Rapid Application Development - Case Study 1

Mickey was ready to lead his second project at Square-Tech, a giant in the PC spreadsheet world. His previous project had been tough. The original schedule had called for his team to deliver Square-Calc version 2.0 in 12 months, but it had taken 18. The team had known at the outset that the date was aggressive, so for almost the entire 18 months they were on a death march, working 12-hour days and 6- or 7-day weeks. At the end of the project, two of the six team members quit, and Bob, the strongest developer on the team, set out from Seattle on his bicycle for parts unknown. Bob said he wasn't quitting, and he sent Mickey a postcard from Ottumwa, South Dakota—a picture of himself riding a giant jackalope—but no one knew when he would be back.

Square-Calc 3.0 needed to be released 12 months after version 2.0, so after two months of project cleanup, post mortems, and vacations, Mickey was ready to try again. He had 10 months to deliver version 3.0. Mickey met with Kim, his manager, to discuss the project plan. Kim was known for being able to eke out every possible bit of work from the developers she managed. John, from user documentation, and Helen, from QA, were also present.

"Version 3.0 needs to leapfrog the competition," Kim said. "So we need a strong effort on this project. I know that your team didn't feel that the company was fully behind them last time, so this time the company is ready to provide all the support it can. I've approved individual private offices, new state-of-the-art computers, and free soda pop for the whole project. How does that sound?"

"That sounds great," Mickey said. "All these developers are experienced, so I mainly want to provide lots of motivation and support and then get out of the way. I don't want to micro-manage them. I'd like to have each developer sign up for a part of the system. We had a lot of interface problems last time, so I also want to spend some time designing the interfaces between the parts, and then turn them loose."

"If this is a 10-month project, we're going to need visually frozen software by the 8-month mark to get the user documentation ready on time," John said. "Last time the developers kept making changes right up until the end. The README file was 20 pages long, which was embarrassing. Our user manuals are getting killed in the reviews. As long as you agree to a visual freeze, your development approach sounds fine."

"We need visual freeze by about the same time to write our automated test scripts," Helen added. Mickey agreed to the visual freeze. Kim approved Mickey's overall approach and told him to keep her posted.

As the project began, the developers were happy about their private offices, new computers, and soda pop, so they got off to a strong start. It wasn't long before they were voluntarily working well into the evening.

Months went by, and they made steady progress. They produced an early prototype, and continued to produce a steady stream of code. Management kept the pressure on. John reminded Mickey several times of his commitment to a visual freeze at the 8-month mark, which Mickey found irritating, but everything seemed to be progressing nicely.

Bob returned from his bike trip during the project's fourth month, refreshed, and jumped into the project with some new thoughts he'd had while riding. Mickey worried about whether Bob could implement as much functionality as he wanted to in the time allowed, but Bob was committed to his ideas and guaranteed on-time delivery no matter how much work it took.

The team members worked independently on their parts, and as visual freeze approached, they began to integrate their code. They started at 2:00 in the afternoon the day before the visual freeze deadline and soon discovered that the program wouldn't compile, much less run. The combined code had several dozen syntax errors, and it seemed like each one they fixed generated 10 more. At midnight, they decided to call it a night.

The next morning, Kim met with the team. "Is the program ready to hand over to documentation and testing?"

"Not yet," Mickey said. "We're having some integration problems. We might be ready by this afternoon." The team worked that afternoon and evening, but couldn't fix all of the bugs they were discovering. At the end of the day they conceded that they had no idea how much longer integration would take.

It took two full weeks to fix all the syntax errors and get the system to run at all. When the team turned over the frozen build two weeks late, testing and documentation rejected it immediately. "This is too unstable to document," John said. "It crashes every few minutes, and there are lots of paths we can't even exercise."

Helen agreed. "There's no point in having testers write defect reports when the system is so unstable that it crashes practically every time you make a menu selection."

Mickey agreed with them and said he'd focus his team's efforts on bug fixes. Kim reminded them of the 10-month deadline and said that this product couldn't be late like the last one.

It took a month to make the system reliable enough to begin documenting and testing it. By then they were only two weeks from the 10-month mark, and they worked even harder.

But testing began finding defects faster than the developers could correct them. Fixes to one part of the system frequently caused problems in other parts. There was no chance of making the 10-month ship date. Kim called an emergency meeting. "I can see that you're all working hard," she said, "but that's not good enough. I need results. I've given you every kind of support I know how, and I don't have any software to show for it. If you don't finish this product soon, the company could go under."

As the pressure mounted, morale faded fast. More months went by, the product began to stabilize, and Kim kept the pressure on. Some of the interfaces turned out to be extremely inefficient, and that called for several more weeks of performance work.

Bob, despite working virtually around the clock, delivered his software later than the rest of the team. His code was virtually bug-free, but he had changed some of the user-interface components, and testing and user documentation threw fits.

Mickey met with John and Helen. "You won't like it, but our options are as follows: We can keep Bob's code the way it is and rev the test scripts and user documentation, or we can throw out Bob's code and write it all again. Bob won't rewrite his code, and no one else on the team will either. Looks like you'll have to change the user documentation and test scripts." After putting up token resistance, John and Helen begrudgingly agreed.

In the end, it took the developers 15 months to complete the software. Because of the visual changes, the user documentation missed its slot in the printer's schedule, so after the developers cut the master disks there was a two-week shipping delay while Square-Tech waited for documents to come back from the printer. After release, user response to Square-Calc version 3.0 was lukewarm, and within months it slipped from second place in market share to fourth. Mickey realized that he had delivered his second project 50 percent over schedule, just like the first.