Extreme Success / "If You build It, They Will Come"

David Hussman
Edison Ed Inc.
4327 Garfield Ave South
Minneapolis, Minnesota 55409 USA
01-612-743-4923
david@edisoned.net

ABSTRACT
Having been involved in more than a few positive experiences with extreme programming, I started thinking about how the XP development community will handle extreme success. Though all organizations would enjoy having extreme success, what happens when fast moving and successful XP projects collide or converge? What can be done to help promote synergy between project teams that may not be aware of each other? How can we avoid moving from extreme programming into extreme chaos?

In an attempt to address some of the issues that may surface around extreme success, this paper will present two XP environments. The first of these involves a small team creating a framework so compelling that the team was unprepared for the response. The second situation is more general in nature, examining the problems confronting a company trying to coordinate several successful XP projects sharing some dependencies

Keywords
architecture, planning game, customer team, multiple XP projects, project dependencies, scaling

1 INTRODUCTION

Having embraced XP as a developer and a coach, I have seen the success XP brings to projects, as well as the smiles it brings to the developer's faces. But how does XP address dependencies between projects? How does the customer team react to the news that the developers are waiting on functionality that is being implemented by a different team, which may or may not be using XP? Does an organization using XP on multiple projects need some kind of "super customer team" to help synchronize development efforts? How realistic is it to believe that all projects at a larger company will embrace XP or some other agile process?

After witnessing problematic interactions between multiple XP projects more than once, I found myself starting discussions around this topic with other XP practitioners. When I found out that there seemed to be more questions than answers, and I was not alone in my observations, I decided that an attempt should be made to raise the topic to a larger audience for discussion.

The Environment

While working as a consultant for a company which has spent a great deal of effort exploring XP as a corporate development process, I found myself looking for information describing the synchronization of planning for one or more XP projects. Having done a bit of searching, as well as having asked one or more XP coaches, I discovered the information I wanted was not in great supply. I had found some information in "Planning and Running an XP Iteration" by Cara Taber and Martin Fowler. Although the article describes one way to scale XP for a large team, the situation at my engagement was different (yet I do not think it is or was unique).

Once the move to embrace XP had taken hold at the company, XP was finding success on more than one project. Though the planning game helps steer projects toward success, the persons responsible for overseeing the organization of more than one project, or team, are faced with bringing projects together, scaling existing projects due to success or demand, and dealing with possible overlap between multiple projects.

The company's struggles were further magnified when two or more successful XP projects were sharing dependencies. Although XP promotes communication and cooperation, when several teams have found success with differing designs, asking the teams to look toward that which is best for the company, raised interesting questions. It seemed possible that the very project dedication and ownership cultured by XP had now inadvertently become a problem. When this situation arose, it appeared that the rules of the planning game, as well as the traditional XP roles, needed to embrace change.

2 THE FIRST ENCOUNTER

To help describe the problem, I have selected two development efforts in which I was either a developer or a coach during this past year. The first of these involves using XP to build a framework used to build other applications.

In an effort to find a general solution to a large business problem, a small group of developers within the organization created a framework that provided the ability to quickly build efficient applications. At the same time, several other teams within the company, which I will refer to as the application teams, were addressing similar problems using a disconnected collection of solutions. When the framework was found to be a more effective way to deliver functionality, many of the application teams started to rely on the framework as an integral component in their designs.

Not expecting this response, the framework team suddenly found that they were not able to respond to the

increased demand for new features and other development support. Having heard that XP might help organize their efforts, the framework team decided to give it a try. Although XP helped the team to clarify requirements and define team velocity, the team was still confronted with an ever-growing development task list.

Communications Breakdown

Even though the framework team was successfully using XP, the application teams were not able to wait for features to be implemented. Now that the applications teams were dependent on the framework, they started implementing features, and extending the framework, rather than waiting for the framework team's response to feature requests.

It was around this time that the framework team started to focus on the core of the framework. Without a clearly defined customer, the framework team had promoted the tech lead to the customer role. Although XP planning was working well for the framework team, the application teams and their project managers had become more and more frustrated with the lack of response to their requests for new features.

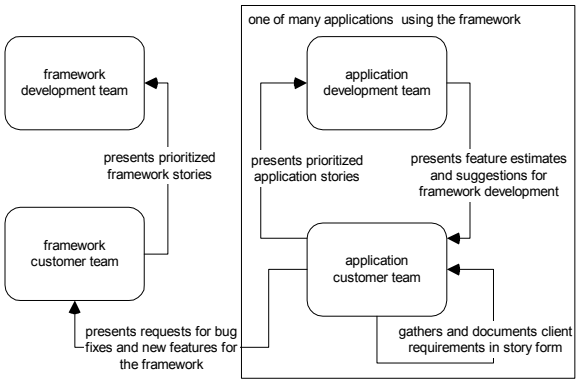
At the same time the framework team had become frustrated by the amount of time they were spending supporting the application teams. The framework team felt that they should be working on the deeper problems within the framework. Not unlike the classic problem that exists between those that create requirements and those that implement the requirements, the two teams had reached an impasse, and communications had broken down.

A Proposed Solution

In an effort to get the two teams working together again, a group of individuals from both the framework and the application teams met with a group of XP practitioners to see if there was a way in which XP could be used to help. As the group talked, it became clear to everyone that the real problem was not about technology, it was about process. More specifically, it was about scheduling and expectations. The success of the framework came with an increase of requests from the application teams as well as the support teams. Fear was growing that the company was building upon a framework which needed modification if it was to handle the increased demand.

The Customer's Customer

A group consensus was reached that all parties would benefit by creating a customer team for the framework developers. This customer team would interact with the customer teams for the applications. As is the case with every XP project, a good customer is essential. Due to the fact that the framework had to support a variety of applications, address a complex domain, and grow beyond its internal issues, the new customer team would need a diverse and knowledgeable composition of players. It was decided that the team should contain players who understood the needs of the applications, possessed domain expertise, and understood some of the problems that existed within the framework. It was during this meeting the following diagram was created as a communications tool.



The Modified Process

For all XP projects, a more informed and organized customer team will most often present a clearer road map to the developers. In this spirit, the group discussed how the new customer team would make its players available to the many persons that would request changes to the framework. The application customer teams would still be responsible for gathering and prioritizing requirements for their individual projects, but to rekindle communication and provide a simpler focus to the framework developers, only the framework customer team would own the responsibility of creating, organizing, and presenting stories to the framework developers.

A variation was added to the iteration planning for the application teams. While tasking and estimating a story, the application development team was empowered to suggest that a task or a story, be addressed within the framework. At this point, the application customer team may choose to remove the story from the table. This done, they could now choose to present the story to the framework customer team, or ask the application development team to proceed based on their estimates for the task or story. If the application customer team chooses to bring the task or story to the framework customer team, the framework customer team prioritizes the request.

The modified process was designed to solve two problems. The first of these was to allow the framework developers to stop focusing on scheduling and defending the features they were choosing to implement, and spend more time focusing on building the framework. Also, by sending all framework requests through the framework customer team, one group will have a better understanding of the features most desired by all application teams. This would hopefully provide a broader vision during feature selection and prioritization.

The second problem to be solved had to do with the scheduling issue previously mentioned. The application customer team was now responsible for making scheduling decisions for the applications based on information from the application development team and the framework customer team.

The Right Stuff?

In a perfect XP world, all parties would be using XP and the planning would be used across all applications. But as we know, there are no perfect worlds, and I wonder how many companies will choose to have a single planning game for all development efforts. The determination of whether the solution proposed here will work will be evaluated over the months to come. For now, at least the groups are working together and there is some vehicle for managing expectations as well as providing exposure to the framework development schedule.

3 A BROADER VIEW

To further the discussion, let's examine a more general version of the first encounter. What happens when there is a need to synchronize concurrent XP projects within an organization? When applications do not have any shared dependencies, or when there is one large XP project, this problem is unimportant. Once XP projects, which were initiated independently of each other, become dependant on each other, planning and design across the projects becomes problematic.

Imagine a company where several teams, somewhat disconnected from each other, are using XP and finding success. All projects are proceeding well, velocities are high, and quality is good. As each team's design materializes, the designs are distinctly different. This is fine until it is decided that the projects must be deployed together or merge to solve a business problem. At this point, with a significant portion of development completed, each team is vested in their design.

With significant development completed, and more than one solution working well for any of the individual projects, who decides which of the projects will be modified to work within another project's architecture? More importantly, what criteria will be used to make this decision?

The Company Customer

Now that we have moved toward emergent design, and away from big design up front, is it acceptable to create another form of the customer team to promote synergy among the XP projects? In the agile process community, where "the architecture group" is no longer in favor, is it possible to employ a solution similar to that used by the framework team in the previous scenario? Let's consider the idea of a "company customer."

What if the company customer were to field integration stories from the many applications groups? This would then be a similar solution to the one employed by the framework team. As the central focus of integration requests, the company customer would indeed have the best and broadest view from which to help with synchronization of cross cutting development efforts. For the many companies now facing enterprise application integration issues, this "company customer" might have an excellent vantage point.

But who would be in the company customer team? Even more important, how will the company prevent this new team from becoming yet another architecture group, destined to fail? Maybe the solution lies in learning from the past.

Many architecture groups have been driven by a technical mantra which fell on deaf ears, ears that were more interested in hearing solutions that make sense to the business. Driven by a focus to define a corporate technology stack or something similar, architecture teams of the past may have lost sight of that which matters most, building solutions meaningful to the company. The company customer would assess and prioritize development efforts based on business needs and not technologies.

Company Customer Composition

Similar to the customer team created in the first encounter, the company customer will need a diverse collection of members. To succeed, I think it would be best to rotate members from all customer teams within an organization through the company customer.

Of course a great deal of this discussion is based on an assumption that there are customer teams and not the single customer as proposed in early XP writings. It seems clear that we need to further our definition of the customer as a team and not an individual. As the XP community clarifies the composition of the customer team, it might become easier to understand which member(s) might be good candidates for the company customer. It might also become clearer if and when a company customer team is needed.

4 CONCLUSION

I am interested to see if the idea of a company customer raises any interest within the agile community. Two years ago, XP helped me to save a project from being shut down. It also helped a group of frustrated developers start laughing and enjoying their time at work. As more developers and managers find the XP difference to be a breath of fresh air, I think XP will need to continue to grow and embrace the change it has initiated.

Of course it is a given that any company would be happy to be facing too many successful projects. As a strong advocate of XP, I want to see XP, and other agile methodologies embraced by the software development community at large. But although XP works well for a single project, we need discussion and definition for corporate XP. Projects that share dependencies on each other or projects within a company (large enough that having one large XP project simply does not make sense) will need some common focus and steering to prevent the integra-

tion issues of the past, or that which I have called extreme chaos.

Maybe it makes sense to create customers for a customer, or maybe XP might help us to create a company customer that can act as a benign corporate guide. Or maybe there are many other emerging solutions that will help synchronize XP projects. My hope is that this paper will spawn more conversation in this area within the XP and agile communities at large.

ACKNOWLEDGEMENTS

Similar to many notes I see in XP publications, this document is merely a distillation of many coffee conversations. The majority of these conversations focused on discussing the way in which XP was or was not working on one or more XP projects, and why. If fact, I often find a new understanding of XP when I am asked to state or defend why I think XP will help any development effort.

I gratefully acknowledge and thank all individuals that suffer though my tangential ranting on a daily basis. I believe that developing software is a process of people using a language that, like the notes on a page of sheet music, merely represent a collection of information, meaningless until it is used or played in harmony.

REFERENCES

1. Planning and Running an XP Iteration" by Cara Taber and Martin Fowler on-line at: http://martinfowler.com