

# Traversing Software's "Last Mile": XP Meets Corporate Reality

by **Dave Rooney**

XP is a development process that focuses on delivering business value to the customer at a quick but sustainable pace. It is characterized by short iterations of two to three weeks in duration, followed by acceptance testing by the customer. This process provides very fast feedback to the programmers, allowing them to correct problems earlier or avoid them altogether.

The practices of XP focus mainly on the development aspects of a system, and they are very good at delivering high-quality systems that actually meet their customers' needs. However, XP doesn't pay much, if any, attention to the issues involved in deploying an application to the desktop of the end user.

In the broadband Internet access arena, the term "the last mile" refers to the technical, economical, and regulatory issues involved in providing high-speed access to the homes of subscribers. There already exists a glut of network capacity at the metropolitan, regional, and global levels, but a bottleneck exists at the neighborhood level — the last mile. This same issue exists in the software development world, where agile processes such as Extreme Programming allow us to develop software quickly. Our "last mile" is the deployment process.

What follows is a story of Extreme Programming success for the development and testing of an application, but a deployment process that was anything but agile.

## THE PROJECT

In August 2001, I joined the development team for a client-server application at a large Canadian federal government department. The application had been in production since 1998, and a release was being rolled out to production as I arrived. Everyone was scurrying about in the usual panic to fix last-minute problems in order to get the release out the door. The system did get shipped, owing to the heroics of a few ... another day in the IT trenches.

For this project, the development and customer teams were colocated, and the customer was already in charge of creating and managing the requirements. This showed promise for being an agile environment! A few weeks after I arrived, the development team leader moved on to another project and was replaced, purely by chance, by an old friend of mine.

I didn't find out until later that the team leader who left was a bit of a control freak who insisted that he do all the analysis and simply

generated specifications for the developers. To his credit, the design of the system was pretty good. However, there was a terrible communications gap between the developers and the customer, despite the fact that they were colocated. The customer never saw the system until it went into user testing after a six- to eight-month development cycle, and the developers weren't privy to many issues brought up at weekly status meetings, which only the team leader attended.

## GOING EXTREME

I decided to pitch the idea of using XP to the new team leader, knowing that if she thought it wouldn't work, she would say so. She agreed to try it out, and together we sold it to our project manager. He immediately bought into the notion of high levels of communication in XP and was all for giving it a go. "It couldn't be any worse" was one of the comments.

We sat down with the customer and explained the XP approach. They were quite enthusiastic, since they had been clamoring for ages for more communication with the development team during the development process. We set about getting the requirements (stories) together for the next

release and providing high-level estimates for the time required. In this particular case, we had an absolute date by which certain stories had to be in production. We worked back from that date, accounting for deployment time and so on, and split the stories into iterations.

I'll never forget the meeting in which we set the completion date for the first iteration. I told the customer that the date was set in stone, and we would adjust the scope if necessary. They looked skeptical to say the least. However, they played along, and we then went about breaking the stories into tasks and getting to work on them.

### **SMASHING DEVELOPMENT SUCCESS**

It's worth noting that there were three developers on this project, so we obviously couldn't use pair programming at all times. In addition, we didn't use the standard unit testing frameworks because it would have meant changing a lot of existing code, which was deemed to be too much of a risk. We did, however, implement a test utility that allowed us to quickly test the code that was most affected by the changes for this new release. Testing that took 10-15 minutes in previous versions of the system now took about 5 seconds!

While we were developing, we called the customer in numerous times to have a look at what we were doing. After all, they were only a few feet away! In previous

versions, the customer wasn't even allowed to get screen shots, let alone play with the system. The previous team leader's rationale for this was that the screen shots might change before this system was released.

We met the target date for the first iteration with no difficulty and gave the customer a working version of the system for their testing. A few minor problems surfaced during their acceptance testing, which we fixed right away without even having to write stories.

We started the second iteration and were actually ahead of schedule (due mainly to the faster testing turnaround time) when the customer called from a meeting with an outside contractor. We were told that a significant chunk of what we were working on was going to have to change. I had just checked in the code from the area most affected about half an hour earlier! The customer faxed some of the changes from Toronto and then met with us a day or two later when they returned. We quickly adjusted the stories and tasks to accommodate the changes and restarted development. We completed the iteration on time, despite the changes, and again the customer performed their acceptance testing.

Iteration 3 also was completed ahead of schedule, so we were able to add several additional small stories that had been bumped from the original release plan. The customer tested; we fixed any problems.

The customer then performed a final, across-the-board set of acceptance and regression tests for which they had scheduled four weeks. They were done in two and a half weeks, including Windows 2000 testing that wasn't originally in the test plan.

We packaged the system and shipped it on the day it had been promised six months earlier. In terms of the number of additions/changes implemented, the release was by far the largest since the original 1.0 release in 1998. It was the only release ever to be shipped on the proposed release date.

Throughout the process, the stress level was low, and team spirits were quite high. We actually had fun, just as the proponents of XP promised!

### **REALITY BITES**

Not mentioned in the success story above is that, in this government department, any application to be deployed must undergo a certification process to ensure that it meets certain corporate standards for release documentation, cohabitates with other applications, and works as advertised when installed. All applications must go through this process, regardless of size or scope. This process is completely outside the control of any development team.

For a typical release of this system, the certification process takes four to five weeks from the time we complete development and final acceptance testing to when the system can be deployed to the

desktop. The group that handles the certification process also bills my customer between Cdn \$30,000-Cdn \$40,000 per release.

**We have enough work to be putting out releases every three months for at least another year, but the budget simply doesn't exist to handle the deployment costs.**

Obviously, this creates a tremendous problem. My customer's annual budget for development is such that, due to the high deployment cost, we are only able to provide the users with a single release this fiscal year. Not surprisingly, the end users are quite frustrated by our inability to quickly respond to change because of the time required to deploy the application. I have stated in meetings that we have enough work to be putting out releases every three months for at least another year, but the budget simply doesn't exist to handle the deployment costs.

Bear in mind that no one is disputing the necessity and importance of integration and release package testing. However, the customer (and their customer, the end user) is frustrated by the inability to quickly respond to change because of the time required to deploy the application.

Our project is not unique in this regard. The certification process has frustrated development teams

throughout the department for years. Despite attempts to make the process more flexible, the certification and deployment costs of any application are prohibitive.

A recent survey by the organization to which the certification group belongs elicited responses to a "general comments" question such as:

- The cost associated with obtaining [the organization's] services has not been addressed, and it should be. [The organization] is pricing themselves right out of the market. [The organization's] costs are breaking our budgets.
- [The] database group is not working as a partner with [the development organization], and they dictate solutions and standards.
- Timely communications and improved timelines on completing tasks [are needed].
- [The organization] has moved away from being a customer service organization to being a process-driven organization that cares very little about the client.

These issues aren't confined to applications developed using Extreme Programming; they exist for all other systems in this government department. Indeed, when anyone mentions the organization responsible, a roll of the eyes tells all. Having said that, this organization is the only game in town; you must go through the certification group if you are to deploy to the desktop.

I recently posted a question to the Extreme Programming mailing list<sup>1</sup> that mentioned these problems with the deployment process. I was hoping that others out there had bridged the last mile, so to speak, and could enlighten me. The responses I received, however, were from others who could only sympathize and relate their own deployment issues.

## ROOT CAUSES

### Culture Clash

In my search for "a better way," I talked to several people in my organization about their perspectives on why these deployment issues exist. First, this government department was formed when two smaller departments and several agencies were merged together in 1993 as part of a cost reduction and consolidation plan. This led to a certain level of culture shock.

At that time, I worked for one of the departments, and we were actively using agile principles such as colocation with the customer and iterative development. As part of the support for the application on which I was working, I had access to all of the regional sites across Canada and could diagnose problems and deploy a patch, if required, within 24 hours. I was not unique in this regard; many people had similar access in order to provide the best support possible.

<sup>1</sup><http://groups.yahoo.com/group/extremeprogramming>.

The other department, which was somewhat larger, used (and continues to use) many legacy mainframe systems. When the time came for me to deploy a new full release of my application, I had to go through the official deployment process. I was provided with a list of the steps that the certification group took to ensure that the application would deploy properly and would cohabitate with other standard desktop applications. This list was the same one used to certify the deployment package for the department's financial system, which is mainframe-based. That was my first clue to what was to come!

I discovered very quickly that the certification group had little previous experience with PC and LAN applications. At first they insisted that all of the certification steps be followed to the letter, but I was able to have the list contracted a bit because of steps that applied only to the mainframe world. It took some time, but the application was deployed, and the end users were happy.

Shortly after that release, there was a request for a new report in the system. Building the report was no problem, but I was informed that in order to deploy the updated application, we would have to go through the entire certification process again and would be billed accordingly. We managed to get the update shipped quickly by calling it an emergency release, despite the fact that no emergency existed. After that, I reverted back to using my direct access to all of the regional sites to deploy minor

updates. For anything more than that, we bit the bullet and used the certification and deployment process.

**We managed to get the update shipped quickly by calling it an emergency release, despite the fact that no emergency existed.**

Again, we weren't the only group experiencing these problems, and we certainly weren't the only group using surreptitious means to subvert the deployment process. The process has become less mainframe oriented since the mid-1990s, but it hasn't been reduced in time or cost.

#### **Mainframe Focus**

The second issue was hinted at above — the mainframe focus of the certification and deployment organization. This comes from the director of that organization, who apparently sees any application that doesn't exist on a mainframe as a toy. I'm sure that's a gross overgeneralization, but that mentality does exist.

My experience dealing with the people who actually perform the testing has been quite positive. They, too, want to get the application shipped and on the desktop. However, their policies, procedures, and priorities are dictated from above, as are the rates they charge groups for their services.

#### **Scope of Testing**

The third issue is the scope of the testing performed by the certification group. After some cajoling, they gave us an outline of their test strategy:

- Ensure completeness of the release "package" (application, database scripts, installation instructions, operating instructions)
- Review the instructions; ensure that they are the same in English and French
- Test the package installation using the instructions
- Attempt to uninstall the application
- Run the application and verify database connectivity

So far, I'm with them 100%, but here is where I diverge:

- Test the application's response to issues such as power loss or network disconnection while running (Are there any side effects such as data corruption?)
- Perform high-level functional and application behavior test (i.e., navigation, queries, add/update/delete items)
- Conduct help testing and language consistency testing (All of our applications must support English and French.)
- Do print testing
- Do workstation performance testing
- Conduct error-handling testing (They never really explained what this was!)

- Perform cohabitation testing with office automation products and other corporate applications

The first five items make perfect sense to me, and they logically fall within the responsibilities of an organization responsible for deploying applications to the desktop. However, I (and many others) take issue with the last seven items. The belief among the development teams is that those items are outside the scope of certification testing and instead belong to the development and customer teams. We feel that this is an area of change that could bring significant reductions in the time and cost required to deploy an application.

**Making the certification and deployment process more agile becomes a key factor in reducing the time and costs associated with delivering software to the customer.**

### BABY STEPS

As the well-known proverb says, “A journey of a thousand miles begins with a single step.” This is true in our situation as well. While we were developing the last release of our application, the customer’s organization opened up a new Unified Acceptance Testing Lab. It consists of some 24 workstations of varying configurations

and power and covers all of the Windows versions in use in the department, both English and French. The lab is able to set up a workstation for the particular needs of an application (e.g., the application must have Oracle SQL\*Net, Microsoft Outlook, and Application “X” installed) within 24 to 48 hours. The lab also provides access to automated testing tools.

The customer used this environment during the final acceptance testing of the last release of our application, and it proved very valuable to have clean workstations on which to test the application and its install package. The goal of this testing environment is to completely supplant the need for the certification group to perform any application testing beyond installation and startup. The expectation is that the customer will provide their acceptance test plan and will sign off that it has been met. In our last release, we provided both the acceptance tests and a list of unit tests as deliverables. These were easily generated from the customer’s acceptance test documents and the development team’s tool for automating the unit tests.

Since this new testing lab is part of the customer’s organization, it is considered a capital cost, and the development teams are not charged for using it. In our case, we could conceivably reduce a four-to five-week certification testing process to two to three weeks, with a proportional reduction in costs.

### CONCLUSION

Extreme Programming and other agile development processes enable programmers to write better-quality software and react to change faster. XP also dictates frequent small releases in order to allow faster feedback from the end users and thus minimize the risk that the wrong software is developed. Certification of an application for deployment ensures that it meets corporate standards for documentation, will install and uninstall properly, and minimizes the risk that the application will adversely affect other systems. Traversing the last mile to an end user’s desktop requires a balance of these two sometimes conflicting aspects of the corporate IT world. Making the certification and deployment process more agile becomes a key factor in reducing the time and costs associated with delivering software to the customer.

*Dave Rooney is the principal of Mayford Technologies, Inc. and has been using Extreme Programming since the beginning of 2001 with great success. Mr. Rooney has been developing business applications for the past 15 years with Java/J2EE, C/C++, and PowerBuilder. He was initially drawn to XP by its focus on delivering business value to the customer. He quickly realized its potential for reducing or eliminating many of the obstructions in place in many software projects and appreciated XP’s emphasis on creating well-tested, well-written code.*

*Mr. Rooney can be reached at E-mail: [dave.rooney@mayford.ca](mailto:dave.rooney@mayford.ca); Web site: [www.mayford.ca](http://www.mayford.ca).*