Microsoft again has forced us to think about the past and future of Windows. It announced in March 2006 that the update to XP, now called Windows Vista and more than five years in the making, would again be delayed, this time to early 2007 for the consumer version. This announcement comes after a series of revelations in 2005 in the Wall Street Journal and other publications about how the latest Windows project, then called Longhorn, had so many problems that the project collapsed into gridlock.\(^1\) We now know that the chaotic “spaghetti” architecture of Windows and its 50 million or so lines of code was one of the major reasons for this gridlock. Making even small changes in one part of the product led to unpredictable and destabilizing consequences in other parts since most of the components were tied together in complex and unpredictable ways. Even 4,000 or so software developers and an equivalent number of software testers was not enough to get Longhorn working.

Although this problem was years in the making, last year the Windows executives finally took some action. They resolved to make the code more modular (and thus easier to build, test, and debug) as well as to revamp their engineering and testing processes to reject buggy code automatically, at least to some extent. The Microsoft executives then decided to throw out years of coding work on Longhorn and go back to the Windows 2000/XP server code base, which had many fewer defects in it, and then directed the team to build more modular features around this. They had already removed the most important new set of features for search and file management, and would later ship these separately. Finally, in August 2005, Microsoft released a beta version of Vista and scheduled the final release for December 2006—allowing itself a little less than a year and a half for final testing.

Once more, however, we learn the Windows team needs more time to get the product out the door. And once again, we hear computer manufacturers, electronics retailers, and software applications producers around...

the world are frustrated because delays in shipping new versions of Windows have major ripple effects in these other multibillion-dollar markets. We also have yet another reason to question Microsoft’s credibility: Can we believe any promised ship date for software products of the complexity of Windows?

To its credit, Microsoft has said it is not putting the short-term business issues first; it is primarily concerned with making sure that Windows Vista is of the highest quality possible. My concern, though, is about what is humanly possible and what is impossible. Microsoft has made important changes in how it builds Windows, but these are incremental moves. Are they enough to solve what appears to be a monumental problem? Or is Microsoft fated to box itself repeatedly into a corner with a tar pit for a floor, to borrow the analogy used by Frederick Brooks in his classic book on the difficulties of software engineering, *The Mythical Man-Month*, published in 1975. Brooks argued, based on his experiences at IBM, that the more a software company struggled with a large software project and tried to add people, the later the project became because as the team size grew, the complexity of internal communications grew exponentially.

This process already seems to have occurred in the Windows group. It does seem that the more people Microsoft adds to the team, the more complex and late the projects have become. We must ask whether or not it is even possible to guarantee the ship date as well as the quality—in terms of functionality, stability, or security—of a software system that exceeds 50 million lines of code and requires so many thousands of engineers. Windows Vista is of a size and complexity perhaps never seen before in desktop software. It has become Microsoft’s equivalent of the Apollo moon project, or perhaps a quagmire more resembling what the U.S. military faced in Vietnam.

The unfortunate part to this story is that Microsoft need not have taken this path. The contrast in my mind is to what software development used to be like at Microsoft. In 1995, Rick Selby and I, in our book *Microsoft Secrets*, analyzed the best-run groups at the time (mainly Excel, Word, and Windows NT). We wrote about the set of engineering practices (principally, use of “milestones” to break one big project into shorter, smaller, more focused projects, and “daily builds” with regression tests to create always-working versions of the product and keep everyone’s changes and fixes synchronized). These techniques allowed Microsoft to create many new features and ship increasingly complex products like Windows NT and Office with some level of control and predictability. The Windows NT group then consisted of about 400 people, approximately half of whom were developers and most of the rest testers. The code base was then about four million lines—a big but manageable size. Windows NT, which shipped in the mid-1990s, was the biggest group and the biggest product Microsoft had ever managed. But the company did an admirable job on its first enterprise-class operating system and gave Unix vendors a run for their money. This product established new standards for quality from Microsoft, albeit with lots of help from engineers and managers hired from more established software producers such as DEC and AT&T.

So there was a time when Microsoft’s development groups were relatively nimble and lean. They used to be cocky, and rightfully so, given the market success of their products and the success of the company, which trounced competitors such as IBM, Lotus, WordPerfect, Novell, and others. For example, in researching *Microsoft Secrets*, we heard and read about how Microsoft engineers, while working with IBM on OS/2, mocked IBM’s development style. IBM used a thousand or so engineers on its part of OS/2, mocked IBM’s development style. IBM used a thousand or so engineers on its part of OS/2, while Microsoft used only a few hundred to build an equivalent amount of code that ran faster, better, and cost less to produce. I wonder today what happened to this culture of being lean and mean, and of prize small teams of talented programmers rather than hordes of engineers and managers with,
apparently, widely varying ability. Perhaps the old Microsoft culture died as Windows grew to such enormous size that only a Napoleonic-sized army could deal with it. But brute force has a limit, and apparently the Windows team has been running up against that limit.

But money can go a long way. I still have every confidence that Microsoft will ship the mass-market version of Windows Vista sometime in 2007 and that it will be the best operating system Microsoft has produced to date in terms of functionality, stability, and security. It had better be, after what will total in excess of six years of development and probably more than $2 billion of investment in programming and testing.

But even if brute force is enough to get one more version of Windows out to consumers, what does the road ahead look like? Not so promising. With every new version, Windows seems to be almost doubling in the size and complexity of the code as well as in the size of the project team. Windows 95, for example, was 15 million lines of code; Windows XP was 35 million lines of code [1]. Moreover, the amount of time needed to test and stabilize the product seems to be nearly doubling with each new product release. Will the next desktop version of Windows have 100 million lines of code and require a team of 16,000 people to build and test? Will Microsoft need five years to write and rewrite the code and then another three years to debug and test after the team reaches that magical but often illusive goal of “code complete”? In the meantime, as we wait for the successor to Vista, we can only guess what Microsoft’s rivals—Google, Apple, and the Linux community—will have achieved.

This is not to say the size of the Windows team is the only problem. A neatly modularized and layered architecture, better automated testing tools, and disciplined engineering practices like design and code reviews can enable a large team to work like many small teams. But the combinatorial testing problem still arises quickly enough as the number of modules and subsystems grow. In addition, Microsoft is committed to maintaining compatibility with thousands of applications produced by other vendors over many years, and this adds to all of Windows’s difficulties. Furthermore, Microsoft is under increasing pressure from Google, Apple, and Linux to keep adding (not cutting) features to improve functions as well as security. So it is difficult for Microsoft to go back in time to the days of smaller and leaner software.

Of course, there may be a brighter future for Microsoft and Windows than I have depicted here. Perhaps Chairman Bill Gates, CEO Steve Ballmer, Senior VP Steve Sinofsky (who will head the Windows team in the future), or some other executive will have an epiphany and decide to restructure the Windows team and the Windows product much more aggressively. I think they need to revert to a smaller team that can communicate better and move more quickly. They also must revert to a smaller core for what is now Windows and then make that core as rock-solid and secure as Linux or Unix, and build new features as tight, modular applications that run along with Windows, not as an integral part of it. This would mean a further departure from Gates’s strategy of “integrated innovation” (the real reason why, I think, Windows Longhorn grew to 50 million lines of spaghetti code).

True, it would make Microsoft more vulnerable to antitrust accusations that it is illegally bundling products with Windows (like Internet Explorer, Media Player, or new search and file management systems) that it should be selling separately. But, realistically, a more radical strategy for reshaping Windows the product and Windows the team may be the only road ahead that keeps Microsoft out of the tar pit.

Reference

Michael Cusumano (cusumano@mit.edu) is a professor at the MIT Sloan School of Management and author of The Business of Software, Free Press, 2004.