DOI:10.1145/2856457

George V. Neville-Neil

Kode Vicious
Code Hoarding

Committing to commits, and the beauty of summarizing graphs.

Dear KV,

Why are so many useful features of open source projects hidden under obscure configuration options that mean they will get little or no use? Is this just typically poor documentation and promotion, or is there something that makes these developers hide their code? It is not as if the code seems broken. When I turned these features on in some recent code I came across, the system remained stable under test and in production. I feel code should either be used or removed from the system. If the code is in a source-code repository, then it is not really lost, but it is also not cluttering the rest of the system.

Use It or Lose It

Dear Use It,

There are as many reasons for exposing or hiding code as there are coders in the firmament of programming. Put another way, there is more code hidden in source repos than are dreamt of in your ... well, you get the idea.

One of the most common forms of code hiding I have encountered when working with any code, not just open source, is the code that is committed but to which the developers are not fully committed themselves. Sometimes this is code that supports a feature demanded by sales or marketing, but which the developers either do not believe in or which they consider to be actively harmful to the system.

"I'd like a stop button."

"A stop button, if used, will destroy all of our data."

"That's OK, I want a stop button, and I want it to be *red*."

These types of features are often buried in the system and can be turned on only after clicking through dialog boxes with increasingly dire warnings.

A more aggravating form of buried code comes from developers who are unwilling to commit to a feature, usually because of a lack of understanding of the code. It is not uncommon in long-running projects for developers to come and go and for newer developers to fear existing code they have not taken the time to understand or measure. A host of excuses will be deployed against enabling the code—because it is too slow, too old, or has too many bugs. The problem is often compounded by the often-stubborn nature of most developers, who, until they are hit with overwhelming evidence, refuse to believe anything that does not match their worldview. I cannot legally recommend actually hitting developers, but I will say that I find my hard-covered notebooks do tend to make a very nice thudding sound on conference-room tables.

If you are working on a project and come up against this type of intransigence, there are better and worse ways of dealing with it. Arguing piecemeal, or tit-for-tat, with a developer is time consuming and, like teaching a pig to dance, only aggravates the pig. One way to handle the topic is political, a word I realize is anathema to much of this audience. Make your case to other developers for building a consensus on the right way of handling a feature. This is, surprisingly, a worthwhile exercise that has longer-term benefits to a project. It turns out it is sometimes easier to convince someone of something when other people they trust are all saying the same thing.

Making measurements and gathering evidence in support of turning on a feature are also valid methods of dealing with developer intransigence. As a side benefit, any test code and results you generate can be used again later, as test code, when the code inevitably changes. If you put the evidence on nice, heavy paper, it can, as pointed out earlier, be used more directly to make your case.

KV

Dear KV,

I have a co-worker who has spent the past several months making large-scale changes to our software in his private branch. While some of these changes are simple bug fixes that get placed back into the main branch of development, a lot of it is performance-related work that he emails about every week or two. These email messages contain nearly endless pages of numbers that

are explained only briefly or not at all. I want to ask if he has ever heard of graphs but fear it might just make him think I want to review more of his long email messages. Is there a better way to explain to him that we need a summary rather than all the data?

Bars and Lines

Dear Bars,

It continues to amaze me that most people never had to write lab reports in high school science classes. I don't know about you, but that is where I learned to summarize data and to create graphs of the data from experiments carried out during the class. My favorite statement about the use of graphing comes from Robert Watson, who teaches a graduate class on operating systems research at the University of Cambridge: "Graphs serve a number of functions in lab reports—not least, visually presenting a large amount of data, allowing trends in that data to be identified easily (e.g., inflection points), but also to allow a visual comparison between different sets of results. As such, if we set up an experimental comparison between two sets of data (e.g., static vs. dynamic performance), putting them on the same graph will allow them to be easily compared."

That is an instruction to graduate students who are required to summarize their data in lab reports for his course. If you remove the words "lab reports" in the first sentence, you could simply try beginning there in explaining to your co-worker why no one is reading his email messages full of data.

Once you have convinced him that graphs are good, you will probably have to help make good graphs. Unlike when we made graphs in class, with paper, pencils, and straightedges, there is now a plethora of software available that will take raw data and graph it on your behalf. The problem is you still need to know which question you are trying to answer and whether the summary is helping or hindering your understanding of the underlying problem.

Graphing a set of numbers is not just a matter of dumping them into a program and seeing a pretty picture; it has to do with taking a set of data and making an easy-to-understand representation that can be shared among a group of people in order to make a judgment about a system. To make any sort of judgment, you need to know what you are trying to measure.

All the measurements you are trying to graph need to share a common trait; otherwise, you risk comparing apples and oranges. For example, a repeated measure of network bandwidth, over time, is a measurement of the number of bits per unit of time (often seconds) seen during some sampling period, such as "every five minutes." As long as all the measurements to be compared share those attributes, all is well, but if the sampling period between two sets were different—for example, if one is hourly and another is every five minutes—that would be a false comparison.

When graphing data for comparison it is important to ensure your axes line up. Generating five graphs with software that automatically resizes the graph to encompass all the data gives results that cannot be compared. First find the maximal x- and y-ranges of all your data, and use these maxima to dictate the length and markings of all of the graphs so they can be compared visually. If one set of data has significant outliers, it can make the entire set of graphs useless. As an example, a time-series graph of network latencies that is mostly measured in microseconds will be rendered unusable by a single measurement in milliseconds, as all that you will see is a slightly thickened line following the y-, or time, axis, and a single spike somewhere in the middle of the graph. Graphs that do not handle outliers correctly are completely useless.

Most graphing software does a poor job of labeling data, and yet the labeling is an important and intrinsic part of the story. What does that thin blue line really indicate? It is going up and to the right. Is that good? Or is that the number of unclosed bug reports? Picking reasonable colors and labels for your data may sound like too much work, but much like commenting your code, it will pay off in the end. If you cannot remember which dots are from which dataset and therefore what the graph even means, then the graph is useless.

Finally, a quick note on summarizing data. An important distinction that is often overlooked or misunderstood is that between the mean and the median. The mean is the simple average of a set of numbers and is the most commonly used—and misused—type of data summary. The problem with using the mean to summarize a set of data is that the data must be normal for the mean to have any meaning. When I say "normal," I do not mean to imply it had a happy childhood. Normal data is evenly distributed along a bell curve, which does not describe many sets of data. If data is heavily skewed, or has significant outliers, then the median is preferred. The median is more difficult to calculate by hand, but I hear these computer things that are all the rage can calculate it for you.

While there are large open source and commercial packages that will graph your data, I prefer to start simply, and, therefore, I commend to you ministat (https://www.freebsd.org/cgi/man.cgi?query=ministat), a program written by Poul-Henning Kamp that is included in the FreeBSD operating system. It generates simple text graphs based on columns of data and will do all the work of telling you about means, medians, and statistical significance. If, at some point, your data has to be communicated to those who are from another world, as you indicated in your question, you can then plot it with R (https://www.r-project.org), but that's a story for another time.

KV

Related articles on queue.acm.org

Commitment Issues
Kode Vicious
http://queue.acm.org/detail.cfm?id=1721964

Unlocking Concurrency
Ali-Reza Adl-Tabatabai, Christos Kozyrakis, and Bratin Saha
http://queue.acm.org/detail.cfm?id=1189288

Software Needs Seatbelts and Airbags
Emery D. Berger
http://queue.acm.org/detail.cfm?id=2333133

George V. Neville-Neil (kv@acm.org) is the proprietor of Neville-Neil Consulting and co-chair of the *ACM Queue* editorial board. He works on networking and operating systems code for fun and profit, teaches courses on various programming-related subjects, and encourages your comments, quips, and code snips pertaining to his *Communications* column.