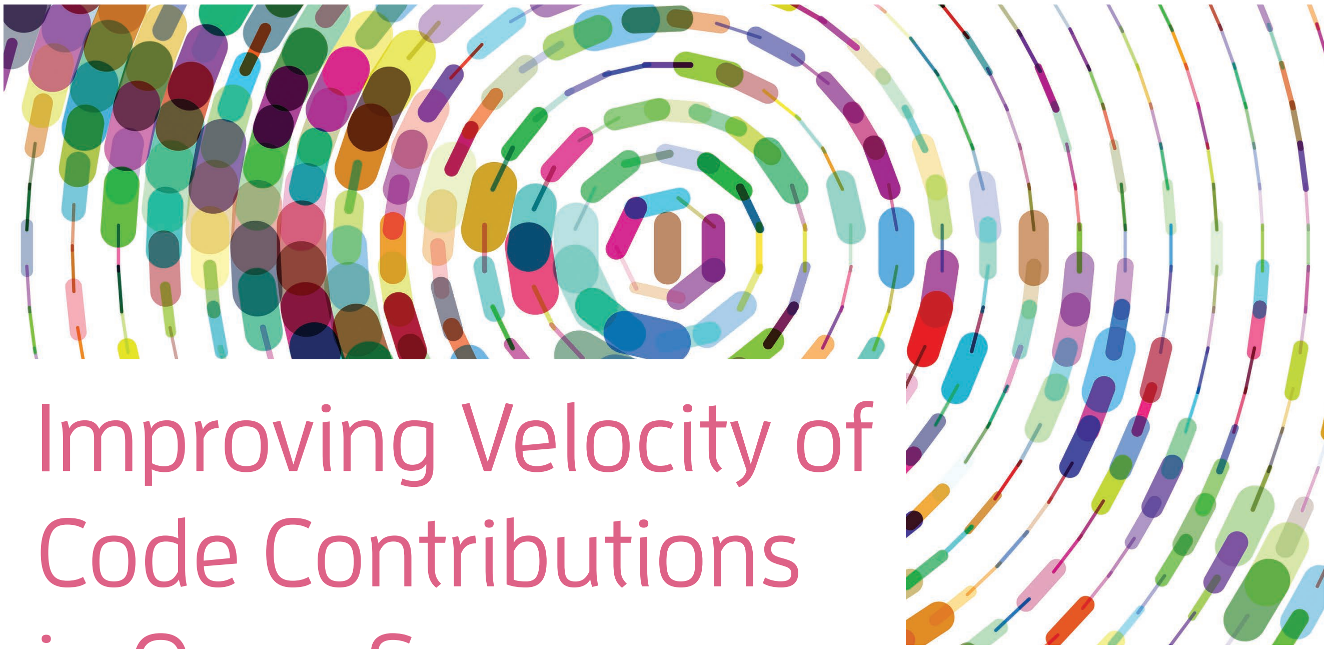

Improving Velocity of Code Contributions in Open Source

Irina Zaks, Stanford Open Source Lab
Timothy Lehnen, Drupal Association
Aaron Zaks, University of California at Berkeley

Velocity is speed with direction. If we are not moving in the right direction, higher speed does not bring us to our goal quicker. "If one does not know to which port one is sailing, no wind is favorable," said Seneca.

Our case study is focused on modeling processes in open source software that accelerate new feature development. The goal of our study is to 1) examine open source contribution patterns using data from development of the project messaging module and 2) identify and quantify processes that make a difference between success and failure in open source projects and explore patterns that impact the velocity of contributions.

In the study, we are using data provided by Drupal Association related to development of the "project messaging" module in Drupal core. Previous research outlines phases and components of the software development life cycle for open source software, including various techniques for managing a community of contributors to accelerate innovation. We are using phases of development similar to the patterns found in prior literature. We plan to extend the model developed in this study to analyze more projects, including both Drupal with composer and Backdrop CMS, a fork of Drupal. In addition to quantitative analysis of the data (commits, messages, issues) we will provide qualitative insight into how Drupal community's specific governance and contribution models impact the end result—delivering features that the community prioritized for the project.

The term *open source* refers to software that is made freely available to the public, allowing anyone to access, modify, and distribute the source code.[1]

Drupal is an open source content management system that supports efficient development of complex websites,

Digital Object Identifier 10.1109/MC.2023.3327893
Date of current version: 3 June 2024

EDITOR DIRK RIEHLE
Friedrich Alexander-University of Erlangen Nürnberg;
dirk.riehle@fau.de

and is widely used in higher ed, ecommerce, government, and nonprofits.

In this article, we are measuring "commits" as a unit of code contribution. Commits are not an absolute measure of code quality as they vary by features that they add, number of lines of code, and other parameters, but they are used herein as a standard unit in measuring software development.

We outline here major factors that impact the velocity of software development in an open source context and is well defined in previous research:

1. community size
2. community engagement
3. size of the development team
4. communication and collaboration
5. project complexity
6. developer experience
7. code quality
8. development methodology
9. quality assurance processes
10. funding and resources
11. project governance.

The following phases of the software development process are well defined in the industry:

1. planning
2. analysis
3. design
4. implementation
5. testing
6. deployment
7. maintenance.

In open source projects, these phases are not necessarily rigidly followed. It is not uncommon for a contribution to start with code (Phase 4—Implementation) as a motivated developer tries to demonstrate an idea.

However, as we will show, the more ambitious the feature/contribution/initiative, the more likely it is that every phase in this lifecycle will be reached—even if out of order.

FROM THE EDITOR

Welcome back to our column on open source! In a past article, Goggins et al. discussed how to measure open source project and community health, an important topic on the mind of everyone who is using open source and depends on it. In this article, Zaks et al. look at a particular aspect, the focus and speed of code contributions to an open source project, to discuss what it can tell us about the project's health. As always, stay happy, stay healthy, be safe.—*Dirk Riehle*

RESEARCH DATA

Analysis of the "project messaging" module contribution process

The Drupal project messaging module was introduced in 2021 to provide a channel for messaging about the Drupal project within the administrative interface for end users.

What started as a community module was then prioritized as a "strategic initiative." In Drupal, strategic initiatives are promoted by project leadership, given priority for code review, and contributors are actively recruited toward these efforts.

Contribution channels

Development and collaboration for Drupal projects happens in three major spaces: Drupal.org, Gitlab, and in Slack channels. Each space can have its own participants and supports different stages of the development process.

1. Drupal.org is the center of collaboration for the project. Crucially, it hosts the issue tracker where feature requests, bugs, policy decisions, documentation, and so on are planned and prioritized. Readers may be familiar with commercial tools such as JIRA, Trello, and so on that provide similar functionality.
   Units of activity in an issue occur in comments, and we will measure the volume of comments on issues in our analysis.
2. Slack provides a real-time chat service organized into channels. Real-time collaboration tools like this often substitute for in-person collaboration that would happen in a traditional office environment in proprietary software development. The unit of measurement for Slack activity is a "message."
3. The Drupal project also uses self-hosted instance of Gitlab: git.drupalcode.org. GitLab is an open source version control and collaboration platform using common workflows like merge requests (MR), commits, and so on. The unit of measurement in this space is "commit."

In this case study, we analyze each of these collaboration channels, based on the number of participants and units of contribution (comments/messages/commits). We plot this activity over time, overlaying other critical factors such as holidays, conferences, and so on.

We used stacked bar charts to visualize two major patterns that are important for our research: 1) distributions of comments/commits/messages to show two patterns over time (per month or per week) and 2) distribution of users that participated in each channel. Color coding by user instantly

Phase	Issues	Slack	git.drupalcode.org
1) Planning	x	x	
2) Analysis	x	x	
3) Design	x	x	
4) Implementation	x	x	x
5) Testing	x	x	x
6) Deployment			x
7) Maintenance	x		x

shows how users participated throughout the project and quantifies participants turnover at various phases and in various channels.

We are using a grouped graph to show correlation between trends in three processes that result at the end as a released feature/project.

Data was provided in the following way:

- *For comments*: Data was queried from the Drupal.org database and provided in .csv format.
- *For commits*: Data was pulled from the git.drupalcode.org api in JavaScript Object Notation (JSON) format. The JSON is read and processed in such a way that every element of the array has two essential attributes: "date" and "author." date: : JS Date. object: : String.
- *For messages*: Slack's analytics reports were used to provide data on slack activity in specific project channels during the study period.

We opted to use the open source "d3js" visualization library.[2]

Stacked graph: The data are "bucketed" by month. Then, within each bucket, the number of messages (or issues/comments or commits) are counted by the author, with the total number of participants shown at the top of each bar. Finally, a color palette is constructed by matching each author to uniformly spaced colors on d3's interpolateTurbo() scheme.

The array of authors is collected according to a sorted list of messages, so the position on the rainbow is indicative of how early they participated in the project. For example, the first person will always be represented by a yellow bar, and the last by a red bar.

Source code repo and data source

Code is available in this git repo at: https://github.com/litelordtrue/IEEE-Velocity-Research

ANALYSIS

Issues and comments on Drupal.org

Comments summary
Started: 31 March 2021
Completed: 31 May 2023
Issue url: https://www.drupal.org/project/drupal/issues/3206643
Total Comments: 238
Total users/participants: 38

We can clearly see two peaks in discussions with almost a year interval in between (Figure 1). We can also see that the pool of participants changed almost completely.

Looking more closely at the specific users represented in the data, we find that in 2021 staff at the Drupal Association engaged community members and sponsored developers to create a first iteration of the feature. This process included the first four of our common development phases: planning, analysis, design, and implementation, but the project then stalled during the testing and deployment phase.

Challenges included not knowing which community members or working groups needed to sign off on the changes; having different individuals within those groups offering divergent feedback; and ultimately having the final gatekeepers of the project fully occupied with other roadmap items for the next Drupal release, causing this feature to miss its initial window for inclusion.

When conversations resumed in the middle of 2022, around the time of the next release window, some of the missing stakeholders were finally engaged, but the feedback retreaded ground already covered in the first design phase and required significant changes. The sponsored contributors who had done a majority of the work were no longer available, and the community contributors could not respond to all the feedback in time for this next window.

When we reached early 2023, we then had the attention of more crucial stakeholders, including the release managers who will make the ultimate call to include the feature. Crucially, the Drupal Association was also able to designate a staff member to respond to stakeholder feedback in much faster time. However, yet again the Design phase was revisited, and the implementation repeated yet again to accommodate feedback that had changed still further from 2021.

Slack conversations

Drupal Slack https://drupal.slack.com/ is used for communication in the Drupal community and currently has 1,098 public channels. A dedicated channel was created for the Project Messaging Initiative and used at various times to reach out to key stakeholders (Figure 2).

Slack discussions summary
Started: 11 August 2021.
Last data export: 30 May 2023.
Channel url: https://app.slack.com/client/T06GX3JTS/C02AFHAUNET
Total messages: 909.
Total users/participants: 47.

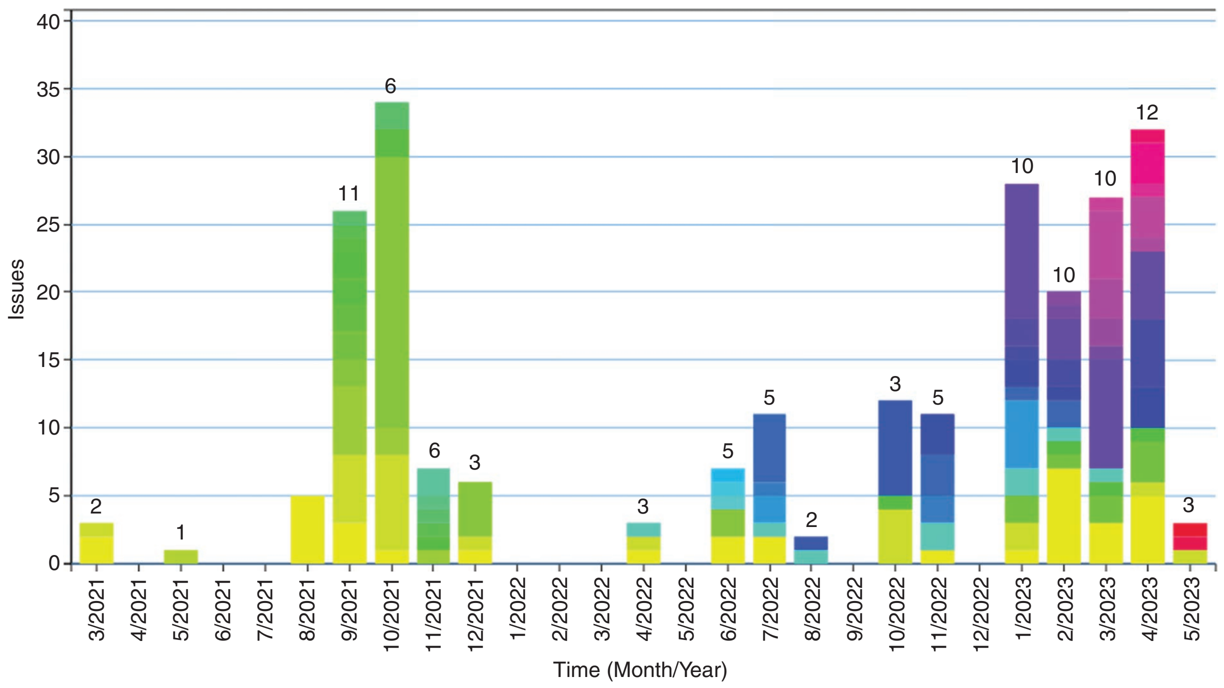

FIGURE 1. Distribution of comments over two years (color coded by Drupal.org user) showing total number of participants per month on top of each bar.

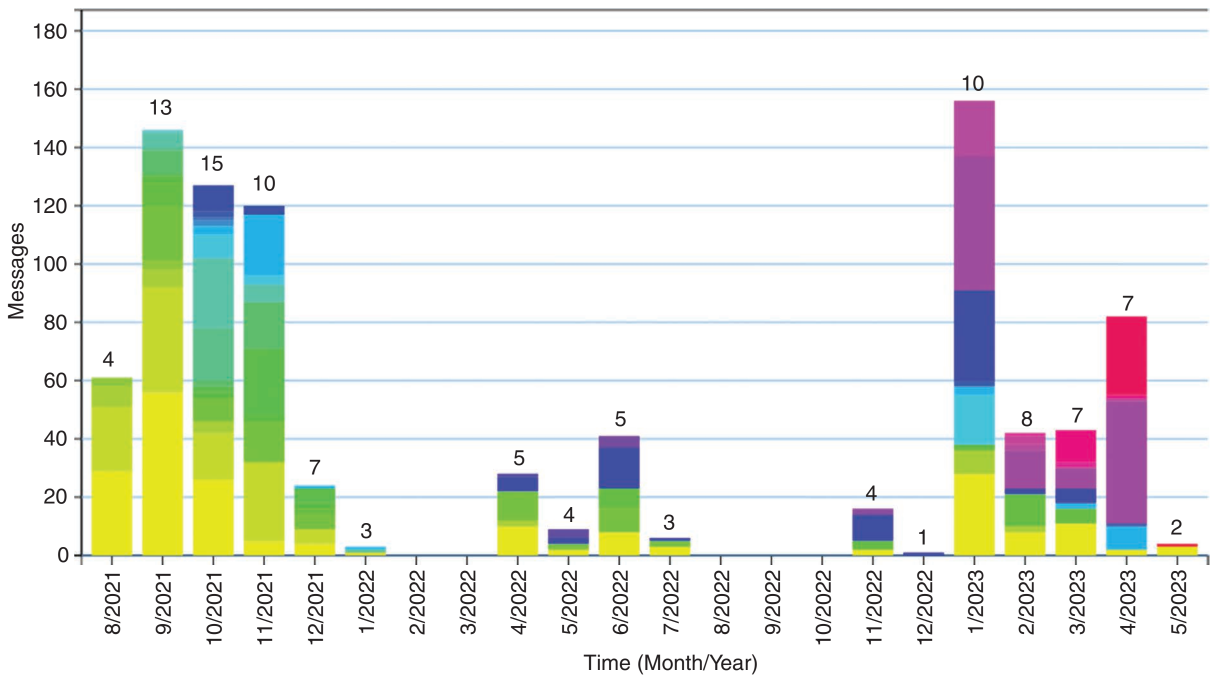

FIGURE 2. Distribution of messages in slack conversation over two years (color coded by slack user showing total number of participants per month on top of each bar).

Data source. https://drive.google.com/drive/folders/1cxfXtmcfka34uKKRE96iS-f2ORtzFKku

Commits on git.drupalcode.org

Commits summary
Started: August 2022.
Completed: April 2023.
Issue url: https://git.drupalcode.org/issue/drupal-3206643/
Total MR: 5.
Final MR https://git.drupalcode.org/project/drupal/-/merge_requests/2889/commits, JSON data https://git.drupalcode.org/api/v4/projects/59858/merge_requests/2889/commits?page=1&per_page=100

Total Commits: 126.
Total users/participants: 8.

Contribution to the Project Announcements Initiative in 2021 happened entirely in a contributed module. In Drupal terms, a contributed is a community-created Drupal extension but must be installed separately. It is common in Drupal to see major features developed this way before being nominated for inclusion in Drupal core.

The commit data are of limited utility in understanding the development lifecycle of this initiative. At best, it shows a sharp uptick in activity after the work has passed through the "core acceptance gates"—as final adjustments and changes are made for the commit to core (Figure 3) .

OVERLAYING EACH CHANNEL, TOGETHER WITH KEY MILESTONES

Putting each channel of contribution activity together, we see a consistent pattern of large amounts of real-time collaboration (Slack) correlated with issue comments (canonical recording of decisions) and then, ideally, commits.

The time period just before and after DrupalCon North America and DrupalCon Europe are correlated with a high frequency of activity. A sprint of communication activity (but without resources to make commits) was insufficient to hit the minor release target in 2022. Examining the data more closely, we find key contributors and gatekeepers became unavailable because of the close proximity to the next minor release cycle.

The renewed effort in 2023, with the highest amount of real-time collaboration, dedicated developer resources to make commits, and enough of a time horizon before the next minor release finally got the feature across the finish line.

It also shows a strong correlation between real-time communication in Slack and progress in the issue itself as displayed in the chart in Figure 4. Notably, *more* users, including more of the key stakeholders were present earlier in the Slack conversations than in the issue.

Different channels are used more in different phases of the project. Real-time communication (in this case through Slack) was reliably needed all the way to the point of the feature finally being committed to core. Work

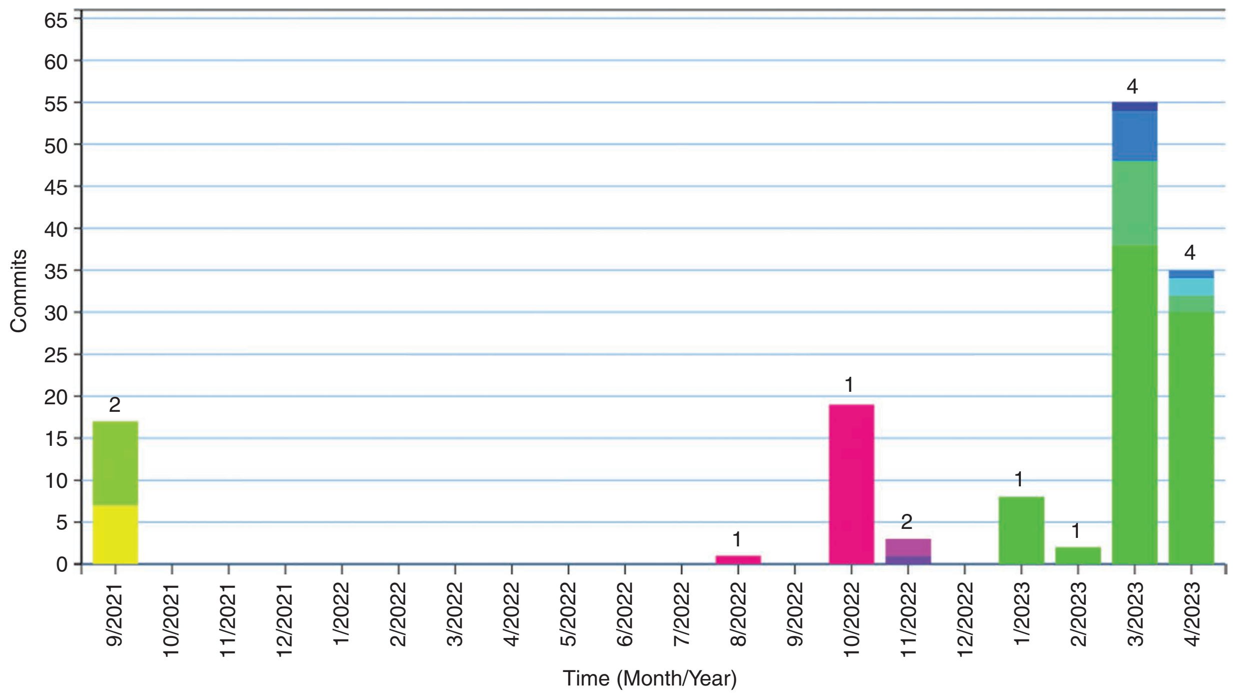

FIGURE 3. Distribution of commits in all MRs over two years (color coded by GitLab users showing total number of participants per month on top of each bar).

in the issue queue was most vital to record decisions of subsystem maintainers, the gatekeepers on whether a feature can reach the release managers. Commit activity was not a reliable indicator of progress by itself, as even the large amount of early activity in 2021 ultimately had to be redone in 2023.

INTERPRETATION

When this initiative was first prioritized in 2021, we saw all the ingredients that we would expect to be required to do the project right.

- The project did not skip ahead of the planning or analysis phases.
- Requirements were built in discussion with subsystem maintainers who would ultimately be the gatekeepers to the features acceptance.
- Comprehensive design assets were created and circulated to those same maintainers to try and secure approval in advance before implementation time was wasted.

Both sponsored and volunteer work was engaged to develop a complete module in the contributed ecosystem space.

In 2022, Drupal community successfully engaged the correct stakeholders, but the process again failed because the level of feedback that would require revisiting the implementation process was much greater detail than expected. User experience and accessibility components that were considered “settled” from the initial review on the design mocks turned out to require significant changes. And because the community did not have enough resources among volunteer, sponsored, or staff contributors, those pieces were not completed before the release window passed.

In 2023, each puzzle piece was finally put together.

- Issues on Drupal.org clearly documented the acceptance gates in the Drupal.org issue, as well as the state of each.
- We *not only* had a great deal of real-time communication to support the effort, that communication was with the *right* stakeholders, and those stakeholders were engaged at a time that they had sufficient bandwidth to participate.
- We had a major contributor (in this case a Drupal Association staff member) who

We plan to extend the model developed in this study to analyze more projects, including both Drupal with composer and Backdrop CMS, a fork of Drupal.

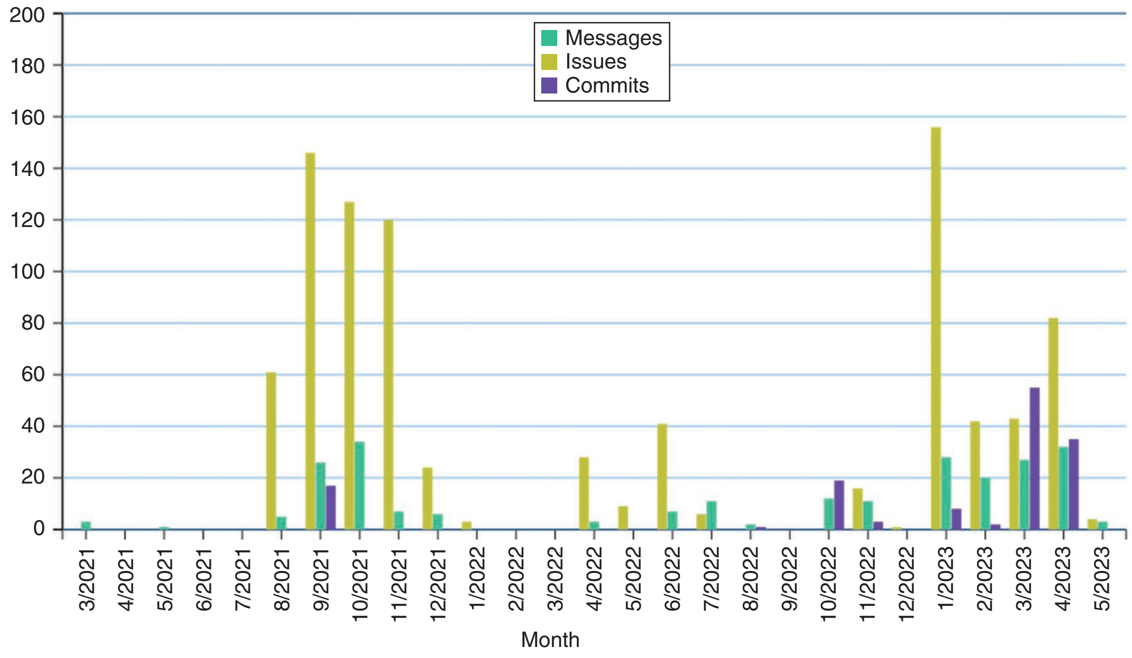

FIGURE 4. The relationship between work happening in different channels and supports our “classic” workflow hypothesis: “consistent pattern of high activity in Slack following with the comments on Drupal.org and then resulting in commits in code.”

was prepared for significant revisions and feedback and was able to implement those quickly enough to get the project over the finish line—before another release window was missed.

maintainers (such as the accessibility subsystem maintainers for example) may not have been adequately documented back in the issue in the first iteration.

The Drupal project messaging module was introduced in 2021 to provide a channel for messaging about the Drupal project within the administrative interface for end users.

We believe that open source projects can learn from this case study and implement tools to help avoid pitfalls that cause progress to stall.

What to watch out for

- Design reviews and core acceptance criteria are not a deterministic process. An open source project is a living thing, and a review done by a working group may change significantly over the course of a single year, or especially two.
- Any contribution left "done but waiting for commit" will inevitably find itself referred back to analysis, design, and implementation again.
- Contributors must be prepared to capitalize on maintainer attention *rapidly*—a great deal of implementation activity happened in both 2021 and 2022, but it nevertheless had to be revised or repeated in detail when we were able to align the resources of the implementer with the reviewer.
- Watch out for retreading old ground, where key design decisions and acceptance criteria from various Drupal maintainers (such as the accessibility subsystem maintainers for example) may not have been adequately documented back in the issue in the first iteration.

How can this be improved?

- Better documentation of core acceptance criteria, but perhaps more critically more tooling to concretely document the current status of a project with respect to these criteria would help avoid a project finding itself in no-man's land between two acceptance gates.
- A stronger process for keeping track of what active issues need to be reviewed and triaging them based on their strategic importance. In the intervening two years, the Drupal project also started a "needs review" initiative which has had a positive impact.
- Better tooling and automation for acceptance gates, where automated tools are extensively used in test, but not for tracking acceptance criteria and notifying key decision makers. This automation would help maintain momentum in driving a major feature or initiative to the finish line.

Next steps

- Use this methodology for other open source projects to identify what can help improve velocity for future projects.
- Refine data analysis with additional metrics like MR review time, about the MR open, time they were waiting for review, time to close, total time open, and so on.
- Measure your project against these metrics and identify changes that will help your project to move faster in the right direction. C

ACKNOWLEDGMENT
This research was carried out with the collaboration and support of the Drupal Association. Researchers interested in studying the Drupal community are encouraged to contact support@drupal.org. The authors would also like to thank Alex Gilgur and Adam Goodman for their support and guidance.

REFERENCES

1. "The open source definition," Open Source Initiative, West Hollywood, CA, USA, Jul. 7, 2006. Accessed: Sep. 5, 2023. [Online]. Available: https://opensource.org/osd/
2. "Getting started." D3.js. Accessed: Sep. 15, 2023. [Online]. Available: https://d3js.org/getting-started

IRINA ZAKS is a cofounder of Stanford Open Source Lab, Palo Alto, CA 94305 USA. Contact her at irina.zaks@stanford.edu.

TIMOTHY LEHNEN is chief technology officer at Drupal Association, Portland, OR 97232 USA. Contact him at tim@association.drupal.org.

AARON ZAKS is a mathematics student at The University of California at Berkeley, Berkeley, CA 94720 USA. Contact him at aaronzaks88@gmail.com.