Implementing Value-Based Extreme Programming

by Monica Yap
Implementing Value-Based Extreme Programming
by Monica Yap

Introduction

Agile methods such as Scrum and Extreme Programming are not known for carefully tracking to time and cost estimates. On most projects, schedule slips are common and cost increases are predictable. At the end of every iteration some stories get dropped, usually because the scope or some other factor of the story wasn’t fully understood. Every slip and increase reduces total business value, and project overruns force other projects to be delayed or canceled.

This negative value impact could be avoided. Agile methods must encourage accountability for on-time delivery and scope based on value and cost. What is missing in most Agile implementations is a value-based feedback mechanism involving shared responsibility between the customer and the team. This white paper describes how one company addressed these challenges and the lessons we learned along the way. We’ll discuss how we achieved on-time delivery, assisted customers in selecting high value features, provided shared responsibility, and facilitated individual team member empowerment.

Background

Since 1999, I have introduced Extreme Programming (XP) and Scrum practices, built Agile teams, and participated in Agile development within multiple companies. I have filled the roles of Scrum Master, developer, coach, and development manager. I have experienced tremendous positive effects from incorporating Extreme Programming practices into development teams. However, I have yet to find the common Agile methods that can guarantee to deliver business value in each iteration, regardless of the requirement. In this article I will address why this is critical and how it might be achieved.

Wireless Data Services (WDS) Global offers services to wireless companies and mobile phone manufacturers. Services include custom-built and hosted Web sites for mobile phone setup, customer service, and self-help applications. Due to fast growth in the wireless industry, most of WDS’s customers are extremely sensitive to delivery dates and are struggling to keep up in a rapidly changing market.

The following is an account of my experiences working with the team at WDS from February 2004 to October 2007, when I was head of group development. Typical WDS projects were multi-release Web sites where each release took one or two weeks. We had three project teams around the globe with each XP team consisting of six to ten developers. Our customers were usually offsite. In any given two-week iteration, each team was working on multiple projects for different customers.
**Use of Extreme Programming**

We have employed XP methodology and practices [1] for more than two years. These practices include:

- User stories
- Planning game
- Short iterations
- Test-driven development
- Refactoring
- Pair programming
- Continuous integration
- Collective code ownership

We have achieved great results with XP, including regular iterative delivery — usually pushing features to production each week — and providing customers with quick time to market, low defect rates, and a rapid response to changes.

**Identifying Challenges**

After the successful integration of XP practices, we started to look at other areas around development that needed improvement. This section discusses the additional challenges we faced.

**Prioritization with Multiple Customers**

We were serving multiple customers simultaneously with no objective way to balance and prioritize their needs. When faced with simultaneous requests from multiple customers, the resource allocation sometimes went to the customer with the highest future sales potential, or to whichever customer shouted the loudest, and was not based on the project’s potential business value. For example, if customer X is a critical customer, we may develop for customer X’s project with an expectation of 50 percent profitability even though customer Y’s project was expected to generate a much higher 70 percent profitability.

**Low-Value Requests**

We would typically do whatever an important customer requested, including developing unnecessary or low-value features. For example, a customer might request UI features that wouldn’t enhance the overall functionality of the Web tool. This extra work caused project overruns and an ongoing maintenance burden.

**Changing Unclear Scope & Requirements**

Due to unclear scope and requirements, we did not always meet our commitments. This caused delays to other projects and customer dissatisfaction. A typical example:

*Customer X:* “Please build us one just like this other one you have built, with minor adjustments to styling and steps.”

*Development:* “Sure, no problem, we can deliver that in two weeks.”
Customer X (one week later): “Our marketing has re-designed the styling, so the new styling is this, and we would like all the steps in one page instead of 5 pages.”

Development: “But the site has already been built with the original style and the pages are built as well, we will need an extra week to complete these changes.”

Customer X: “You mean you cannot deliver as you promised us last week?”

Lack of Individual Responsibility
With collective code ownership and other team practices, individual developers did not feel personally accountable for meeting delivery dates and scope commitments.

Addressing Challenges
These challenges are common to Extreme Programming teams. The following steps can be taken to address these challenges.

Prioritize Across Multiple Projects
The project sponsor should make priority decisions based on total project value, including development effort.

Meet Commitments, Control Changes
In most cases where stories are not completed, it’s due to an underestimation of the story scope, or significant scope change by the customer. The customer and development must work together to determine the scope of the story and be accountable for any drastic changes.

When scope change is proposed, tradeoffs must be made based on the timeline and budget. Development should give an estimate based on their confidence level, and both parties should restructure the stories accordingly.

Eliminate Excess Features Early
The Extreme Programming’s YAGNI principle (You Ain’t Going to Need It) [1] relates to software design, and the same principle should be applied to story scope. Both parties should be committed to delivering in the shortest time possible with no excess or low-value features.

Encourage Personal Involvement
Encourage more individual team member involvement by allowing them to take ownership of the stories.

Improving Processes
Over time, we identified four Agile practices that merge the XP principles of implement highest value features first and don’t do anything extra with Lean software principles such as eliminate waste [2]. These new practices helped us to address these critical areas:

• Value-based investment decisions
• High-confidence stories first
• Incremental story delivery
• Story ownership

Based on our experience, they are most effective when applied together. However, they can still provide tremendous value when used in isolation.

Value-Based Investment Decisions

Problem: Development resources are often wasted on low-value projects. Much of the value gained from completed projects is lost due to schedule overruns and the costs of delaying or canceling other projects.

Context: Development resources are often seen as a fixed cost and projects are assigned for reasons other than creating the highest overall business value. As a fixed cost, development time is treated as a sunk cost and no value is placed on it. The lack of project sponsor accountability for total business value created by projects leads to improper business decisions. Finally, without a feedback loop it is difficult to recognize, much less correct, either problem.

Solution: To make informed decision on project selection so that scarce development resources are allocated to projects that provide the highest value, the project sponsors need financial feedback in terms of business value.

• Charge project sponsors for ongoing development costs to remove the fixed cost nature of development.
• When scope increases are suggested, inform the sponsor of the additional charges.
• Any potential slip in project schedule due to scope changes should be balanced against the cost of delaying future projects.
• Perform a retrospective to determine if the project provided adequate value based on the full cost.

Conclusions: Incrementally paying for resources as they are used creates a natural self-limiting effect that leads to optimized usage of costly development time. Value-based reviews expose poor investments and help project sponsors make better decisions on future development investments.

High-Confidence Stories First

Problem: Risky work items are often badly estimated, causing schedule overruns and lost value.

Context: Conventional wisdom suggests that high-risk work items be attacked first as they are most likely to jeopardize the schedule. But this may so dramatically impact the schedule that critical stories are pushed to the next iteration, resulting in a ripple effect that prevents whole features from being implemented.

Solution: At the iteration planning meeting, the customer comes with a set of stories of relatively the same priority as candidate work items for the iteration. The goal is to deliver as many of the candidate stories as possible by the end of the iteration.
WHITE PAPER: Implementing Value-Based Extreme Programming

1. Separate stories into two piles: known and risky. Known stories are those you feel confident can be completed within the estimated time. Risky stories aren't as well defined so their estimates are uncertain.

2. For the stories in the known pile, offer fixed-quote estimates based on past performance.

3. Implement and deliver all the known stories before beginning any risky stories.

4. If possible, break risky stories into a known story and a smaller risky story and complete the known portion first.

5. For the stories remaining in the risky pile, use incremental story delivery (discussed in the next section) to further lower the risk.

Figure 1 illustrates this approach.

Conclusion: Delivered stories accumulate value, incomplete stories have a negative value, and stories that have not been worked on have no value. Therefore, delivering more stories within an iteration is preferred over wasting time on risky stories that will jeopardize the schedule and cause less risky stories to be dropped from the iteration.

Incremental Story Delivery

Problem: Lack of commitment to deliver agreed upon stories for each iteration.

Context: During planning, developers commit to achieving specific story goals within an iteration. Once the iteration begins, it is typical for stories to take longer than the estimates, causing other stories to be dropped. Each story dropped is lost value. Both the customer and the team must stay focused on delivering value by actively managing story scope.

Solution:
- Commit to completing all stories selected for the iteration.
- Prioritize known stories with high confidence estimates over risky stories with low-confidence estimates.
- For the risky stories, continuously refocus effort and reduce scope to deliver all or a useful subpart of the story objectives within the estimated timeframe. This fine-grained scope management requires frequent customer interaction.
• Deliver the story within the targeted timeframe and put any leftover objectives into a new story for a future iteration.
• When time runs out for a risky story, put it on hold and move it to the bottom of the iteration list.
• At the end of the iteration, determine if there is time left to complete the reprioritized tasks. If not, remove them from the iteration.

**Conclusion:** The delivery of tightly focused stories reduces the introduction of unneeded features, while ensuring continuous delivery of the maximum number of high-value stories. Halting stories that slip helps reinforce the practices of focusing the story and incrementally delivering the story within the timeframe. Delivering on commitments builds customer trust.

### Story Ownership

**Problem:** Stories are not being delivered within the targeted timeframe.

**Context:** Agile teams lack both a feedback loop and an attitude of individual responsibility.

**Solution:** Hold development team members responsible for managing the scope and cost of stories from the time the project is first proposed.

• Development team members assist in breaking the project into stories. Stories are initially used for estimating time and costs.
• One team member signs up to be the story owner.
• The story owner communicates with the customer to understand the story and to create an estimate.
• The story owner negotiates the scope with the customer to reduce overall costs of the project.
• The story owner and customer negotiate the cost of the story. Stories with a known cost use a fixed pricing model. Uncertain stories are more likely to use a target cost or time-and-materials pricing model. Customers should assist in shaping stories so that the majority of stories have fixed costs.
• Once the project begins, the story owner tracks the story to make sure it stays on scope and comes in on time.
• The story owner acts as the point person for any discussions with the customer about the story scope or status.
• During the iteration the team reviews each story owner’s effectiveness by comparing time and scope delivered to original expectations.

**Conclusion:** To ensure on-time delivery, the project should be broken down into the smallest possible deliverables, with each deliverable carefully tracked and managed to ensure overall success. The story owner acts as the shepherd to deliver the story with the smallest scope and costs, ensuring the highest value.
**Results**

When we first introduced the budget and pricing model changes, a lot of concerns were raised, but we knew the changes could help solve our problems. Over time we discovered the other practices and put them in place incrementally. Though we have only been using these strategies for a few months, they have addressed the worst of our problems and have had a clear positive impact. I often hear both development and project sponsors asking customers questions such as, “What value does this feature provide for you, and is it worth the cost?” An unexpected benefit was the change in the way the business treats and communicates with development. Both groups have now formed a cross-functional team and work together to achieve common goals.

**Cost Accountability**

We enabled development to manage its own budget and track actual development costs. The development team began estimating project cost in terms of development time. Tracking budgets and costs exposes the correlation between revenue and expenses.

**Team Member Empowerment**

Our development team member acting as story owner has helped us to manage the budget for the project, determine pricing model, and ensure stories are successfully delivered.

**Elimination of Waste**

Our team member acting as story owner begins working with customers when the project is first proposed. This provides cost feedback on the project as a whole, which then helps customer refine the minimum set of features needed. With the reduction of unneeded features, development is able to deliver them in the shortest time possible.
On-Time Delivery
We have been able to meet our commitments with a few exceptions.

- Forty-seven percent of iterations are delivered on time.
- Thirty-five percent delivered more than what was committed.
- Eighteen percent of commitments are not fully met.

Conclusion
Lean and Extreme Programming techniques both focus on providing value to the customer and on eliminating waste. They complement each other by appealing to different audiences within an organization while providing everyone with tools for achieving the highest possible business value. Using Extreme Programming helps development implement engineering practices that ensure a high-quality product. Employing Lean principles [2] helps development translate estimates into terms such as cost and value, which are understood by the business. The cost and value exposure helps customers avoid the expense of unnecessary product features.

About the Author
Monica Yap is an Engagement Manager and Agile Coach for SolutionsIQ, focusing on building and leading successful Agile teams for outsource projects. Monica has over sixteen years of development experience in e-commerce, wireless, and various other industries, as well as over eight years of experience leading agile teams, providing quality products through the use of continuous refactoring, unit testing, pair programming, small releases, and an evolving architecture. She has worked with distributed Agile teams over four years (U.S., UK, and Singapore). She and her teams have demonstrated tremendous successes, with 90 percent of the iterations being delivered on time, meeting or exceeding customer expectations.

Monica is a regular presenter at seminars (including Agile2005, Agile2006, and Agile2007) and volunteer for technical and process groups, including the Seattle Agile user group, Seattle Java user group, and Seattle APLN.

References

About SolutionsIQ
SolutionsIQ offers a full spectrum of services to develop software and fulfill technical talent needs, while improving your Agile knowledge and capabilities. Clients include AT&T (Cingular), Amazon, Corbis, Expedia, Federal Home Loan Bank, Infospace, Key Bank, Nike, Nordstrom, Safeco, US Bank, and Washington State University. A Microsoft Gold Certified Partner, SolutionsIQ is also a member of the Java Community Process, Scrum Alliance, Software Association of Oregon, and Washington Technology Industry Association. Learn more at www.SolutionsIQ.com.