CHAPTER TWELVE

# Software Gold Rushes

*Prosperity doth best discover vice, but adversity doth best discover virtue.*

FRANCIS BACON

*The root of all superstition is that men observe when a thing hits, but not when it misses.*

FRANCIS BACON

In January 1848, James Marshall discovered gold in California's American River near a mill he was building for John Sutter. At first Marshall and Sutter dismissed the pea-sized nuggets as a nuisance; they believed the attention that gold would bring would spoil Sutter's plans to build an agricultural empire. But within months word spread, and by 1849 thousands of men and a handful of women from around the world had contracted Gold Rush Fever. They headed to California to make their fortunes in what became known as the California Gold Rush. The rush west created a new economy driven by high-risk entrepreneurialism and fueled by dreams of striking it rich. Precious few 49ers actually realized that dream during the gold rush days, but the dream lives on in many modern software companies and individual software developers.

The California Gold Rush was unique in that the gold was found in riverbeds instead of embedded in hard rock. That meant that, at first, anyone with a tin pan and an entrepreneurial spirit had a chance to make a fortune. But by mid-1849, most of the easy gold had been found, which meant that a typical miner spent ten hours a day in ice cold water, digging,

sifting, and washing. As time passed, this backbreaking work yielded less and less gold. There were occasional lucky strikes well into the 1850s, which provided just enough good news to encourage thousands to keep digging. Most failed every day, but they kept on for years.

After the early days of the gold rush, miners had to use more advanced techniques to extract gold. By the early 1850s, a single miner could no longer work a claim by himself. He needed the help of other people and technology. At first, miners banded together informally to build dams, reroute rivers, and extract the gold. But soon more capital-intensive techniques were needed, and the informal groups of miners were replaced by corporations. By the mid-1850s, most of the miners who remained were corporate employees rather than individual entrepreneurs.

## *Software Gold Rushes*

The advent of a major new technology often means the beginning of what I think of as a "software gold rush." Companies and individual entrepreneurs rush into new technology areas, hoping that a little bit of hard work will produce a product that will make them wealthy. I've personally seen software gold rushes with the advent of the IBM PC and Microsoft DOS operating system, the migration from DOS to Windows, and the growth of Internet computing. More new-technology gold rushes will undoubtedly follow.

Gold rush software development is characterized by high-risk, high-reward development practices. Few companies have established competitive presences in the marketplace during the early days of a new technology, and many of the new-technology gold nuggets—successful new products—seem to be lying on the ground, waiting for anyone with the right mix of innovation and initiative to pick them up. Software 49ers rush into the new technology, hoping to stake their claim before anyone else does. The typical gold rushers are two guys working in a garage, legendary dynamic duos such as Bill Gates and Paul Allen of Microsoft, Steve Jobs and Steve Wozniak of Apple Computer, and Bob Frankston and Dan Bricklin of VisiCalc.

The practices employed by software developers with gold rush fever are usually associated with hacking rather than engineering: informal processes, long hours, little documentation, bare-bones quality assurance practices—in other words, hero-based code-and-fix development. These practices require little training and low overhead, and they expose projects to a high risk of failure.

The odds of striking it rich during a software gold rush are about as good as they were during the California Gold Rush—for every success story, there are hundreds of projects that go bust. But these small failures aren't nearly as interesting as the huge successes, and so we don't hear very much about them. Two guys who work hard and don't strike it rich aren't a very good news story, unless by chance there's something interesting about their garage.

As with the California Gold Rush, software projects run with hero-based development in gold rush periods are successful from time to time. The gold rush projects are so enormously lucrative when they do succeed that they convince software developers that high-risk practices can work, and thus the rare but widely publicized successes spread Gold Rush Fever and help to keep hero-based practices alive.

## *Post-Gold Rush Development*

Post-gold rush software development is characterized by more methodical, lower-risk, more capital-intensive, more labor-intensive development practices. Projects use larger teams, rely on more formal processes, adhere to more standards (compatibility with legacy code, industry-wide protocols, and so on), and work with much larger code bases. The emphasis is less on rushing software to market quickly and more on reliability, interoperability, and usability. Projects focus on software engineering considerations that hardly matter during a gold rush but that matter a lot after a technology matures.

Gold rush-style development practices have even lower odds of working in a post-gold rush phase than they did during the risky gold rush phase. In the early days of a new technology, there are few established players

or products. The technological barriers to entry are low, and early products can be small, unpolished, and unreliable and still succeed. As with the California Gold Rush, fewer people and less capital are needed to stake a claim during the early days of a new technology. The first version of Word for the Macintosh was a gold rush product that consisted of just 153,000 lines of code. Two guys in a garage have a chance to compete against the major corporations when a successful product can be built with 153,000 lines of code. As the technology matures, however, the easy gold runs out, and successful companies have to compete on the basis of more capital-intensive projects.

One of the most damaging mistakes that successful gold rush companies make is to persist in using gold rush development approaches as the technology matures and their projects scale up. To compete successfully in the post-gold rush phase, the successful project needs to do a lot more than simply multiply the number of guys and get a bigger garage.

Post-gold rush customers are more demanding. Gold rush customers are what Everett M. Rogers, author of *Diffusion of Innovations,* calls "Innovators" and "Early Adopters."[1] They tend to be technologically savvy, drawn to new technologies, and forgiving of the rough edges that go along with them. Gold rush products can be much less polished than the products that come later and still be successful. Post-gold rush customers are what Rogers calls "Early Majority," "Late Majority," and "Laggards." They are risk averse and want polished products that work reliably. This demand sets a higher bar for post-gold rush products. The current version of Word for Windows, a post-gold rush product, consists of more than 5,000,000 lines of code.

One surprising implication of gold rush dynamics is that the companies that are successful during one gold rush are likely to fail during the next gold rush. The archetypal post-gold rushers are the companies that became established during an earlier gold rush. These companies repeat Marshall and Sutter's mistake of seeing new-technology gold as a nuisance that will interfere with their well-laid plans for extracting maximum value from the claims they staked during the last gold rush. Examples of companies that were slow to pick up new-technology gold nuggets include IBM during the early days of PC-DOS; Lotus during the early days of Windows;

and Microsoft during the early dawn of the Internet, although Microsoft recovered from its early mistakes. The most compelling example in modern computing has to be Xerox. Many of the fundamental ideas of modern desktop computing were invented at Xerox's Palo Alto Research Center, including the GUI interface, the mouse, and the Ethernet. But Xerox was so busy trying not to lose the copy machine war that it lost the computing war without ever really entering it.

Other post-gold rush companies are just too bloated to compete effectively in gold rush markets. The overhead necessary to sell to Early Majority and Late Majority customers isn't needed during a gold rush. In a gold rush market, you can cut your products to the bone and still do well with the innovators and early adopters who count the most in those markets.

We'll undoubtedly see this boom and bust pattern repeated during whatever technology cycle follows the Internet. Some of the companies that had the greatest successes in the early days of the Internet—such as Amazon.com, eBay, and Yahoo—will miss the next wave; only time will tell which will successfully navigate the next great transition.

## *The Sense and Nonsense of Gold Rush Economics*

From a macroeconomic viewpoint, thousands of individual software developers taking on entrepreneurial risk voluntarily—with a few lucky entrepreneurs striking it rich and the rest chalking up their losses to experience—is tremendously beneficial. No one but the individual entrepreneur pays for the failures, and everyone has a chance to benefit from buying and using the few products that succeed. But how can an individual company harness this dynamic? What company could possibly afford to fund thousands of individual entrepreneurs during a gold rush phase just to find the one or two that successfully develop new gold rush technology? Even companies with extensive research facilities like AT&T, IBM, Microsoft, and Xerox can't afford to fund thousands of projects in each new technology area, which is one reason that software company acquisitions during a gold rush phase are more sensible than they at first appear.

Some industry observers thought Microsoft was crazy to pay $130 million to acquire Vermeer Technology, original creators of FrontPage, when Vermeer had only about $10 million in annual revenue. But from the entrepreneurial gold rush point of view, paying $130 million for the one success in a thousand is a cheap alternative to funding a thousand internal dead-end entrepreneurial experiments.

## *Scaling Up and Scaling Down*

Post-gold rush software engineering practices have unequivocally proved their worth on large projects. (Doubters can turn to Chapters 13 and 14.) They also have a lot to offer smaller projects. Larry Constantine describes an Australian Computer Society Software Challenge in which three-person teams had to develop and deliver a 200 function-point application in six hours.[2] This is a significant challenge, equivalent to writing about 20,000 lines of code in a traditional third-generation language or about 5,000 lines of code in a visual programming language.

A team from Ernst and Young decided to follow a formal development methodology—a scaled-down version of their regular methodology—complete with staged activities and intermediate deliverables. Their approach included careful requirements analysis and design. Many of their competitors dived straight into coding, and for the first couple of hours the team from Ernst and Young lagged behind.

But by mid-day they had developed a commanding lead. At the end of the day the team from Ernst and Young lost, but not because their systematic approach had failed. They lost because they accidentally overwrote some of their working files, squandering their afternoon work and delivering less functionality at the end of the day than they had demonstrated at lunchtime.

Would the team from Ernst and Young have won without the configuration-management snafu? The answer is yes. They reappeared a few months later at another rapid-development face-off—this time with version control and backup—and they won.[3] Their success was achieved not

by stripping down their earlier approach but by identifying weaknesses in their old process and making improvements.

This general value of applying systematic process improvement within small organizations has been confirmed by a Software Engineering Institute study.[4] The success rate of process improvement programs in organizations with fewer than 50 software developers has been just as good as it has been in larger organizations. Moreover, small organizations have fewer of the problems that inhibited success in larger organizations, such as organizational politics and turf guarding.

## *Back to the Gold Rush*

Gold rush software projects are inherently risky, but the use of haphazard software development practices has unnecessarily added even more risk. Developers working on gold rush projects have been saddled for decades with the methodological equivalents of tin pans and shovels. The result is that many of their insights, ideas, and innovations are needlessly lost, just as the Ernst and Young team's work was needlessly lost due to lack of source code control.

Systematic approaches to software engineering are necessary for post-gold rush projects to succeed; they are equally useful for projects still in the gold rush stage. What would have happened if the original designers of VisiCalc, Lotus 1-2-3, MacOS, the Mosaic Web browser, and other groundbreaking products had overwritten their working files? How many innovative products have we never heard of because their developers *did* overwrite their working files? And how many have succumbed to more subtle errors?

During a gold rush, you can be terribly sloppy and not very skilled and still make a fortune, but the odds are against you. After a gold rush, you have to be more disciplined and more skilled just to break even. The entrepreneurial buzz a person gets from participating in a gold rush project is one of life's great thrills, but there's no conflict between entrepreneurial energy and the use of effective software development practices. By

examining the practices that work well in post-gold rush environments, you can gain insights into practices that work best when you have Gold Rush Fever, increasing your chances of ultimately hitting pay dirt.

## *Notes*

1. Rogers, Everett M., *Diffusion of Innovations,* 4th Ed., New York: The Free Press, 1995.
2. Constantine, Larry, "Under Pressure," *Software Development,* October 1995, pp. 111–112.
3. Constantine, Larry, "Re: Architecture," *Software Development,* January 1996, pp. 87–88.
4. Herbsleb, James, et al., "Software Quality and the Capability Maturity Model," *Communications of the ACM,* June 1997, pp. 30–40.