Mapping Software Ecosystems: A Survey of Values and Practices in 18 Ecosystems Appendices Only

Anonymous Author(s)

APPENDIX A: SURVEY TEXT

For transparency and replicability, we list all evaluated questions of the survey including their exact phrasing. We exclude a small number of questions about community health and motivation that we have not used in this paper. When <ecosystem> or <package> appears, the text was substituted for the participant's own selected ecosystem and package.

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 **??** for the 11 value questions; **results shown in Figure ??**.
- 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 toget what they need or want.
  - End users: People who rely on the ecosystem's packages, but do not necessarily contribute packages themselves
    - Highest influence
    - High influence
    - Medium influence
    - Low influence
    - Lowest influence
    - I don't know/Not Applicable
  - Developers: People who write or maintain packages [influence scale as above]
  - Leaders: A small group of people who organize the community and set policies [influence scale as above]
  - Gatekeepers: People who decide if submitted packages will be made available [influence scale as above]
  - Sponsors: Organizations that supply developers, code, tools, money, etc. to the community [influence scale as above]
- 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]

FSE 2018, Lake Buena Vista, FL USA
2016. 978-x-xxxx-xxxx-x/YY/MM…$15.00
DOI: 10.1145/nnnnnnn.nnnnnnn

- 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: [agreement scale + I don't know]
  - ——**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 **??** for the 11 value questions; **results shown in Appendix ??**.
- (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 ??a**
  - 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 ??a**

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 ??b** [agreement scale as above plus "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 ??d**
- I release <package> on a fixed schedule, which <package> users are aware of. — [agreement+don't know scale as above] **results shown in Figure ??j**
- 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 ??i**
- 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 ??c**
- 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 ??k**
- 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 ??g**
- 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 ??f**
- 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 ??e**
- 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 ??h**
- 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 development 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 ??f**

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 ??e**
- 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 ??f**
- 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 ??g**
- How do you typically declare the version numbers of packages that <package> depends — **results shown in Figure ??i**
  - 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.

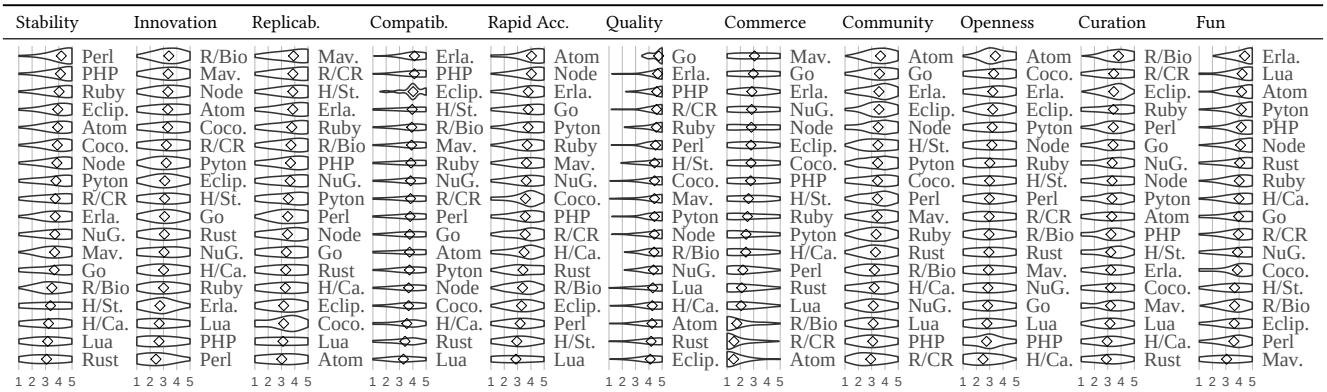
- When adding a dependency to <package>, I usually do significant research to assess the quality of the package or its maintainers, before relying on a package that seems to provide the functionality I need. — [agreement scale, as above] **results shown in Figure ??d**
- It's only worth adding a dependency if it adds a substantial amount of value. — [agreement scale, as above] **results shown in Figure ??c**
- I often choose NOT to update <package> to use the latest version of its dependencies. — [agreement scale, as above] **results shown in Figure ??h**
- 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: [Five choices, with just the ends labeled: Great importance . . . Little importance]
  - It is the satisfaction of seeing the results of the work I do.
  - It gives me the chance to do things I am good at.
  - I really enjoy it. It is fun.
  - It gives me a sense of personal achievement.
  - 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 gives me status in the community.
  - I can fix bugs or problems that cause me trouble.
  - I can add features I want or need to use.
  - It's part of my job.
- (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]

Appendix B: Personal Values



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

Appendix C: Recommended revised list of values

Stability: Packages should be backward compatible, 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")
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
Usability: Ensuring that tools and procedures for developers are easy to use
Standardization: Limiting choice of working styles and standardizing libraries, to prevent incompatibility and duplication of effort
Technical diversity: Making it easy for developers to work with a wide variety tools, processes, and libraries
Social Benevolence: Empowering programmers, making resources available to all users, and behaving with altruism