Culture and Breaking Change: A Survey of Values and Practices in 18 Open Source Software Ecosystems

Christopher Bogart,
Anna Filippova,
Christian Kästner,
James Herbsleb
Carnegie Mellon University, USA

Abstract

Software ecosystems have become one of the most important ways to organize software development, and to maintain and reuse code packages. But coordination can be a major challenge in software ecosystems when packages change, since packages tend to be highly interdependent yet independently maintained. The culture of an ecosystem includes those values and practices associated with managing change. We conducted a survey of thousands of developers in more than a dozen ecosystems, asking them about the values and practices that make up their communities' distinctive cultures; as well as the perceived power of different stakeholders and perceived health of the community. This dataset release shares anonymized data from the survey.

1. INTRODUCTION

Software ecosystems—communities built around a shared programming language, platforms, and dependency management tools—have become one of the most important ways to organize software development, and to maintain and reuse code packages. Ecosystem style development is efficient, since common functionalities need only be developed, maintained, and tested by a single author or team, instead of many authors reimplementing the same functionality.

Coordination is a major challenge in software ecosystems, since packages tend to be highly interdependent yet independently maintained [2, 10, 4, 1]. In particular, sharing the same resources requires coordination when those resources change. Changes one developer makes to a shared package may affect many other people, for example creating new bugs or fixing old ones, introducing new features, or reorganizing or renaming components. Any of these actions may require rework from developers of software that uses that package.

A recent case study by Bogart et al. [2] investigated the role of ecosystem values, change practices, tools, and policies of three such ecosystems (Eclipse, R/CRAN, and Node.js/NPM). The study revealed stark differences among the three ecosystems: They had markedly different priorities (stability, rapid access to current research, and frictionless progress for developers, respectively), and their practices and policies reflected those priorities, for example in how they vet new changes and archive old revisions, how version numbers are specified and referred to, and what responsibilities the authors of packages have when making changes. These results are consistent with studies of open source and corporate *culture* [13, 8, 14, 5, 3, 9]: Culture arises from both values and practices which mutually reinforce each other, and culture has been shown to strongly influence performance [5]. To study culture in software ecosystems in depth (e.g., whether the formation and propagation of culture can be achieved without collocation and frequent informal contact assumed necessary in corporate settings [3]), we need to understand the current values and practices that developers hold within and across ecosystems.

To further this research, we conducted a survey, asking questions of developers in a sample of ecosystems about perceived ecosystem values, personal values, ecosystem health, relative stakeholder power, motivations, and practices, focusing especially on perceptions and practices that would be more difficult to study by mining software artifacts in repositories.

1.1 Survey design

The survey consisted of 108 questions; seven free text questions, three short blanks (ecosystem, package name and gender), and the rest multiple-choice scales. After an informed consent screen, participants first chose from a list of ecosystems, (or could write in another; we simply grouped these as "other" for analysis), then we presented blocks of questions in the following order: values, ecosystem health, stakeholder power, upstream practices, downstream practices, motives, and demographics.

Ecosystems. We considered ecosystems with a *network structure*, in which packages can depend on other packages and some infrastructure helps with sharing and compatibility. We started with a list of software repositories from Wikipedia[1] and added additional ecosystems with an active community that we could find.

We excluded ecosystems with a flat structure where packages depend only on a single shared platform (e.g., we excluded Android, where apps can interact through generic mechanisms rather than absolutely requiring another specific app) and ecosystems obviously too small to hope to get at least a few dozen responses. We also excluded ecosystems if they were different enough from the ones in the original Wikipedia list, that it was not possible to write clear questions that would apply across ecosystems. This excluded operating-system-level package managers like apt, rpm, and brew, and scientific workflow engines.

We conducted the survey with a list of 31 ecosystems. We recruited with a goal of at least 40 participants

DOI: 10.1184/R1/5108716

[1] https://en.wikipedia.org/wiki/Software_repository

from each ecosystem. Afterward, we excluded 13 ecosystems from the analysis for which we did not receive at least 15 complete surveys: C++/Boost, Bower, Perl 6, Smalltalk, Tex/CTAN, Julia, Clojure/clojars, Meteor, Wordpress, SwiftPM, PHP's PEAR, Racket, and Dart/pub, leaving us with 18 ecosystems for our analysis: *Atom (plugins), CocoaPods, Eclipse (plugins), Erlang/Elixir/Hex, Go, Haskell (Cabal/Hackage), Haskell (Stack/Stackage), Lua/Luarocks, Maven, Node.js/NPM, NuGet, Perl/CPAN, PHP/Packagist, Python/PyPi, R/Bioconductor, R/CRAN, Ruby/Rubygems,* and *Rust/Cargo*.

Values. We took the values mentioned in the Bogart et al. study [2] as a starting point then systematically searched the web pages of all ecosystems for clues of other potential values. For example '*fun*' is mentioned as an explicit value in the Ruby community. [2] We assembled a list of 11 values with the following descriptions:

- *Stability:* Backward compatibility, allowing seamless updates ("do not break existing clients")
- *Innovation:* Innovation through fast and potentially disruptive changes
- *Replicability:* Long term archival of current and historic versions with guaranteed integrity, such that exact behavior of code can be replicated.
- *Compatibility:* Protecting users from struggling to find a compatible set of versions of different packages
- *Rapid Access:* Getting package changes through to end users quickly after their release ("no delays")
- *Quality:* Providing packages of very high quality (e.g. good security or correctness)
- *Commerce:* Helping professionals build commercial software
- *Community:* Collaboration and communication among developers
- *Openness* and Fairness: ensuring that everyone in the community has a say in decision-making and the community's direction
- *Curation:* Providing a set of consistent, compatible packages that cover users' needs
- *Fun* and personal growth: Providing a good experience for package developers and users

In the survey, we asked participants separately about the *perceived values* of the community—*"How important do you think the following values are to the <ecosystem> community?"* We used a seven point rating scale, adapted from Schwartz's value study [12]: *"extremely important", "very important", "important", "somewhat important", "not important", "community opposes this value",* and *"I don't know".* The first five options were separated visually from the last two to make clear that only the former were designed to approximate regular intervals (as recommended by Dillman et al. [6]).

In addition, we also asked participants a similar value question on the same scale about their *own values* with respect to a single package they worked on in the ecosystem. To encourage participants to think about concrete work that they are doing we asked for the name of a specific package that they worked on and used that package in the question: *"How important are each of these values in development of <package> to you personally?"*

[2]For example, in an interview Matsumoto said, *That was my primary goal in designing Ruby. I want to have fun in programming myself" [15].*

Health. We asked about *perceived health* of the ecosystem, to investigate whether people think the ecosystem's values are being achieved. The questions were designed to represent failure to satisfy each of the values asked about, when we could frame such a question that made sense (there were no questions tied to *fun* or *replicability*). We also asked if they perceived problems with recruiting or retaining developers to the ecosystem, inspired by the observation in Bogart et al. [2] that Eclipse might be having trouble retaining developers. The question asked about health in the last 6 months on a 5 point agreement scale and was phrased negatively (e.g. *"Package interfaces are generally too unstable in <ecosystem>"*).

Power. We asked about *powerful stakeholders* of the ecosystem, to investigate whether community norms placed less burden on the most powerful stakeholders. We asked about *end users*, *developers*, *leaders*, *gatekeepers*, and *sponsors*, and whether each had more influence than others over the direction of the ecosystem. We also asked if they believed the distribution of influence was "fair", and asked two open-ended questions to pick up groups or situations we might not have anticipated.

Practices. In the *practices* part of the survey asked about many software engineering practices. The full list and exact phrasing of our questions can be found in Appendix A. Surveyed practices encompassed the participant's personal practices and experiences with respect to documentation, support, timing, and version numbering for releases, selecting packages on which to depend, and monitoring dependencies for changes. These were asked either on an agreement Likert scale as above or on a frequency scale from *"never"* to *"several times a day"*. A subset of 15 questions relating to communication with developers of downstream packages were skipped for participants who indicated that they did not maintain a package used by others. To limit the length of the survey, we focused primarily on questions that cannot be answered or are difficult to answer by mining software repositories.

Demographics, motivation. We also asked *demographics* questions asking the participants' age, gender, role in the ecosystem, experience in open source and in software development generally, and computer-science education level. We also adapted almost verbatim a battery of questions taken from Roberts et al. [11] designed to distinguish three kinds of motivations for participation in open source: intrinsic, instrumental, and status-based.

1.2 Recruitment

We invested in significant outreach activities to recruit participants for the survey. First, we created a web page and twitter account to describe the state of current research in this area, in a form easily accessible to practitioners. We encouraged readers of the web page to take the survey to contribute additional knowledge about values in ecosystems. Second, we attended community events, including *npm.camp 2016*, to talk to developers and community leaders from multiple ecosystems about our research; several prominent community members tweeted about our web page and survey, resulting in surges of responses (CRAN and NPM particularly). Third, we promoted our web page and the survey in ecosystem-specific forums and mailing lists to *"developers who write <ecosystem> packages,"* hoping that our web page would spark interest in the topic. We also posted on twitter with hashtags ap-

propriate for different ecosystems. Finally, for 21 ecosystems in which our outreach activity did not yield sufficient answers, we solicited individuals directly by email. After checking with Github to make sure our practices conformed with their terms of service, we sent 8,137 emails to package authors. We sampled these from packages in various ecosystems as culled from libraries.io, sending them a survey invitation if their package was shared on Github, they had chosen to make their emails public on that site, and they had not blocked their email address from ghtorrent.org's[7] Github archive.[3].

Participants and their demographics. We succeeded in recruiting 2321 participants to take the survey between August and November of 2016. 932 of them completed the survey; however, we put value and health questions near the beginning, so there are more than 1200 answers to those questions. Statistical analysis of answers to early questions did not reveal any systematic differences between people who completed the survey and those who did not.

Respondees averaged 8.8 years of development experience, 7.2 years in open source, and 4.6 in the ecosystem they answered about. Slightly more than half (59%) had college degrees in CS. The most claimed role in the ecosystem was package lead developers (59%); Others ranged from the 8.5% who claimed a role in the founding or core team of the ecosystem, to 11% who only drew on ecosystem packages for their own projects. The average age was 33, with 152 18-24 year olds, and 6 over 65. Of those who gave their gender, 95.9% identified themselves as male, 3.2% as female, and 0.8% gave another gender.

1.3 Display statistics

Figures 1, 2, 3 and 4 show the distributions of answers for selected survey questions. The figures were drawn by eliminating skipped or "don't know" values, merging "Not important" with "opposed to this value" answers, and drawing a violin plot, with a diamond symbol at the mean position. The violin bodies are smoothed, so the image portrays the mean and only a rough distribution.

1.4 Anonymization

Because the survey included several demographic questions and asked about smallish software ecosystems, it could be possible to individually identify some people who took the survey. To protect their anonymity, we have made the following modifications to the data, merging options and omitting questions so that unique and identifiable combinations of answers do not occur.

- *Ecosystem* changed to "other" if fewer than 15 people completed the survey
- *Ecosystem role* We have merged the "founder" and "lead+" roles into a single category, "central"
- *Years of experience in ecosystem, OSS, and software development* For all these questions we have merged answers into two categories: "<5 yrs", "5+ years"
- *Package name* We omit the name of the package the user chose to answer questions about
- *Age and gender* We have omitted these; most respondents were young and male, so other responses become an identification risk

We also exclude the free text fields, many answers gave identifying information or wrote in a distinctive style.

If you need the omitted data for research, please contact us and we can discuss mutual interests and the process for applying for IRB[4] approval.

[3]We did not use githubtorrent.org's dataset except to extract this information

2. THREATS TO VALIDITY

As is typical of a survey, our sample could not be truly random; there may be selection bias relating to who we were able to reach via the venues we chose. We tried to mitigate this by sampling from forums, twitter, and direct mail. The survey was also quite long (and was advertised as such up front); people with less patience for long surveys, or less interest in questions of breaking changes, values, and practices, may have self-selected out. This could be significant if people with impatience for long surveys also have different software engineering practices and beliefs.

Another possible concern is that respondents may not have a broad perspective on what practices and values exist among ecosystems; they may rate practices as common or uncommon in their ecosystem, but this may be an ecosystem-specific notion about what standard they are comparing it against. For example an ecosystem that highly values stability may be more stable than others, but rate their ecosystem as unstable because their standards are so high. Future studies could calibrate such expectations against change data mined from repositories in the ecosystems.

We had difficulty recruiting sufficient participants from smaller ecosystems, like Perl 6 or Clojure; small ecosystems may have different characteristics than large ones. We do have some small ecosystems, Stackage and Lua, and they are outliers in some ways. So further exploration of small ecosystems, for example with interviews or analysis of artifacts, should be a priority for future work.

3. ACKNOWLEDGMENTS

This work was supported by NSF awards 1546393, 1302522, 1322278, 1111750, 0943168, 1318808 and 1552944, the Science of Security Lablet (H9823014C0140), the U.S. Department of Defense through the Systems Engineering Research Center, LARC (Living Analytics Research Centre), a grant from the Alfred P. Sloan Foundation, the Google Open Source Program Office. We also want to thank the many volunteers who responded to our survey.

References

[1] Cyrille Artho, Kuniyasu Suzaki, Roberto Di Cosmo, Ralf Treinen, and Stefano Zacchiroli. 2012. Why Do Software Packages Conflict? 141–150.

[2] Christopher Bogart, Christian Kästner, James Herbsleb, and Ferdian Thung. 2016. How to Break an API: Cost Negotiation and Community Values in Three Software Ecosystems. In *Proc. Int'l Symposium Foundations of Software Engineering (FSE)*. ACM Press, New York.

[3] John Coleman. 2013. Six components of a great corporate culture. *Harvard Business Review* 5, 6 (2013), 2013.

[4] Alexandre Decan, Tom Mens, Maëlick Claes, and Philippe Grosjean. 2016. When GitHub meets CRAN:

[4]IRB = Institutional Review Board, the ethics committee that oversees research to protect the privacy and safety of participants.

An Analysis of Inter-Repository Package Dependency Problems. *International Conference on Software Analysis, Evolution, and Reengineering* (2016), 493–504. DOI: http://dx.doi.org/10.1109/SANER.2016.12

[5] Daniel R Denison. 1990. *Corporate culture and organizational effectiveness*. John Wiley & Sons.

[6] Don A Dillman, Jolene D Smyth, and Leah Melani Christian. 2014. *Internet, phone, mail, and mixed-mode surveys: the tailored design method*. John Wiley & Sons.

[7] Georgios Gousios. 2013. The GHTorrent dataset and tool suite. In *Mining Software Repositories*. IEEE Press, Piscataway, NJ, USA, 233–236. http://dl.acm.org/citation.cfm?id=2487085.2487132

[8] Eric von Hippel and Georg von Krogh. 2003. Open source software and the "private-collective" innovation model: Issues for organization science. *Organization science* 14, 2 (2003), 209–223.

[9] John P Kotter. 1992. *Corporate culture and performance*. Simon and Schuster.

[10] Romain Robbes, Mircea Lungu, and David Röthlisberger. 2012. How do Developers React to API Deprecation? The Case of a Smalltalk Ecosystem. In *Proc. Int'l Symposium Foundations of Software Engineering (FSE)*. ACM Press, New York, Article 56, 11 pages. DOI:http://dx.doi.org/10.1145/2393596.2393662

[11] Jeffrey a. Roberts, Il-Horn Hann, and Sandra a. Slaughter. 2006. Understanding the Motivations, Participation, and Performance of Open Source Software Developers: A Longitudinal Study of the Apache Projects. *Management Science* 52, 7 (2006), 984–999. DOI:http://dx.doi.org/10.1287/mnsc.1060.0554 arXiv:arXiv:1011.1669v3

[12] Shalom H Schwartz. 1992. Universals in the content and structure of values: Theoretical advances and empirical tests in 20 countries. *Advances in Experimental Social Psychology* 25 (1992), 1–65.

[13] Srinarayan Sharma, Vijayan Sugumaran, and Balaji Rajagopalan. 2002. A framework for creating hybrid-open source software communities. *Information Systems Journal* 12, 1 (2002), 7–25.

[14] Katherine J. Stewart and Sanjay Gosain. 2006. The Impact of Ideology on Effectiveness in Open Source Software Development Teams. *MIS Q.* 30, 2 (June 2006), 291–314. http://dl.acm.org/citation.cfm?id=2017307.2017313

[15] Bill Venners. 2003. The Philosophy of Ruby: A Conversation with Yukihiro Matsumoto, Part I. http://www.artima.com/intv/rubyP.html. (2003).

APPENDIX

A. LIST OF QUESTIONS

For transparency and replicability, we list all evaluated questions of the survey including their exact phrasing. We exclude a small number of questions about power structures, community health, and motivation that we have not used in this paper.

Part I: Ecosystem.

- Please choose ONE software ecosystem* in which you publish a package**. If you don't publish any packages, then pick an ecosystem whose packages you use.
  * "Software ecosystem" = a community of people using and developing packages that can depend on each other, using some shared language or platform
  ** "Package": A distributable, separately maintained unit of software. Some ecosystems have other names for them, such as "libraries", "modules", "crates", "cocoapods", "rocks" or "goodies", but we'll use "package" for consistency.
  [selection or textfield, substituted for <ecosystem> in remainder of survey]

Ecosystem Role.

- Check the statement that best describes your role in this ecosystem.
  - I'm a founder or core contributor to <ecosystem> (i.e. its language, platform, or repository).
  - I'm a lead maintainer of a commonly-used package in <ecosystem>.
  - I'm a lead maintainer of at least one package in <ecosystem>.
  - I have commit access to at least one package in <ecosystem>.
  - I have submitted a patch or pull request to a package in <ecosystem>.
  - I have used packages from <ecosystem> for code or scripts I've written.
- About how many years have you been using <ecosystem> in any way?
  - < 1 year
  - 1 - 2 years
  - 2 - 5 years
  - 5 - 10 years
  - 10 - 20 years
  - > 20 years

Ecosystem values.

- "How important do you think the following values are to the <ecosystem> community? (Not to you personally; we'll ask that separately.)" — see Section 1.1 for the 11 value questions; **results shown in Figure 1**.
- How confident are you in your ratings of the values of <ecosystem> above?
  - Not confident
  - Slightly confident
  - Confident
  - Very confident
- "Is there some other value the <ecosystem> community emphasizes that was not asked above? If so, describe it here:"

Important groups of participants.

- "In some ecosystems, some participants have more influence than others over the direction of the ecosystem. In the <ecosystem> ecosystem, rate which groups are more or less likely to get what they need or want." **The following choices were presented**
  - Highest influence
  - High influence
  - Medium influence
  - Low influence
  - Lowest influence
  - I Don't Know/Not Applicable

  for each of these five stakeholder groups.
  - *End users* People who rely on the ecosystem's packages, but do not necessarily contribute packages themselves
  - *Developers* People who write or maintain packages
  - *Leaders* A small group of people who organize the community and set policies
  - *Gatekeepers* People who decide if submitted packages will be made available
  - *Sponsors* Organizations that supply developers, code, tools, money, etc. to the community
- "Thinking about who has influence and gets what they want and need: do you agree that the situation is fair?"
  - Strongly agree
  - Somewhat agree
  - Neither agree nor disagree
  - Somewhat disagree
  - Strongly disagree
- (OPTIONAL) Why or why not? – [text field]
- (OPTIONAL) If there are other people or groups in <ecosystem> who are highly influential, please name them. – [text field]

Community Health.

- "Next, we would like to know your perceptions about the health of the <ecosystem> ecosystem. Please focus on the last 6 months and use the radio buttons below to indicate your level of agreement to the following" **The following choices were presented**
  - Strongly agree
  - Somewhat agree
  - Neither agree nor disagree
  - Somewhat disagree
  - Strongly disagree
  - I don't know

  for each of the following questions:
  DEVELOPERS
- <ecosystem> has problems attracting and retaining new developers lately.
- There is a pattern of long- standing developers leaving<ecosystem>.
- Developers who want to innovate and make larger changes are often too constrained from doing so.
- Developers are rather isolated and do not communicate with each other much.
- Diversity among developers is low.
  PACKAGES
- Package interfaces are generally too unstable in <ecosystem>.
- It can be difficult to find appropriate packages within the ecosystem, because many packages offer confusingly similar functionality.
- <ecosystem> is perceived as having low quality packages.

- Package changes do not get made available to end users quickly enough after their release.
- Users often struggle to find a consistent set of compatible versions of dependencies.

PEOPLE

- People in the community do not have enough say in decisionmaking. (e.g. about policies, leadership, criteria for inclusion, etc.)
- <ecosystem> is not very attractive to people building commercial software.

Part II: Package.

- In the following we are going to ask about your experience working on one particular package. Please think of one package in <ecosystem> you have contributed to recently and are most familiar with. If you haven't contributed to a package in <ecosystem>, then name some software you've written that relies on packages in <ecosystem> packages. You may use a pseudonym for it if you are concerned about keeping your responses anonymous. — [text fields, substituted for <package> in remainder of survey]
- Do you submit the package you chose to a/the repository associated with <ecosystem>? (Choose "no" if the ecosystem does not have its own central repository.) — [yes/no]
- Is there any software maintained by other people that depends on the package you chose? — [yes/no]
- Is the package you chose installed by default as part of a standard basic set of packages or platform tools? — [yes/no]
- "How important are each of these values in development of <package> to you personally?" — see Section 1.1 for the 11 value questions; **results shown in Figure 2**.
- (OPTIONAL) Is there some other value important to you personally for <package> which was not mentioned? — [text fields]
- How often do you face breaking changes from any upstream dependencies (that require rework in <package>)? — **results shown in Figure 4a**
  - Never
  - Less than once a year
  - Several times a year
  - Several times a month
  - Several times a week
  - Several times a day
- How often do you make breaking changes to <package>? (i.e. changes that might require end-users or downstream packages to change their code) — [frequency scale as above] **results shown in Figure 3a**

Making changes to <package>.

- I feel constrained not to make too many changes to <package> because of
- potential impact on users. — **results shown in Figure 3b**
  - Strongly agree
  - Somewhat agree
  - Neither agree nor disagree
  - Somewhat disagree
  - Strongly disagree
  - I don't know
- I know what changes users of <package> want. — [agreement+don't know scale as above]
- If I have multiple breaking changes to make to <package>, I try to batch them up into a single release. — [agreement+don't know scale as above] **results shown in Figure 3d**
- I release <package> on a fixed schedule, which <package> users are aware of. — [agreement+don't know scale as above] **results shown in Figure 3j**
- Releases of <package> are coordinated or synchronized with releases of packages by other authors. — [agreement+don't know scale as above] **results shown in Figure 3i**
- When working on <package>, I make technical compromises to maintain backward compatibility for users. — [agreement+don't know scale as above] **results shown in Figure 3c**
- When working on <package>, I often spend extra time working on extra code aimed at backward compatibility. (e.g. maintaining deprecated or outdated methods) — [agreement+don't know scale as above]
- When working on <package>, I spend extra time backporting changes, i.e. making similar fixes to prior releases of the code, for backward compatibility. — [agreement+don't know scale as above]

Releasing Packages.

- A large part of the community releases updates/revisions to packages together at the same time. — [agreement+don't know scale as above]
- A package has to a meet strict standards to be accepted into the repository. — [agreement+don't know scale as above] **results shown in Figure 3k**
- Most packages in <ecosystem> will sometimes have small updates without changing the version number at all. — [agreement+don't know scale as above]
- Most packages in <ecosystem> with version greater than 1.0.0 increment the leftmost digit of the version number if the change might break downstream code. — [agreement+don't know scale as above]
- I sometimes release small updates of <package> to users without changing the version number at all. — [agreement scale, without 'don't know'] **results shown in Figure 3g**
- For my packages whose version is greater than 1.0.0, I always increment the leftmost digit if a change might break downstream code (semantic versioning). — [agreement as above] **results shown in Figure 3f**
- When making a change to <package>, I usually write up an explanation of what changed and why (a change log). — [agreement as above] **results shown in Figure 3e**
- When working on <package>, I usually communicate with users before performing a change, to get feedback or alert them to the upcoming change. — [agreement as above] **results shown in Figure 3h**
- When making a breaking change on <package>, I usually create a migration guide to explain how to upgrade. — [agreement as above]
- After making a breaking change to <package>, I usually assist one or more users individually to upgrade. (e.g. reaching out to affected users, submitting patches/pull requests, offering help) — [agreement as above]

Part IV: Dependencies.

- In the last 6 months I have participated in discussions, or made bug/feature requests, or worked on develop-

ment of another package in <ecosystem> that one of my packages depends on. — [yes/no]
- Have you contributed code to an upstream dependency of one of your packages in the last 6 months (one where you're not the primary developer)? — [yes/no]
- About how often do you communicate with developers of packages you depend on (e.g. participating in mailing lists, conferences, twitter conversations, filing bug reports or feature requests, etc.)? — [frequency scale, as above] **results shown in Figure 4f**

For most dependencies that my packages rely on, the way I typically become aware of a change to the dependency that might break my package is:

- I read about it in the dependency project's internal media (e.g. dev mailing lists, not general public announcements) — [agreement scale, as above]
- I read about it in the dependency project's external media (e.g. a general announcement list, blog, twitter, etc) — [agreement scale, as above]
- A developer typically contacts me personally to bring the change to my attention — [agreement scale, as above] **results shown in Figure 4e**
- Typically I get a notification from a tool when a new version of the dependency is likely to break my package — [agreement scale, as above] **results shown in Figure 4f**
- Typically, I find out that a dependency changed because something breaks when I try to build my package. — [agreement scale, as above] **results shown in Figure 4g**
- How do you typically declare the version numbers of packages that <package> depends — **results shown in Figure 4i**
  - I specify an exact version number
  - I specify a range of version numbers, e.g. 3.x.x, or [2.1 through 2.4]
  - I specify just a package name and always get the newest version
  - I specify a range or just the name, but I take a snapshot of dependencies (e.g. shrinkwrap, packrat)
- What is the common practice in <ecosystem> for declaring version numbers of dependencies? — [same scale as previous + "don't know"]

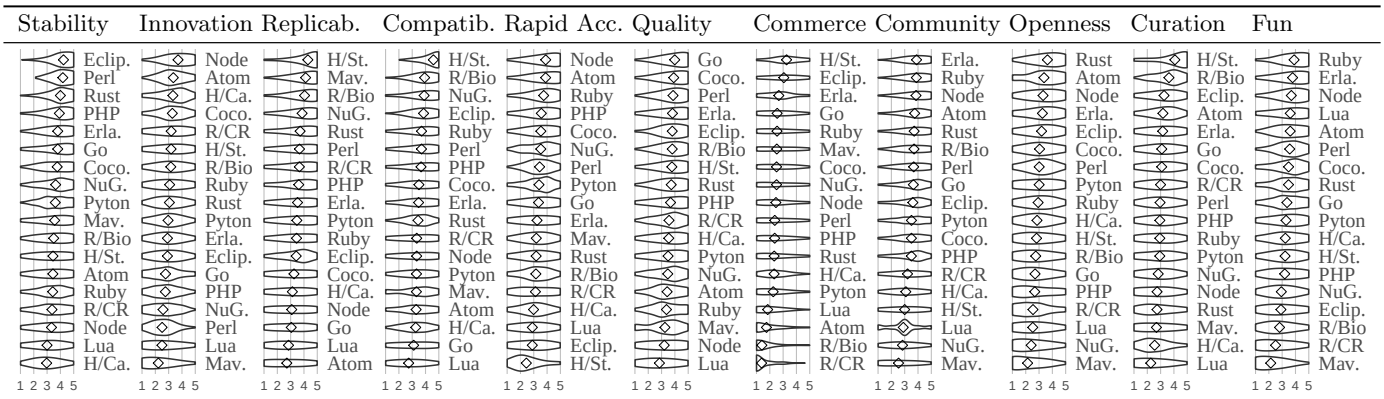
Using or avoiding dependencies.

- When adding a dependency to <package>, I usually do significant research to assess the quality of the package or its maintainers, before relying on a package that seems to provide the functionality I need. — [agreement scale, as above] **results shown in Figure 4d**
- It's only worth adding a dependency if it adds a substantial amount of value. — [agreement scale, as above] **results shown in Figure 4c**
- I often choose NOT to update <package> to use the latest version of its dependencies. — [agreement scale, as above] **results shown in Figure 4h**
- When adding a dependency, I usually create an abstraction layer (i.e., facade, wrapper, shim) to protect internals of my code from changes. — [agreement scale, as above]
- When working on <package>, I often copy or rewrite segments of code from other packages into my package, to avoid creating a new dependency. — [agreement scale, as above]
- When working on <package>, I must expend substantial effort to find versions of all my dependencies that will work together. — [agreement scale, as above]
- (OPTIONAL) Compare <ecosystem> with other ecosystems you've used or heard about – does one have some features that the other should adopt? If so, name the other ecosystem(s) and describe the feature(s). — [text field]
- (OPTIONAL) Why do you think people chose to design these other ecosystem(s) differently from <ecosystem>? — [text field]

Part V: Demographics and motivations.

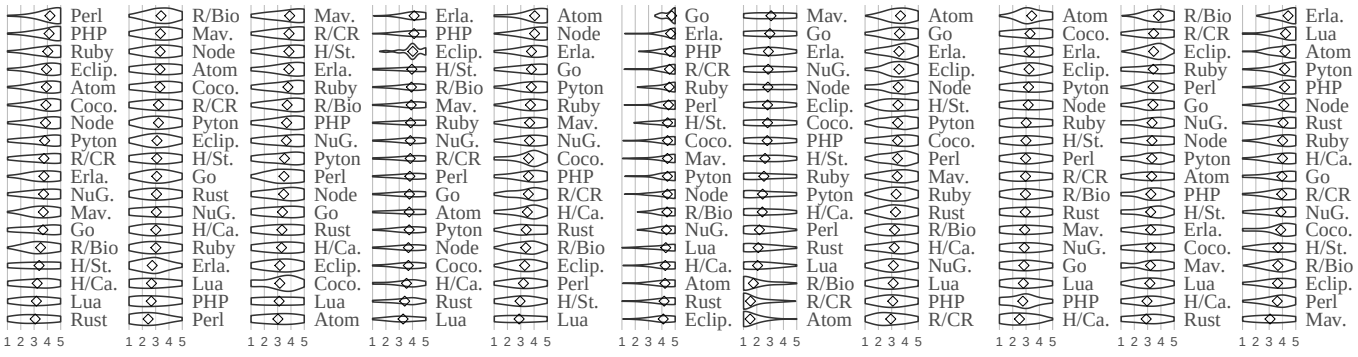
- Age
  - 18-24
  - 25-34
  - 35-44
  - 45-54
  - 55-64
  - 65+
- Gender — [male/female/other]
- Formal computer science education/training
  - None
  - Coursework
  - Degree
- How many years have you been contributing to open source? (in any way, including writing code, documentation, engaging in discussions, etc) — [same time scale as "years used ecosystem" above]
- How many years have you been developing or maintaining software? — [same as previous]
- "People work on software for many different reasons. They also derive different kinds of satisfaction from such work. Following are some reasons other developers have given us regarding their participation in open source projects. Thinking of your own work on <package>, please indicate how important each of these motivations is to you:" **Each of these questions was presented with a five point scale from Great Importance to Little Importance, with intermediate choices unmarked**.
  - i really enjoy it. It is fun.
  - It gives me the chance to attain a recognized qualification or skill.
  - It gives me status at work.
  - It increases my opportunities for a better job.
  - It is the satisfaction of seeing the results of the work I do.
  - It gives me a sense of personal achievement.
  - I can add features I want or need to use.
  - It gives me status in the community.
  - It's part of my job.
  - It gives me the chance to do things I am good at.
  - I can fix bugs or problems that cause me trouble.
- (OPTIONAL) Is there anything else we should have asked, that would help us better understand your experience with community values and breaking changes in <ecosystem> If so, tell us about it: — [text field]

B. SUMMARY PLOTS OF SELECTED RESULTS



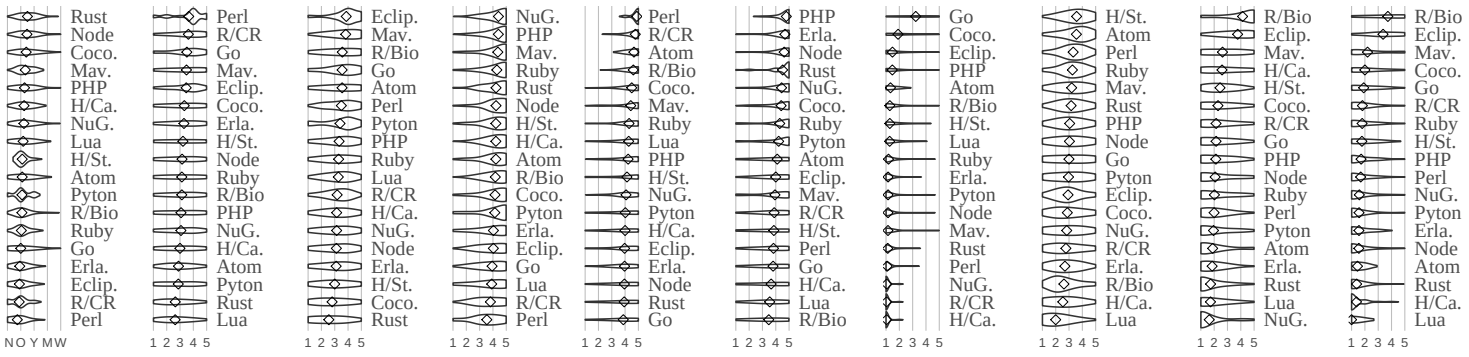
1: not important or opposed, 2: somewhat important, 3: important, 4: very important, 5: extremely important; plots show smoothed distribution; diamond indicates mean

Figure 1: Perceived community values; showing distribution of raw ratings, sorted by average to emphasize range of answers.



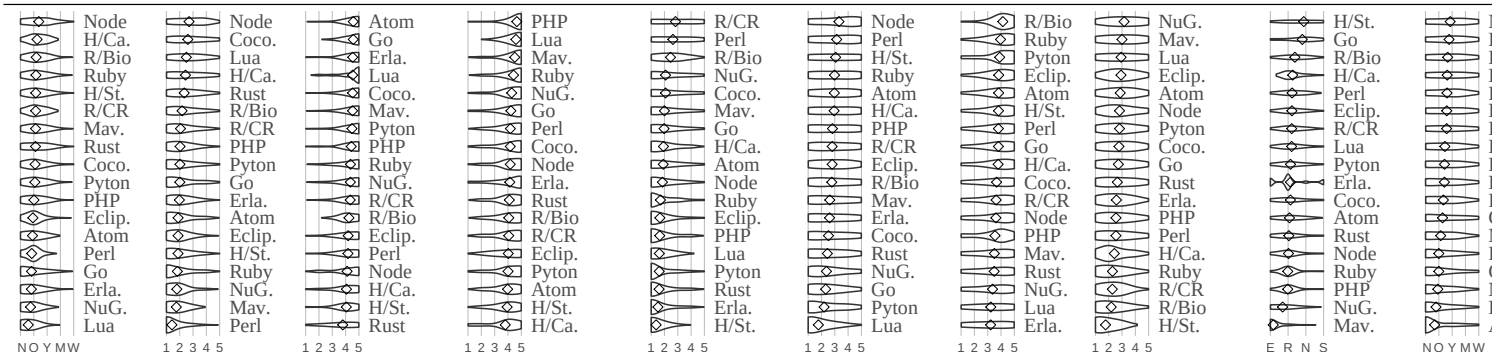
1: not important or opposed, 2: somewhat important, 3: important, 4: very important, 5: extremely important; plots show smoothed distribution; diamond indicates mean

Figure 2: Personal values.



N: never, O: less than once a year, Y/M/W: several times a year/month/week; 1: strongly disagree; 3: neither agree nor disagree; 5: strongly agree

Figure 3: Practices of package maintainers and frequency of performing breaking changes.



N: never, O: less than once a year, Y/M/W: several times a year/month/week; 1: strongly disagree; 3: neither agree nor disagree; 5: strongly agree;
E: exact version number; R: version range; N: just by name; S: snapshot

Figure 4: Practices of package users and frequency of facing breaking changes.