Extreme Programming in a Customer Services Organization

Srinivasa Pulugurtha
Customer Service Engineer
IONA Technologies
200 West Street
Waltham, MA 02451, USA
+1 7819028128
spulugur@iona.com

Jean-Noel Neveu
Customer Service Manager
IONA Technologies
200 West Street
Waltham, MA 02451 USA
+1 7819028274
pedron@iona.com

Francis Lynch
Sr. Customer Service Engineer
IONA Technologies
200 West Street
Waltham, MA 02451 USA
+1 7819028189
flynch@iona.com

Abstract

IONA Technologies has embraced Extreme Programming (XP), company wide. For its part, Customer Services (CS) has played a considerable role as an "on-site customer" to Product Development (PD) [2]. In addition to this, CS has also been using the XP methodology to enhance its existing processes and to resolve certain problems within the department. Since XP is not tailored to solve CS issues, we have had to adapt its practices to suit our needs. This paper illustrates how CS is taking advantage of the XP methodology.

Keywords

Extreme Programming (XP), Customer Services (CS)

1 INTRODUCTION

One of the primary roles of CS is to assist Customers in resolving technical issues. This is mainly achieved through writing test cases for reported problems, logging potential bugs and finding suitable workarounds for as many of these bugs as possible. With more than 4500 customers developing and deploying CORBA & J2EE applications on multiple platforms and programming languages, this is not a trivial task for the 50 or so personnel in the department. The department is spread across 4 key locations worldwide in order to provide 24X7 support.

2 THE PROBLEMS WE WERE FACING

Traditionally CS has followed rigid processes that allowed for very little deviation. Furthermore, IONA Technologies has experienced rapid growth over the last several years. Between the inflexibility of these processes and our rapid growth, we have not been well positioned to provide the type of support we would like to offer. These processes consisted of "Service Level Agreements" between departments that defined conditions that had to be met for an issue to move between departments. When these conditions were misjudged, the issue would bounce back and forth between two departments resulting in a waste of time.

Because the processes in place would dictate the next step there was very little communication between the departments. For example, an engineer fixing a bug in PD would often reject the bug, with reasons such as: not a bug, vague specification, test case does not run or reflect the bug, instead of discussing the issue with the CS engineer who logged the bug. In this situation not only were we using our time ineffectively, we were also delaying patches to our customers.

Another problem that we faced was the ownership of issues within CS. Typically, each CS engineer would be assigned customer Service Requests (SR), our representation of a customer issue. In the past, one engineer would work on an SR from inception to closure. Depending on the team members' areas of expertise and the severity of open issues, the workload may need to be redistributed. This results in frequent reassignment of SR's which can be very time-consuming as the new engineer researches the history of the request. This was inefficient and not much appreciated by customers, who would wonder why they were being asked the same questions multiple times.

3 HOW WE ARE USING XP

Previously CS played a role as an "on-site customer" to PD [2]. In order to better integrate with PD, CS started to introduce some relevant XP practices to its engineering-related tasks (e.g., bug queue management and writing test cases). Following the successful introduction of those practices CS decided to try and apply them to its other, less engineering-related, functions such as SR allocations, analysis and resolution of customer issues, writing of technical articles for IONA's knowledge base. Since there are no earlier examples of a CS organization following XP, we have had to try several variations of these practices before we could find the most suitable approach.

Stand-up meeting and planning game

Requirements for CS change everyday through the arrival of new issues reported by customers. The priorities of existing issues change as well; for example, a critical issue may displace an existing standard issue in severity level. As a result, it is often required that an issue be moved from one engineer to another with only a quick hand over session between the two. To address this and other problems related to SR transfers in general, "stand-up meetings" were introduced.

At first, these would last the recommended 10 minutes and involve quick discussions on active issues. For every issue put forward, members of the team would offer ideas and suggestions and more than one engineer would usually end up working off-line to further analyze the problem at hand. Stand-up meetings also allowed everybody in the team to be aware of what everyone else was working on, thus facilitating the transfer of an issue from one engineer to another. We found that very often

it was necessary to prolong the meetings to 30 minutes or more in order to achieve the best results. Also the stand-up meetings were clearly not addressing certain problems that we were facing, like inbox management and SR reviewing.

After a trial period of about six weeks we decided to adapt our meetings to a more productive format. In this format the meetings would be daily, seated, and would last about 30 minutes or so. We would first discuss unassigned SRs in our inboxes; once a customer issue is addressed as a group it turns out that it is much easier for an engineer to take primary ownership of it. It is also decided at this point if pairing is required and assignment is done accordingly.

We would then bring to the table any existing issues that engineers might judge worth discussing with the team. These might include SRs they are currently working on that are causing difficulties, some interesting knowledge that they want to share, pieces of information they require from PD or any other customer-related problems that might be weighing on their minds while attempting to solve them.

Every engineer walks out of the daily meetings with an idea of the next step for each of his/her issues. This prevents an engineer from feeling isolated and gives him/her a sense of being part of the team.

As part of the interdepartmental communication effort we also obtain the participation of PD in these meetings. The primary advantage of this is that PD gets to be aware, early in the game, if an issue is escalated. In addition to giving their input to our discussions the PD representative(s) sometimes pair up with CS engineers to expedite resolution of a problem. CS also gains an increased knowledge of the internals of the products, which is extremely helpful when dealing with complex issues.

Collective ownership

The team owns the product inboxes. An engineer need not own an issue alone; any issue assigned to a particular engineer can at any moment be reassigned to, or pair "programmed" with, a different engineer. On top of that an engineer is encouraged to request the participation of his/her peers in a brain-storming session on any problematic issue that he/she is working on at any time during the day.

Pair Programming

Within CS the term "pair programming" takes a slightly different meaning in that we often pair up on non-programming tasks. These might include analysis of customer issues, conference calls with customers, and logging bugs. We found that working with another engineer helps us analyze a problem better: hidden clues are less likely to be missed, and engineers become more aware of when extra information is required.

An individual issue can be divided into several tasks. If the issue is complicated enough, a pair of engineers can work separately on those tasks. Who pairs with whom on an SR depends on the outcome of our daily meetings. Some issues are simple enough that no pairing is required. An engineer might pair with more than one person in one day while working on multiple issues.

Before an engineer logs a bug, another engineer must review the bug report and pair programming of any associated test case is strongly encouraged.

Regression and acceptance testing

We use Junit [3] & HTTPUnit [3] wherever we can. We are currently working on porting and tailoring CPPUnit [3] to our needs. We are also building a testing framework that is run every time a patch is released. This serves as an acceptance test suite for bug fixes and also as a regression test framework, augmenting what QA already uses.

On-site Customer

In the same way that we act as an on-site customer to PD [2], our Technical Account Managers (TAM) act as customer proxies to us. Part of their function is to report customer problems, maintain information about customer's projects and work with management in prioritizing issues. They play an important role in our planning game.

4 BENEFITS FROM USING XP

We have not yet done a quantitative analysis of the improvement achieved after embracing XP but the results are self-evident through the following observations.

Communication within CS and with other departments in IONA has improved considerably. This can be observed from the fewer number of emails to internal mailing lists asking technical questions. It was also observed that PD has not yet rejected any of the pair programmed bugs, which was a frequent occurrence in the past.

There is now next to no duplication of effort by two CS engineers and transition of issues between people is much smoother.

5 CONCLUSION

We have not fully implemented all XP practices in our daily CS functions yet, but the ones that we have so far have helped us realize its potential. We intend to keep on experimenting with XP methodology in order to adapt it as best as we can to our CS Organization.

REFERENCES

1. K. Beck, Extreme Programming Explained: Embrace Change, Addison-Wesley, Reading, Mass., 1999.
2. Charles Poole and Jan Willem Huisman, Using Extreme Programming in a Maintenance Environment, IEEE Software Nov/Dec 2001.
3. Junit, CPPUnit, HTTPUnit and other XUnit extensions can be found at http://www.junit.org/news/extension/index.ht