DOI:10.1145/1536616.1536638

Technical Perspective
Maintaining Quality in the Face of Distributed Development

By James Herbsleb

IT WAS A problem that should not have taken three weeks to solve. But the tester was in Germany and the developer was in England. The documentation claimed that if a function was called from a command line with particular parameters, it would return values of particular state variables. If the operator simply entered blank, it would return the values of all the variables. It was this last option that was causing the grief. Entering blank just returned garbage, insisted the tester. The developer couldn't duplicate the problem, and after three weeks of frustrating emails and phone conversations, the developer hopped on an airplane to Germany. A few seconds after sitting down beside the tester, he observed the tester enter the characters "b-l-a-n-k" and hit return, rather than just hitting return by itself. Mystery solved.

This is just one anecdote, but it is emblematic of how most everyone these days thinks of globally distributed development. Whatever benefits might be realized, one is also likely to encounter a steady diet of frustration, delay, misunderstandings, mistakes, and cross-purposes. And there is no shortage of research supporting these intuitions. Study after study has provided rich descriptions of the variety of problems encountered, and quantified their cumulative effects. Delay due to multi-site projects has received particular attention, as the numerous small holdups, quite salient to developers, accumulate into a significant burden.

But enough of this doom and gloom, say Bird et al. In a study of Vista, a very large, widely distributed project, the authors take a close look at the impact of geography on software quality, and, to everyone's surprise—including the authors—they find little or none. Binaries developed within a single building, or across boundaries including different buildings, campuses, or even continents, have virtually the same rates of failures, after controlling for other factors. Here, finally, is some encouraging news for those going global.

The study is important not just for the overall result, but also for the care that was taken in achieving that result. The authors take full advantage of the rare opportunity provided by their impressive data set. They have data from a company directory that allows them to consider many levels of geographic distribution, rather than the coarser binary distributed-versus-collocated distinction typical of previous research. Moreover, their sample size is sufficient to give credibility to their negative results. As a rule, because of the way that statistical tests are used in an experimental context, a negative result (when the predicted differences are not observed) is difficult to interpret. Maybe the effect does not exist, or maybe it does exist but the study was not sensitive enough to observe it. But with a sufficiently large sample and a carefully conducted study, however, one can have confidence that if a substantial effect existed it is highly likely that it would have been detected. This is such a study, and the negative results are convincing.

In a study of Vista, a very large, widely distributed project, the authors take a close look at the impact of geography on software quality, and, to everyone's surprise—including the authors—they find little or none.

The authors also go to great pains to rule out other possible explanations that could cloud the results. For example, maybe only relatively simple binaries are developed in distributed fashion. If that were the case, then perhaps distributed development efforts achieved a dead heat in quality with collocated development only because the distributed teams had an easier task. The authors did a careful analysis of the differences between their distributed and collocated binaries, and found virtually no differences. It appears the comparison was meaningful—apples to apples, so to speak. One small exception to this was a weak tendency for distributed work to involve more people, an intriguing parallel to an earlier finding my collaborators and I encountered when analyzing multisite delay, as the authors pointed out. This hint that somehow more people get pulled into the work when the work spans sites seems worthy of further investigation.

Finally, the authors carefully revisit the literature, and point out that several of the conditions shown in the past to disrupt distributed projects were not present in the project they studied. The sites used a consistent tool set, for example, shared common schedules, and had ample opportunity to overcome cultural differences. This rich description of the context of the project will be very helpful for future researchers who may find different results. It will help us eventually to sort through the potential causes of quality problems as case studies accumulate.

The following paper is an important contribution, a terrific read, and an elegant example of bringing scientific methods to bear on a problem of both theoretical and practical concern. C

James Herbsleb (jdh@cs.cmu.edu) is a professor of computer science at Carnegie Mellon University, Pittsburgh, PA.