The Intermediate Customer Anti-Pattern

Tom Perry
Authorize.net
tperry@authorize.net

Abstract

Scrum focuses on collaboration with the customer, but what if your customer is actually a provider for yet another customer? Then who is your real customer? What if these two possible customers have a hostile relationship? Then who is your customer? These are the kinds of real world questions that we examine in this experience report.

The report describes a situation involving a complex customer relationship and the consequences of failing to identify the correct customer. We share Lessons along with indicators to look for when dealing with your own challenging customers.

1. Introduction

This paper describes one team's struggle to understand how to work with multiple customers with competing goals and a lack of centralized ownership for the product. This paper describes a pattern of behavior that is relatively common in the software world: companies that support IT departments serving the needs of corporate clients. We will begin by describing the engagement in detail. Then we will outline the problem as it evolved. We will also address the solution applied as well as provide a discussion of possible alternative solutions and their pros and cons. Finally, we will outline the consequences of the proposed solutions.

2. Background

We began our engagement in early 2006. The company was in the services business, providing offsite Agile development services. Our teams had been using Agile methods for more than 2 years, so they were experienced with Agile development. We had been using a mix of both Scrum and XP practices on our teams. The level of developer experience on the team was very high. The team started as a two-person team exploring product requirements: one Scrum Master and one technical lead/architect. The team subsequently grew to a 9-11 person team as the development process ramped up.

The IT Customer

The customer was new to Agile development. They were a large IT department for a telecom company that was responsible for maintaining and growing IT assets for client business divisions throughout the company.

The IT department did not have any previous experience with Agile development. The culture was very much waterfall driven and structured. Large silos had developed between departments within the IT business. The management team was growing ever more frustrated with their own inability to deliver software within the expected timeframes. So this group was cautiously interested in Agile development.

Recognizing that they did not know much about Agile development, we were careful to begin the engagement with formal training in Agile development for this customer. We wanted them to understand the terminology and the process so that there was a minimum of friction created when we started actually developing software. We adopted the practice of giving our non-Agile customers training prior to starting an engagement as a best practice for subsequent engagements.

The training went well and we proceeded with some quick analysis to understand the scope of the engagement. In this case, the IT customer had taken over responsibility for maintaining and enhancing a product created by another division ("The end user client business"). This division had initially created this software to support the work they were doing and they were very reluctant to hand off their system to a remote IT group. They were accustomed to being able to get modifications made quickly using their own in-house resources and those of a local team they had years of experience working with. The IT

customer was dissatisfied with the local team and had decided to give the work to our team instead.

The End User Client Business

During the six months prior to our start on the project, the problems with the local team were only exacerbated. The IT group that had acquired the responsibility for maintaining and enhancing the software had no relevant domain knowledge. Therefore, a hostile dynamic emerged where the customers in the end user client business were making increasing demands on the IT department and getting very frustrated.

The end user client business had a large legacy web based system with an architecture that had grown increasingly brittle and unsustainable over the years. Simple modifications required weeks of work not only to identify where changes needed to be made, but to test the inevitable side effects that arose. By transitioning their product development to us, they were facing a situation where they had an underperforming team with domain knowledge being replaced by an Agile team with no domain knowledge. This lack of domain knowledge was going to be a huge challenge for our team.

Engagement Goals and Objectives

We were engaged to provide frequent releases of maintenance features from a large backlog that had accumulated over time. The promise of rapid development and close collaboration with the customer sounded like the perfect solution for our IT client. We were replacing an existing waterfall team that had been unable to release a new version of the product for more than 6 months.

The good news was that we were able to deliver a substantial new milestone in 2 months for one of their minor subsystems. The IT department was very happy with our performance, but they decided to postpone the release because it was lower priority than other work that they needed done. While we had learned a great deal about the system, we had delivered work that was low priority for the customer. After 2 months, we received the production system code. The bad news was that the system was in far worse shape than we had initially assumed. Continuing to apply patches to the existing system was not going to be sustainable – even for an Agile team.

3. The Problem

The more we learned about the system, the more we realized that the lack of domain knowledge at the IT department was really jeopardizing their ability to understand the significance of requested features and bugs. We felt that we understood where to find the best information. Therefore, we went to the end user client business to interview the end users.

We videotaped the users working with the system and interviewed them extensively. By the time we were finished, we felt that we had a genuine idea of the issues that needed to be resolved first.

Unfortunately, as a team, we were left with the impression that we knew the business needs and priorities of the customer business better than our clients within the IT business did. In essence, the team had lost its trust in the IT business. This was going to cause problems for us later.

Politics

Predictably, when we returned to the IT department, there was resentment that we were "going around" them. This aggravated the communication problems between our IT client and the customer business. As they continued to squabble over which bugs and features to address, the situation only got worse.

From the perspective of the end user customer business, the team was not delivering what they asked for. From the IT department's perspective, the team needed to be sheltered from the politics of the dialog. This led to some very confusing communication between the three parties.

In essence, the end user customer business wanted to get control of their application development back from the IT business. They felt, perhaps correctly, that the IT business did not understand the domain well enough to provide the application support they needed. The IT business felt they had a better understanding of large application development and that the client businesses should not be doing that kind of work.

Since the IT business had ownership of the application, they chose to make technical issues such as re-architecting the application their top priority. On the other hand, the end user client business had no understanding of the technical issues, but felt a compelling need for new features and application maintenance. This tension formed the background for the conflict that came next.

Confrontation

This finally culminated in a meeting that brought the whole issue to a head. A domain expert from the end user client business came out to visit with the IT department and with our team. Unfortunately, key representatives from the IT department failed to show up to the meeting. This left the team in an awkward situation where they were stuck in a meeting with a hostile customer, and the politics of the situation were such that communication was confused at best.

As the situation looked like it was headed for a confrontation, the Scrum Master intervened and explained that the situation had been aggravated by the IT department's failure to show up. In defending his team, the Scrum Master rather vigorously "tossed the IT customer under the bus."

The Scrum Master was replaced immediately and the contract continued.

4. The Solution

What had the Scrum Master done wrong in this situation? First, he had mistaken the end user as his ultimate customer. However, his customer, the one paying the bills, was the IT department. By getting closely involved with the end users, the Scrum Master was putting himself and the team in the middle of a political battle between two warring departments. This led to a confrontation where the Scrum Master made himself responsible for some the political issues between the two groups – and he suffered the consequences.

Two groups were asking for help from the team, the IT department and the system users in the end user client business. This made identifying the key stakeholders very difficult. The answer to this dilemma is simple: follow the money. Whoever is paying the money for the engagement is your primary stakeholder. Focus on that individual and do not get involved in the other groups or politics involved.

5. The Consequences

As a team, we spent long hours in review trying to understand what had happened. The team had lost a Scrum Master and much of the trust of the customer. How could we have dealt with this situation differently?

Of course, the natural fear is that by focusing on the intermediary (the IT department) rather than on the end user that we do the people who have to work with the system a disservice. That may be the case. The IT department may direct the team to do work that is not the most beneficial for the end users – that is their prerogative. If they cause too much pain for their end users then they will suffer the consequences. Ultimately, the Product Owner must be the single wring-able neck. They are the ones who are responsible for making these decisions and they are the ones who pay the price if they get it wrong. The Scrum Master needs to recognize this dysfunction and try to inform the Product Owner, but beyond that, they should not get involved with the end users in the feature negotiations.

Postponement

How else could this situation have been addressed? The fact that the intermediate customer was not present for a meeting with the end user was a warning sign. The intermediate customer owns the relationship with the end user. It would have been perfectly reasonable to postpone the meeting due to the absence of the missing party. This would allow all the parties involved to regroup and try again without anyone losing face.

Keeping positive

Of course, the customer had travelled cross-country for the meeting and was leaving that afternoon. There was considerable pressure to continue with the meeting at the time. Perhaps it should be considered in purely Scrum terms. Is it appropriate for the Scrum Master to continue what is effectively a customer negotiation without the Product owner present? Confusing the role of the Product Owner and the Scrum Master is very risky. No matter how good a facilitator the Scrum Master is, they do not have the authority to negotiate with the customer.

A better strategy would have been to call up the IT customer and get them involved in the meeting as soon as possible. Otherwise, there just was not much that the team could effectively do.

6. Conclusions

So in our rush to satisfy the needs of our users don't lose sight of who the paying customers are. They are the ones that need to be satisfied, even if their desires seem to be contrary to the needs of their users. Perhaps this situation is most commonly found when providing services to outside customers. However, the basic lesson is to avoid confusing the role of the Scrum Master and the Product Owner. If you do, you do so at your own peril.

We have described how the engagement started, and how the problem situation evolved. We have

discussed the solution that was applied as well as the possible alternative solutions that might have been applied. After reviewing the tradeoffs involved, we have addressed the consequences of the solution that was applied. This was an important learning experience with one Agile team and we hope it can help others recognize similar situations in the future. Ultimately, we all want a positive relationship with our customers where we satisfy their needs.

Our engagement with these challenging customers was ultimately a successful one. The team was able to deploy major revisions to the application in a very short timeframe. The team learned, adapted, and overcame the challenges of understanding the roles played by the “Intermediate customer”