# Metrics With Greater Utility: The Community Manager Use Case

Sean P. Goggins

November 14, 2018

## 1 Introduction

Community managers take a variety of perspectives, depending on where their communities are in the lifecycle of growth, maturity and decline. This is an evolving report of what we are learning from community managers, some of whom we are working with on live experiments with a CHAOSS project prototyping software tool called Augur (http://www.github.com/CHAOSS/augur). At this point we are paying particular focus to how community managers consume metrics and how the presentation of open source software health and sustainability metrics could make them more and in some cases less useful for doing their jobs.

Right now, based on Augur prototypes and follow up discussions so far, we have the following observations that will inform our work both the the "Growth Maturity and Decline" working group and in Augur Development. There are a few things we have learned from prototyping Augur with community managers. These features in Augur are particularly valued:

1. Allowing comparisons with projects within a defined universe is essential
2. Allow community managers to add and remove repositories that they monitor from their repertoires periodically.
3. Downloadable graphics
4. Downloadable data (.csv or .json)
5. Availability of a "Metrics API", limiting the amount of software infrastructure the CM needs to maintain for themselves. This is more valued by program managers overseeing larger portfolios right now, but we think has potential to grow as awareness of the relatively light weight of this approach becomes more apparent. By apparent, we really mean "easy to use and understand"; right now it is for a programmer, but less so for a community manager without this background or current interest.

## 2 Date Summarized Comparison Metrics

With these advantages in mind, making the most of this opportunity to help community managers with useful metrics is going to include the availability of *date summarized comparison metrics*. These types of metrics have two "filters" or "parameters" fed into them that are more abstractly defined in the Growth, Maturity and Decline metrics on the CHAOSS project.

1. Given a pool of repositories of interest for a community manager, rank them in ascending or descending order by a metric.
2. Over a specified time period or
3. Over a specified periodicity (i.e., month) for a length of time (i.e., year).

For example, one open source program office we talked with is interested in the following set of *date summarized comparison metrics*. Given a pool of repositories of interest to the program office (dozens to hundreds of repositories):

1. What ten repositories have the most commits this year (straight commits, and lines of code)?
2. How many new projects were launched this year?
3. What are the top ten new repositories in terms of commits this year (straight commits, and lines of code)?
4. How many commits and lines of code were contributed by outside contributors this calendar year? Organizationally sponsored contributors?
5. What organizations are the top five external contributors of commits, comments and merges?
6. What are the total number of repository watchers we have across all of our projects?
7. Which repositories have the most stars? Of the ones new this year? Of all the projects? Which projects have the most new stars this year?

## 3 Open Ended Community Manager Questions to Support with Metrics

There are other, more open ended questions that may be useful to open source community managers:

1. Is a repository active?
   (a) Visual differentiation that examines issue and commit data
   (b) Activity in the past 30 days

   (c) Across all repositories, present the 50th percentile as a baseline and show repositories above and below that line.

2. Should we archive this repository?

   (a) Enable an input from the manager after reviewing statistics.

   (b) Activity level, inactivity level and dependencies

   (c) Mean/Median/Mode histogram for commits/repo

3. Should we feature this repository in our top 10? (Probably a subjective decision based on some kind of *composite scoring system* that is likely specific to the needs of every community manager or program office.)

4. Who are our top authors? (Some kind of aggregated contribution ranking by time period [year, month, week, day?]. *nominally, I have a concern about these kinds of metrics being "gameable", but if they are not visible to contributors themselves, there is less "gaming" opportunity.*)

5. What are our top repositories? (Probably a subjective decision based on some kind of *composite scoring system* that is likely specific to the needs of every community manager or program office.)

6. Most active repositories by time period [Week? Month? Year?]. Activity to be revealed through a mix of Retention and Maintainer activity primarily focusing on the latter. Number of issues and commits. Also the frequency of pull requests and the number of closed issues.

7. Least active repositories by time period [Week? Month? Year?]. *Bottom of scores calculated, as above.*

8. Who is our most active contributor (Some kind of aggregated contribution ranking by time period [year, month, week, day?]. *nominally, I have a concern about these kinds of metrics being "gameable", but if they are not visible to contributors themselves, there is less "gaming" opportunity.*)

9. What new contributors submitted their first new patches/issues this week? (*Visualization Note:* New contributors can be colored in visualizations and then additionally a graph can be made for number of)

10. Which contributors became inactive? (Will need a mechanism for setting "inactive" thresholds.)

11. Baseline level for the "average" repository in an organization and for each, individual organization repository.

12. What projects outside of a community manager's general view (GitHub organization or other boundary) doe my repositories depend on or do my contributors also significantly contribute to?

13. Build a summary report in 140 characters or less. For example, "Your total commits in this time period [week? month?] across the organization increased 12% over the last period. Your most active repositories remained the same. You have 8 new contributors, which is 1 below your mean for the past year. For more information, click here."

14. Once a metrics baseline is established, what can be done to move them? [1]

15. Are there optimal measures for some metrics?

    (a) Pull request size?

    (b) Ratio of maintainers to contributors?

    (c) New contributor to consistent contributor ratio?

    (d) New contributor to maintainer ratio?

# 4 Augur Specific Design Change Recommendations

Next is a list of Augur specific design changes suggested thus far, based on conversations with community managers.

1. Showing all of the projects in a GitHub organization in a dashboard by default is generally useful.

2. Make the lines more clear in the charts, especially when there are multiple lines in comparison

3. How to zoom in and out is not intuitive. In the case of Google Finance, for example, a default, subset period was displayed when they used the "below the line mirrored line" interface this is modeled after. That old model makes it fairly clear that the ability to adjust the range of dates is what that box below the line in google finance *is for*. Alternately, Google's more updated way of representing time, providing users choices, and showing comparisons may be even more useful and engaging. In general, its important that the time zooming is more clear.

4. For the projects a community manager chooses to follow, go ahead and give them comparison checkboxes at the top of the page. I think from a design point of view, we should limit comparisons as discussed, to 7 or 8, simply due to the limits in human visual perception.

5. The ability to adjust the viewing windows to a month summary level is desired.

[1] Once we are to this point, I think CHAOSS is kicking butt and taking names.

*Figure 1: In one view, Google lets you see a 1 year window of a stock's performance.*

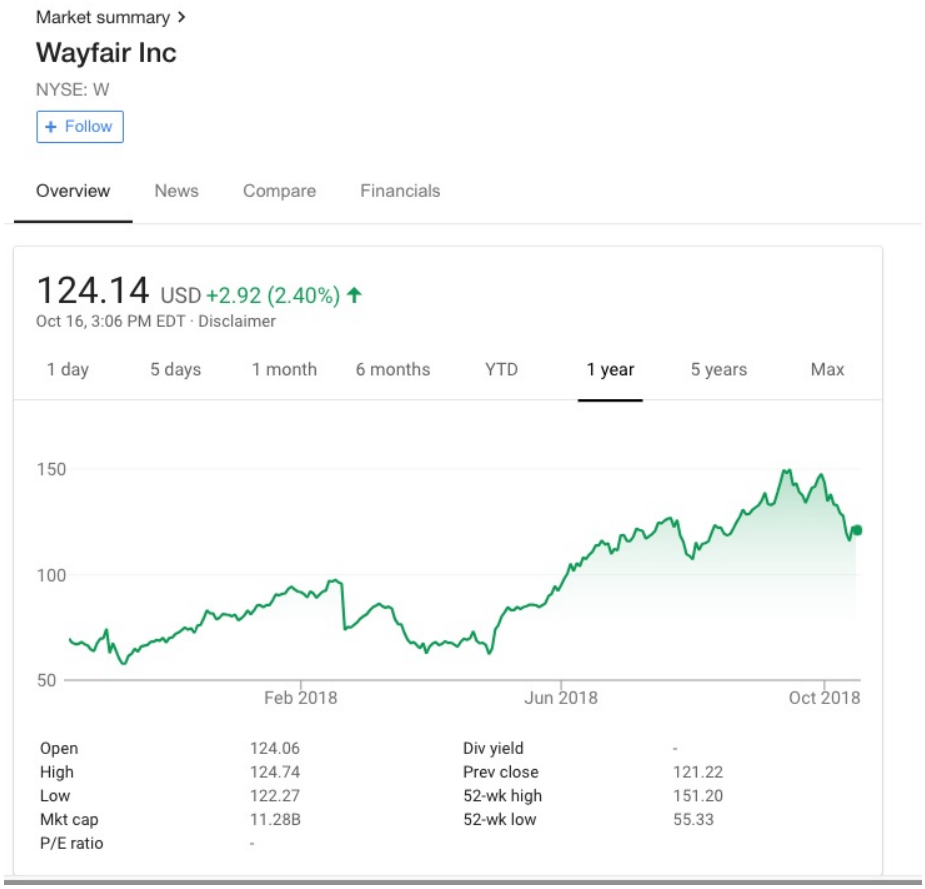


*Figure 2: In another view, you can choose a 3 month period. Comparing the two time periods also draws out the trend with red or green colors, depending on whether or not the index, in this case a stock's price, has increased or decreased overall during the selected time period.*

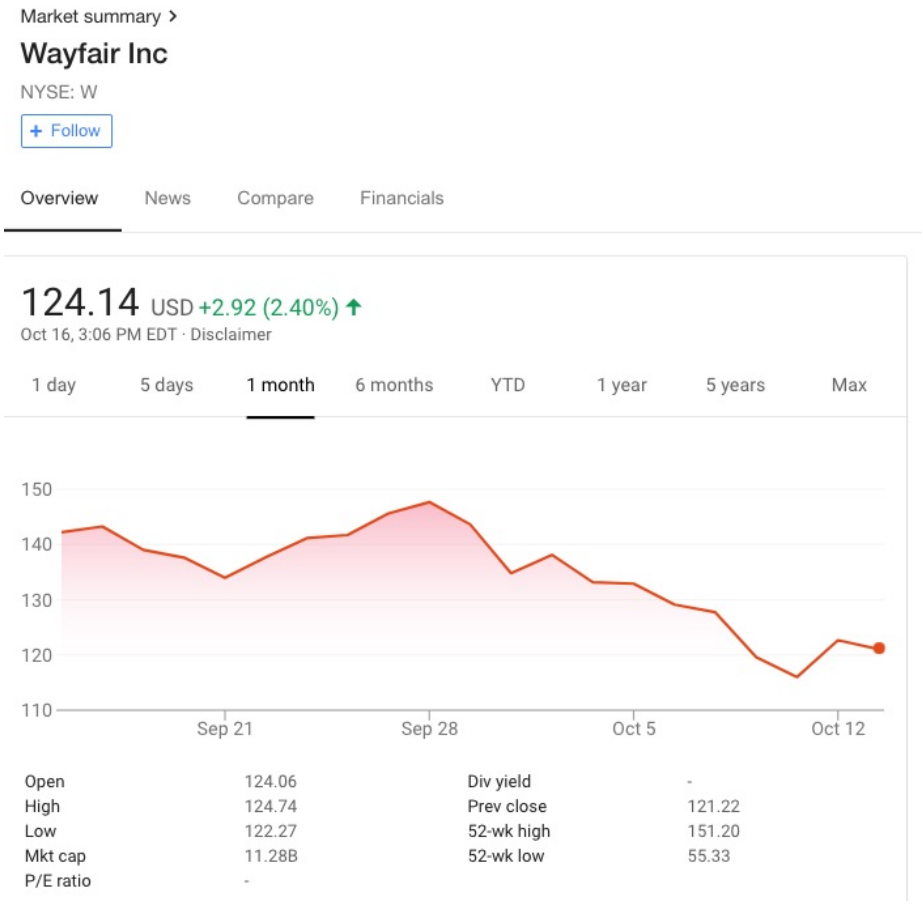
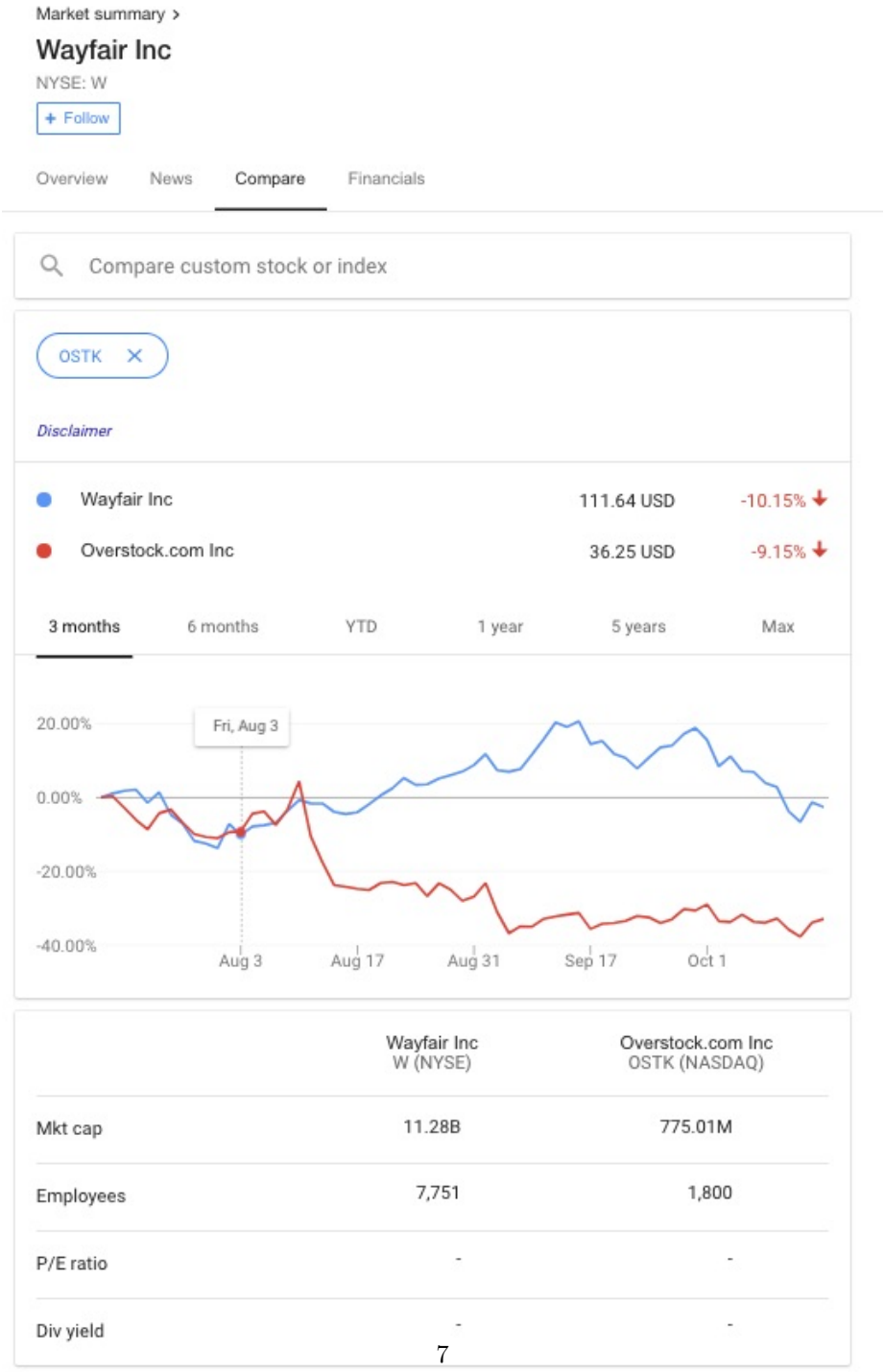


*Figure 3: Comparisons are similarly interesting in Google's finance interface. You can simply add a number of stocks in much the same way our users want to add a number of different repositories.*



6. Right now Augur does not make it clear that metrics are, by default, aggregated by week.

7. New contributor response time. When a new contributor joins a project, what is the response time for their contribution?

8. A graph **comparing** commits and commit comments on x and y axes **between projects** is desired. Same with Issue and Issue comments.

9. In general, the last two years of data gets the most use. We should focus our default display on this range.

# 5 Data Source Trust Issues

1. Greater transparency of metrics data origins will be helpful for understanding discrepancies between current understanding and what metrics show.

   (a) We should include some detailed notes from Brian Warner about how Facade is counting lines of code, and possibly some instrumentation to enable those counts to be altered by user provided parameters.

   (b) Outside contributor organization Data. One community manager reported that their lines of code by organization data seems to look wrong. I did explain that these are mapped from a list of companies and emails we put together, and getting this right is something community managers will need some kind of mapping tool to do. GitDM is a tool that people sometimes use to create these maps, and Augur does follow a derivative of that work. Its probably the case that maintaining these affiliation lists is something that needs to be made easier for community managers, especially in cases where the number of organizations contributing to a project is diverse (there is a substantial range among community managers we spoke with. Some are managing complex ecosystems involving mostly outside contributors. Most are in the middle. And some of contributor lists highly skewed toward their own organization.)

2. GHTorrent data, while excellent for prototyping, faces some limitations under the scrutiny of community managers. For example, when using the cloned repositories, and then going back to *issues*, the issues data in GHTorrent does not "look right". I think the graph API might offer some possibilities for us to store issue statistics we pull directly from GitHub and update periodically as an alternative to GHTorrent.

3. When issues are moved from an older system, like Gerrit, into GitHub issues, in general the statistics for the converted issues are dodgy, even through the GitHub API. We are likely to encounter this, and at some point may want to include Gerrit data in a common data structure with issues from GitHub and other sources.

# 6 New Metrics Suggested

1. Add metric "number of clones"
2. "Unique visitors" to a repository is a data point available from the GitHub API which is interesting.
3. Include a metric that is a comparison of the ratio of new committers and total committers in a time period. Or, perhaps simply those two metrics in alignment. Seeing the number of new committers in a set of repositories can be a useful indication of momentum in one direction or another; though I hasten to add that this is not canonically the case.
4. Some kind of representation of the ratio between commits and lines of code per commit
5. Test coverage within a repository is something to consider measuring for safety critical systems software.
6. Identifying the relationship between the DCO and the CLA.
7. There is a tension between risk and value that, as our metrics develop in those areas, we are well advised to keep in mind.
8. The work that Matt Snell and Matt Germonprez at the University of Nebraska-Omaha are starting related to risk metrics is of great interest. Getting these metrics into Augur is something we should plan for as soon as reasonably possible.

# 7 Design Possibilities

## 7.1 Augur

For Augur, I think the interface changes that enable comparisons and adjust the level of self apparent ways to compress or expand the time, as per the Google examples, are at the top of the list of things that will make Augur more useful for Kate and other community managers. Feedback on these notes will be helpful. I think the new committers to committers ratio is important, as well as enabling comparisons across projects in the bubble graphs as well. Transparency of data sources and limitations of data sources for both the API and the front end, which are above average but not complete, are important.

## 7.2 Growth Maturity and Decline Working Group

Many of the metrics of interest to community managers fall under the "growth maturity and decline" working group. From a design perspective it appears that, possibly, the way that metrics are expressed and consumed by these stakeholders in their individual derivatives of the *community manager use case* is quite far

removed from the detailed definition work occurring around specific metrics. Discussion around an example implementation like Augur is helping draw out some of this more "zoomed out" feedback. The design of system interfaces frequently includes the need to navigate between granular details and the overall user experience Zemel et al. (2007); Barab et al. (2007). This is less of a focus in the development of software engineering metrics, though recent research is beginning to illustrate the criticality of visual design for interpreting analytic information González-Torres et al. (2016).

# 8 Acknowledgements

Many members of the CHAOSS community contributed to this report and analysis. I am happy to share names with permission from the contributors, but I have not requested permission as of the publication date.

# References

Barab, S, T Dodge, MK Thomas, C Jackson, and H Tuzun. 2007. Our designs and the social agendas they carry. *Journal of the Learning Sciences* 16 (2): 263–305.

González-Torres, Antonio, Francisco J. García-Peñalvo, Roberto Therón-Sánchez, and Ricardo Colomo-Palacios. 2016. Knowledge discovery in software teams by means of evolutionary visual software analytics. *Science of Computer Programming* 121: 55–74. doi:10.1016/j.scico.2015.09.005. https://linkinghub.elsevier.com/retrieve/pii/S0167642315002658.

Zemel, Alan, Timothy Koschmann, Curtis LeBaron, and Paul Feltovich. 2007. What are we Missing? Usability's Indexical Ground. *Computer Supported Cooperative Work*.