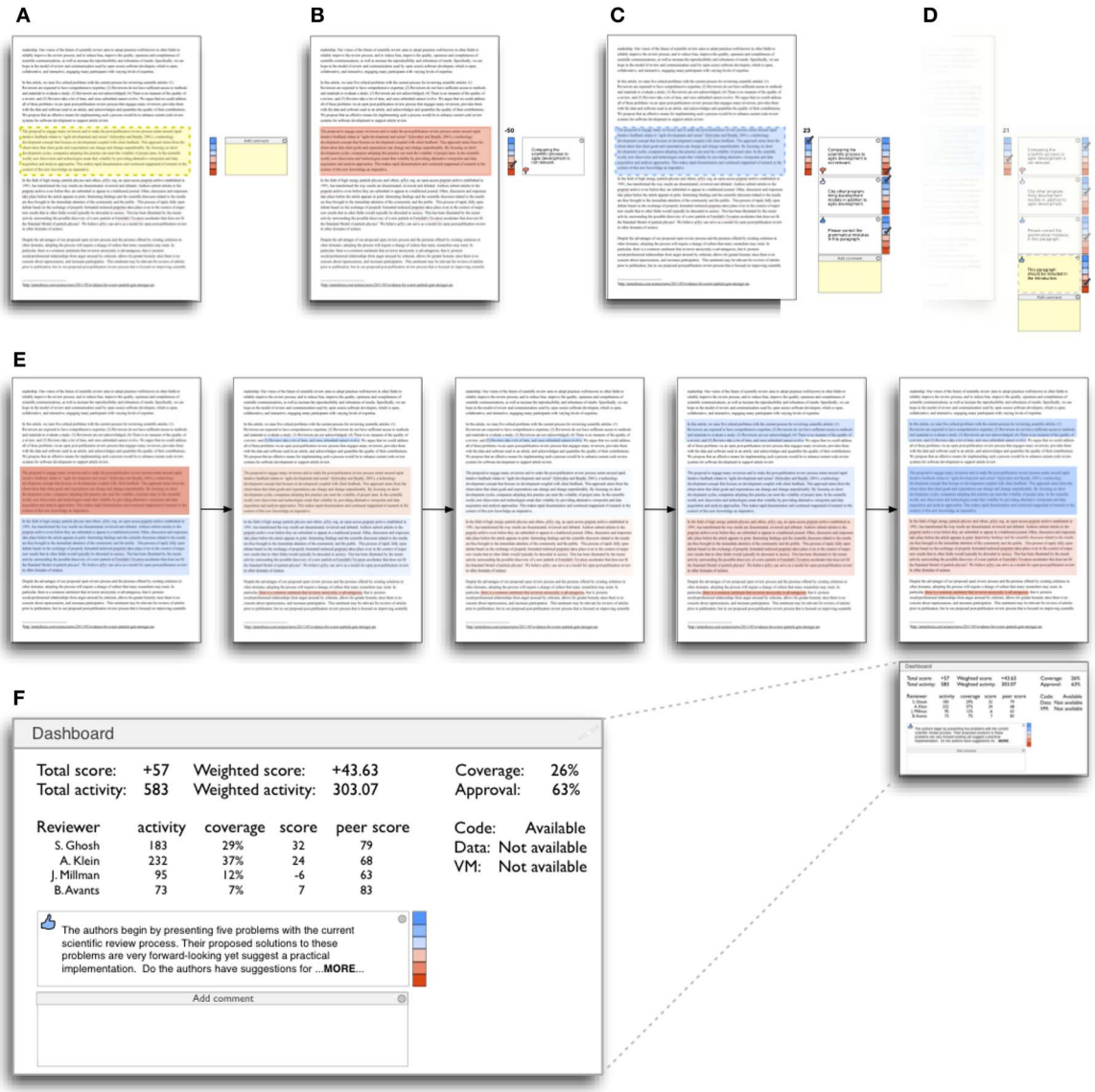

FIGURE 2 | This schematic illustrates color-coded ratings assigned to text in an article or reviewer comment. Such a visualization could help authors, reviewers, and editors quickly assess how much of and how favorably an article has been reviewed, and could be useful in a publishing model where an article is considered published after it garners a minimum rating over an appreciable amount of its content. **(A)** A reviewer selects some text which launches a colorbar for rating the text and a comment box, and **(B)** gives a low rating (red) for the text and adds a negative comment (a thumbs down appears in the comment box to reflect the rating). **(C)** Another reviewer selects the same block of text (which launches a comment box), then rates the text and some of the other comments. A red or blue background indicates a cumulative negative or positive rating. In this example, the positive ratings outweigh that of the initial negative comment, turning the text from red to blue. Each reviewer's vote can be weighted by the ratings received by that reviewer's past contributions to peer review. **(D)** A reviewer selects the bottom-most comment to leave a comment about it. **(E)** The middle row shows how the ratings of an article's text can change over time. **(F)** The bottom row represents a dashboard summary of the ratings assigned to an article, including reviewer activity, coverage, and variation of opinion regarding the article. General comments can also be added for the article as a whole via the dashboard. The dashboard also indicates whether code, data and/or a virtual machine are available for reproducing the results of the article.

potentially drill down through the analysis steps—for example, to look at data from each preprocessing stage of an image analysis pipeline.

Relevance of code review systems

While certain journals (e.g., PLoS One, Insight Journal) require code to be submitted for any article describing software or algorithm development, most journals do not require submission of relevant software or data. Currently, it is considered adequate for article reviewers to simply read a submitted article. However, code reviewers must not only be able to read the code, they must also see the output of running the code. To do this they require access to relevant data or to automated testing results. Code review systems are not meant to store data, but

complement such information by storing the complete history of the code through software version control systems such as Git (git-scm.com) and Mercurial (mercurial.selenic.com). In addition to providing access to this history, these systems also provide other pertinent details such as problems, their status (whether fixed or not), timestamps and other enhancements. Furthermore, during software development, specific versions of the software or particular files are tagged to reflect milestones during development. Automated testing results and detailed project histories provide contextual information to assist reviewers when asked to comment on submitted code.

Enhancing code review systems for article review

As stated earlier, code review systems are built for code, not for data. Code review systems should be coupled with data storage systems to enable querying and accessing code and data relevant to the review.

ACKNOWLEDGE REVIEWERS.

When reviewers are given the opportunity to provide feedback regarding just the areas they are interested in, the review process becomes much more enjoyable. But there are additional factors afforded by opening the review process that will motivate reviewer participation. First, the review process becomes the dialogue of science, and anyone who engages in that dialogue gets heard. Second, it transforms the review process from one of secrecy to one of engaging social discourse. Third, an open review process makes it possible to quantitatively assess reviewer contributions, which could lead to assessments for promotions and grants. To acknowledge reviewers, their names (e.g., Frontiers) and contributions (e.g., BioMed Central) can be immediately associated with a publication, and measures of review quality can eventually become associated with the reviewer based on community feedback on the reviews.

Relevance of code review systems

In software development, registered reviewers are acknowledged implicitly by having their names associated with comments related to a code review. Systems like Geritt and GitHub explicitly list the reviewers participating in the review process. An example from Geritt is shown in supplementary **Figure A2**.

In addition, certain social coding websites (e.g., ohloh.net) analyze contributions of developers to various projects and assign "kudos" to indicate the involvement of developers. **Figure 3** shows an example of quantifying contributions over time. Neither of these measures necessarily reflect the quality of the contributions, however.

Enhancing code review systems for article review

The criterion for accepting code is based on the functionality of the final code rather than the quality of reviews. As such, code review systems typically do not have a mechanism to rate reviewer contributions. We propose that code review systems adapted for article review include quantitative assessment of the quality of contributions of reviewers. This would include a weighted combination of the number (**Figure 3**), frequency (**Figure 4**), and peer ratings (**Figure 2**) of reviewer contributions. Reviewers need not be the only ones to have an impact on other reviewers' standing. The authors themselves could evaluate the reviewers by assigning impact ratings to the reviews or segments of the reviews. These ratings can be entered into a reviewer database, referenced in the future by editors and used to assess contributions to peer review in the context of academic promotion. We acknowledge some reviewers might be discouraged by this idea, thus it may be optional to participate.

QUANTIFY REVIEW QUALITY

Although certain journals hold a limited discussion before a paper is accepted, it is still behind closed doors and limited to the editor, the authors, and a small set of reviewers. An open and recorded review ensures that the role and importance of reviewers and information generated during the review would be shared and acknowledged. The quantity and quality of this information can be used to quantitatively assess the importance of a submitted article. Such quantification could lead to an objective standardization of review.

There exist metrics for quantifying the importance of an author, article, or journal (Hirsch, 2005; Bollen et al., 2009), but we know of no metric used in either article review or in code review for quantifying the quality, impact, or importance of a review, of a comment on a review, or of any portions thereof. Metrics have many uses in this context, including constructing a dynamic assessment of individuals or ideas for use in promotion and allocation of funds and resources. Metrics also make it possible to mine reviews and comment histories to study the process of scientific publication.

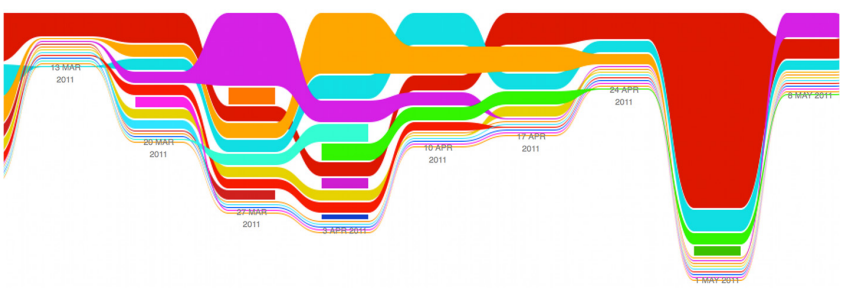

FIGURE 3 | Example of a metric for quantifying contributions over time. This is a screenshot of a ribbon chart visualization in GitHub of the history of code additions to a project, where each color indicates an individual contributor and the width of a colored ribbon represents that individual's "impact" or contributions during a week-long period.

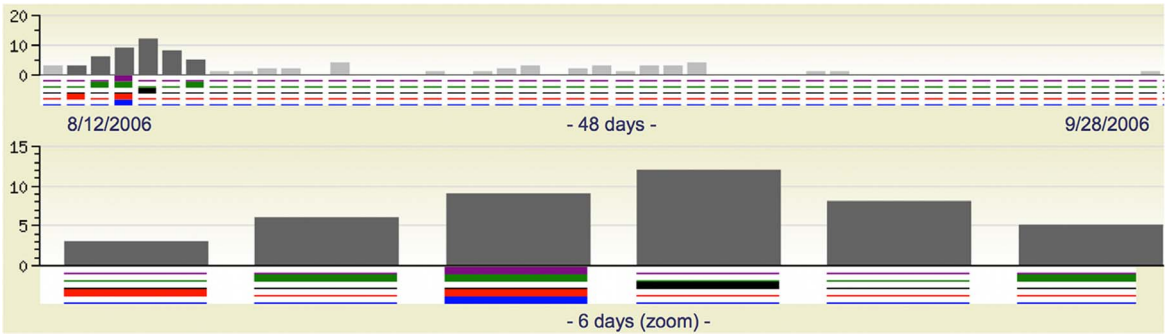

FIGURE 4 | Example of a metric for quantifying contributor frequency. Quotes over Time (www.qovert.info) tracked the top-quoted people from Reuters Alertnet News on a range of topics, and presents their quotes on a timeline, where color denotes the identity of a speaker and bar height the number of times the speaker was quoted on a given day.

Relevance of code review systems

In general, code review systems use a discussion mechanism, where a code change is moderated through an iterative process. In the context of code review, there is often an objective criterion—the code performs as expected and is written using proper style and documentation. Once these standards are met, the code is accepted into the main project. The discussion mechanism facilitates this process. Current code review systems do not include quantitative assessment of the quality of reviews or the contributions of reviewers.

Enhancing code review systems for article review

The classic "Like" tally used to indicate appreciation of a contribution in Digg, Facebook, etc., is the most obvious measure assigned by a community, but it is simplistic and vague. In addition to slow and direct measures of impact such as the number of times an article is cited, there are faster, indirect behavioral measures of interest as a proxy for impact that can be derived from clickstream data, web usage, and number of article downloads, but these measures indicate the popularity but not necessarily quality of articles or reviews.

We propose a review system (**Figure 2**) with a "reputation" assessment mechanism similar to the one used in discussion forums such as stackoverflow.net or mathoverflow.net in order to quantify the quality of reviews. These sites provide a web interface for soliciting responses to questions on topics related to either computer programming or mathematics, respectively (supplementary **Figure A3**). The web interface allows registered members to post or respond to a question, to comment on a response, and to vote on the quality or importance of a question, of a response, or of a comment. In our proposed review system, such a vote tally would be associated with identified, registered reviewers, and would be only one of several measures of the quality of reviews (and reviews of reviews) and reviewers. Reviews can be ranked by importance (weighted average of ratings), opinion difference (variance of ratings) or interest (number of ratings). Reviewer "reputation" could be computed from the ratings assigned by peers to their articles and reviews.

It would also be possible to aggregate the measures above to assess the impact or importance of, for example, collaborators, coauthors, institutions, or different areas of multidisciplinary research. As simple examples, one could add the number of contributions by two or more coders in **Figure 3** or the number of quotations by two or more individuals in **Figure 4**. This could be useful in evaluating a statement in an article in the following scenario. Half of a pool of reviewers A agrees with the statement and the other half B disagrees with the statement. Deciding in favor of group A would be reasonable if the aggregate metric evaluating A's expertise on the statement's topic is higher than that of B. However, such decisions will only be possible once this system has acquired a sufficient amount of data about group A and B's expertise on reviewing this topic, where expertise is related to the "reputation" assessment mentioned above.

EXPEDITE REVIEWS AND ALLOW FOR CONTINUED REVIEW.

Once open and online, reviews can be dynamic, interactive, and conducted in real time (e.g., Frontiers). And with the participation of many reviewers, they can choose to review only those articles and components of those articles that match their expertise and interests. Not only would these two changes make the review process more enjoyable, but they would expedite the review process. And there is no reason for a review process to end. Under post-publication review, the article can continue as a living document, where the dialogue can evolve and flourish (see **Figure 5**), and references to different articles could be supplemented with references to the comments about these articles, perhaps as Digital Object Identifiers (http://www.doi.org), firmly establishing these communications within the dialogue and provenance of science, where science serves not just as a method or philosophy, but as a social endeavor. This could make scientific review and science a more welcoming community.

Relevance of code review systems

Code review requires participation from people with differing degrees of expertise and knowledge of the project. This leads to higher quality of the code as well as faster development than individual programmers could normally contribute. These contributions can also be made well beyond the initial code review allowing for bugs to be detected and improvements to be made by new contributors.

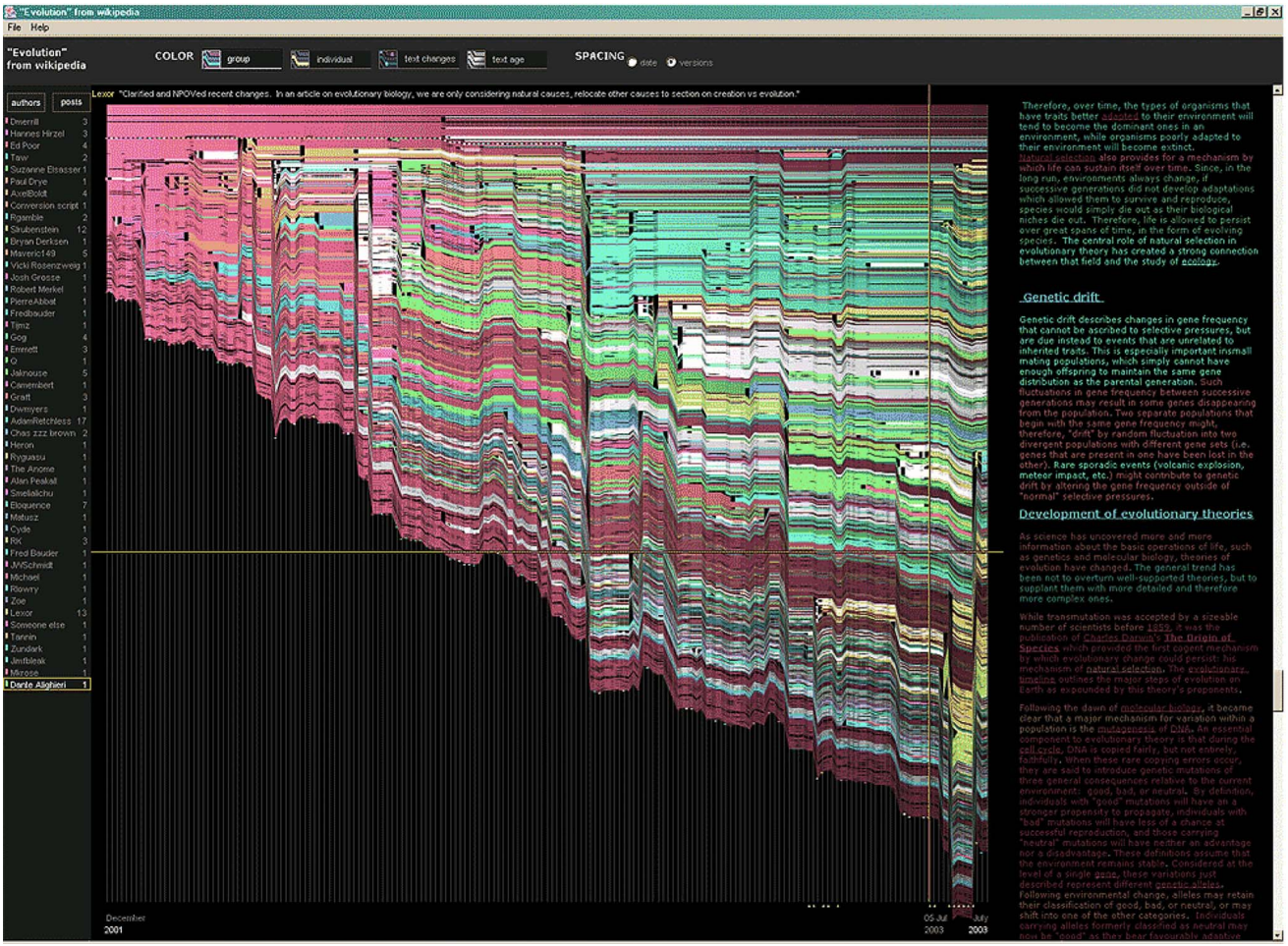

FIGURE 5 | A visualization of the edit history of the interactions of multiple authors of a Wikipedia entry ("Evolution"). The text is in the right column and the ribbon chart in the center represents the text edits over time, where each color indicates an individual contributor (http://www.research.ibm.com/visual/projects/history_flow/gallery.htm, Viegas et al., 2004).

Enhancing code review systems for article review

Current code review systems have components for expedited and continued review. Where they could stand to be improved is in their visual interfaces, to make them more intuitive for a non-programmer to quickly navigate (**Figure 2**), and to enable a temporal view of the evolutionary history of an arbitrary section of text, analogous to **Figure 5** (except as an interactive tool). As illustrated in **Figure 1B** and mentioned in the Discussion section below, co-authorship and review can exist along a continuum, where reviewers could themselves edit authors' text in the style of a wiki (e.g., www.wikipedia.org) and the authors could act as curators of their work (as in www.scholarpedia.org).

DISCUSSION

The current review process is extremely complex, reflecting the demands of academia and its social context. When one reviews a paper, there are considerations of content, relevance, presentation, validity, as well as readership. Our vision of the future of scientific review aims to adopt practices well-known in other fields to reliably improve the review process, and to reduce bias, improve the quality, openness and completeness of scientific communications, as well as increase the reproducibility and robustness of results. Specifically, we see hope in the model of review and communication used by open source software developers, which is open, collaborative, and interactive, engaging many participants with varying levels of expertise.

In this article, we raised five critical problems with the current process for reviewing scientific articles: (1) reviewers are expected to have comprehensive expertise; (2) reviewers do not have sufficient access to methods and materials to evaluate a study; (3) reviewers are neither identified nor acknowledged; (4) there is no measure of the quality of a review; and (5) reviews take a lot of time, and once submitted cannot evolve. We argue that we can address all of these problems via an open post-publication review process that engages many reviewers, provides them with the data and software used in an article, and acknowledges and quantifies the quality of their contributions. In this article, we described this process (**Figure 1B**) together with a quantitative commenting

mechanism (**Figure 2**). We anticipate that such a system will speed up the review process significantly through simultaneous, distributed, and interactive review, an intuitive interface for commenting and visual feedback about the quality and coverage of the reviews of an article. The proposed framework enables measurement of the significance of an article, the quality of reviews and the reputation of a reviewer. Furthermore, since this system captures the entire history of review activity, one can refer to or cite any stage of this evolving article for the purpose of capturing the ideas and concepts embodied at that stage or quantifying their significance over time.

Despite the advantages of our proposed open review process and the promise offered by existing solutions in other domains, adopting the process will require a change of culture that many researchers may resist. In particular, there is a common sentiment that reviewer anonymity is advantageous, that it: protects social-professional relationships from anger aroused by criticism, allows for greater honesty since there is no concern about repercussions, and increases participation. However, in the current system the combination of anonymity, lack of accountability, and access to author material creates the potential for serious problems such as the use of the authors' ideas without acknowledgment of their source. Under the proposed system, people who implement the system will have the option to consider which components remain anonymous but reviewers would be tracked, potentially alleviating this issue. Furthermore, the open post-publication review system prevents any single person from blocking a publication or giving it a negative rating. The transparency of such a system will also reduce any single individual or group's ability to game the system. To further curtail the selfish tendencies of some reviewers, comments they make about the text would themselves be subject to review by others, and it would be in their own self-interest to maintain a high rating in their peer community.

In the long run, the review process should not be limited to publication, but should be engaged throughout the process of research, from inception through planning, execution, and documentation (Butler, 2005; see **Figure 1B**). Open review at every stage of a scientific study would facilitate collaborative research and mirror open source project development closely. Such a process would also ensure that optimal decisions are taken at every stage in the evolution of a project, thus improving the quality of any scientific investigation. We envision a system where the distinction between authors and reviewers is replaced simply by a quantitative measure of contribution and scientific impact, especially as reviewers can act as collaborators who play a critical role in improving the quality and, therefore, the impact of scientific work. Where there is significant concern about exposing ideas before an article is written, reviewers could be drawn from collaborators, funding agencies, focus groups, or within the authors' institutions or laboratories, rather than the general public. In such scenarios either the review process or the identity of reviewers or both could be kept hidden but tracked for the purposes of "reputation assessment" (see above) and accountability.

Changing the review process in ways outlined in this article should lead to better science by turning each article into a public forum for scientific dialogue and debate. The proposed discussion-based environment will track and quantify impact of not only the original article, but of the comments made during the ensuing dialogue, helping readers to better filter, find, and follow this information while quantitatively acknowledging author and reviewer contributions and their quality. Our proposed re-design of the current peer review system focuses on post-publication review, and incorporates ideas from code review systems associated with open source software development. Such a system should enable a less biased, comprehensive, and efficient review of scientific work while ensuring a continued, evolving, public dialogue.

ACKNOWLEDGMENTS

We would like to thank Matthew Goodman, Yaroslav Halchenko, Barrett Klein, Kim Lumbard, Fernando Perez, Jean-Baptiste Poline, Elizabeth Sublette, and the Frontiers reviewers for their helpful comments. Arno Klein would like to thank Deepanjana and Ellora, as well as the NIMH for their support via R01 grant MH084029. Brian Avants acknowledges ARRA funding from the National Library of Medicine via award HHSN276201000492p.

REFERENCES

Bollen, J., van de Sompel, H., Hagberg, A., and Chute, R. (2009). A principal component analysis of 39 scientific impact measures. *PLoS One* 4:e6022. doi: 10.1371/journal.pone.0006022

Butler, D. (2005). Electronic notebooks: a new leaf. *Nature* 436, 20–21.

Chang, G., Roth, C. B., Reyes, C. L., Pornillos, O., Chen, Y.-J., and Chen, A. P. (2006). Retraction. *Science* 314, 1875.

Drummond, C. (2009). "Replicability is not reproducibility: nor is it good science," in *Proceedings of the Evaluation Methods for Machine Learning Workshop at the 26th ICML*. (Montreal, Canada). Citeseer.

Garfield, E. (1955). Citation indexes to science: a new dimension in documentation through association of ideas. *Science* 122, 108–111.

Hirsch, J. (2005). An index to quantify an individual's scientific research output. *Proc. Natl. Acad. Sci. U.S.A.* 102, 16569.

Ioannidis, J. (2005). Why most published research findings are false. *PLoS Med.* 2:e124. doi: 10.1371/journal.pmed.0020124

Leubsdorf, C. Jr. (2011). "Annotum: an open-source authoring and publishing platform based on WordPress," in *Proceedings of the Journal Article Tag Suite Conference*. (Bethesda, MD: National Center for Biotechnology Information US).

Peng, R. D. (2009). Reproducible research and Biostatistics. *Biostatistics* 10, 405–408.

Poschl, U., and Koop, T. (2008). Interactive open access publishing and collaborative peer review for improved scientific communication and quality assurance. *Inform. Serv. Use* 28, 105–107.

Smith, R. (2006). Peer review: a flawed process at the heart of science and journals. *J. R. Soc. Med.* 99, 178.

Viegas, F., Wattenberg, M., and Dave, K. (2004). "Studying cooperation and conflict between authors with history flow visualizations," in *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, (New York, NY, USA: ACM Press), 575–582.

Conflict of Interest Statement: The authors declare that the research was conducted in the absence of any commercial or financial relationships that could be construed as a potential conflict of interest.

Received: 06 June 2011; accepted: 16 March 2012; published online: 18 April 2012.

*Citation: Ghosh SS, Klein A, Avants B and Millman KJ (2012) Learning from open source software projects to improve scientific review. Front. Comput. Neurosci. **6**:18. doi: 10.3389/fncom.2012.00018*

APPENDIX

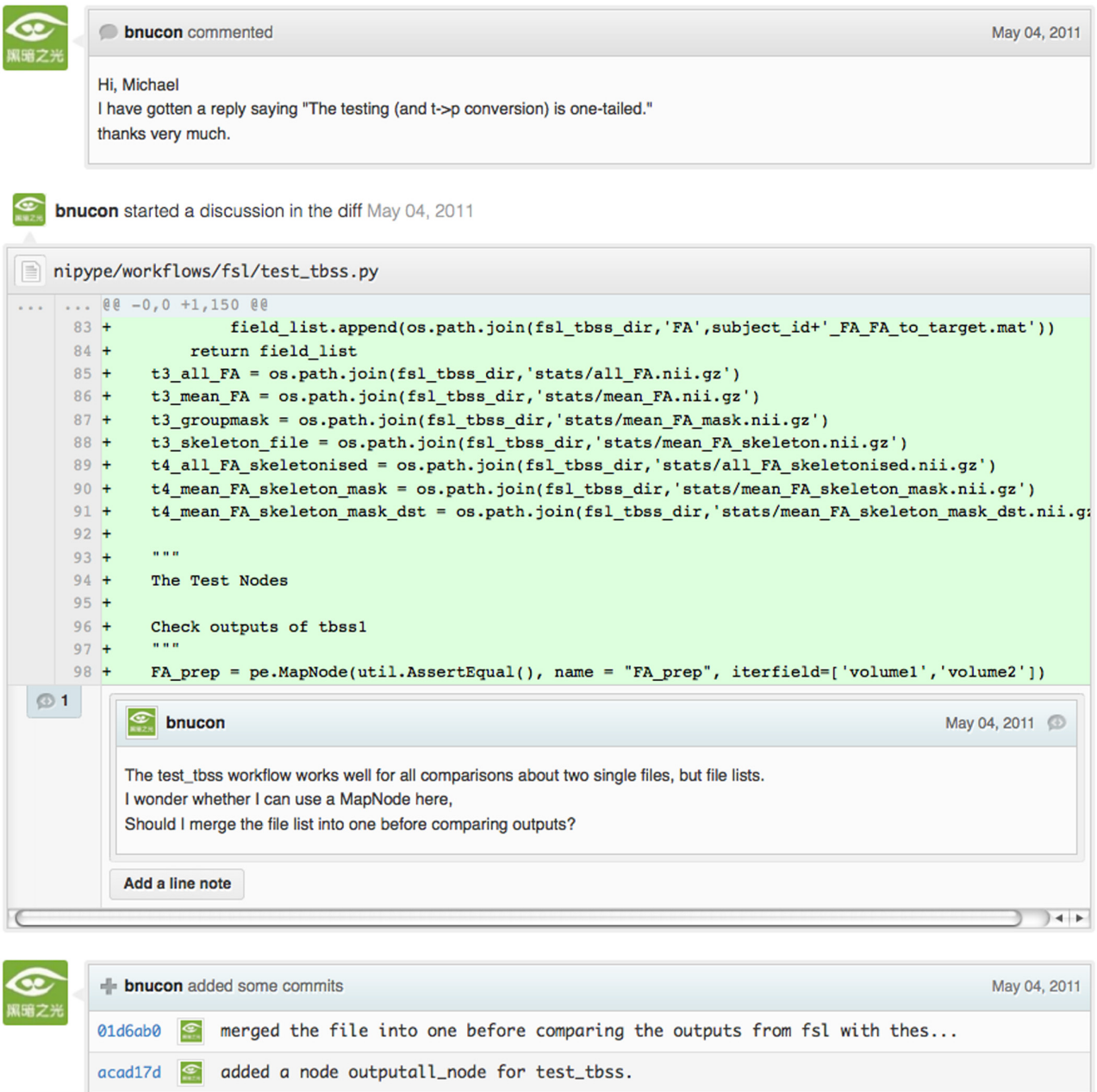

FIGURE A1 | A snapshot from the web interface of a request to merge code into the NiPyPE (nipy.org/nipype) project on GitHub. This demonstrates: **(A)** part of a discussion thread, **(B)** inline commenting of code (for line 98), and **(C)** code updates (commits) taking place as a function of the discussion.

Reviewer	*Verified*	*Code Review*	
Gaëtan Lehmann			
Hans J. Johnson		+1	Looks good to me, but someone else must approve
Andrew Wasem		+1	Looks good to me, but someone else must approve
Jim Miller			

- Need Verified +1 (Verified)
- Need Code Review +2 (Looks good to me, approved)

FIGURE A2 | A web page snippet from the Geritt code review system used for Insight Toolkit (itk.org). This explicitly lists the reviewers who are participating in the review.

How does "Reputation" work?

170

55

On Stack Exchange, users may gain a certain level of *reputation*.

- What does *reputation* do?
- How can a user gain or lose *reputation*?

1 Answer

active oldest **votes**

220

What does *Reputation* do?

As a registered user, your reputation on the site is a part of your identity on the site. It determines, to an extent, your familiarity with the site, the amount of subject matter expertise you have and the level of respect your peers have for you. It can generally only be gained when other users of the site approve of the content you provide.

Reputation also determines a user's privileges within the system. As you gain more reputation, the system learns to trust you and bestows new functionality upon you that low-reputation users cannot access.

As users gain reputation, they gain abilities and responsibilities. The required reputation amounts on different sites can vary slightly; see your site's `/privileges` page for specifics. **Common privilege levels for new sites, public beta sites and "normal" sites are described here.**

FIGURE A3 | A response to a question on stackoverflow.net. The top left number (170) indicates the number of positive votes this response received. There are comments to the response itself and the numbers next to the comments reflect the number of positive votes for each comment (e.g., 220 in this example). (http://meta.stackoverflow.com/questions/76251/how-do-suggested-edits-work).