Value-based Extreme Programming
Monica Yap
WDSGlobal
10809 120th Avenue NE
Kirkland, WA 98033, USA
monica.yap@us.wdsglobal.com

1. Abstract

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 of our stories get dropped, usually due to reasons such as “the story took longer than what we expected”, or “we didn’t find out scope X about the story until we were in the middle of the iteration”. Every slip and increase reduces total business value delivered and project overruns force other high-value projects to be delayed or canceled. Together these costs cause a large negative value impact that could be avoided. Agile needs a better way to measure total value delivered in relation to cost. 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 experience report describes how one company addressed these challenges and the lessons learned: it explains how we achieved on-time delivery, assisted customers in selecting high value features, provided shared responsibility, and facilitated individual team member empowerment. All the practices used are presented in pattern format.

2. Introduction

2.1 My personal background with agile
Since 1999, I have introduced Extreme Programming and Scrum practices, built agile teams and participated in agile development within multiple companies. I have played the roles of Scrum Master, Developer, Coach and Development Manager. I have experienced tremendous positive effects from utilizing Extreme Programming practices on development teams. However, I have yet to find the agile path to regularly providing positive business value. I feel that this is the final end goal for Agile and, while I am not there yet, I will explain what we have achieved and what is left to be done.

2.2 The company
At Wireless Data Services Global, we offer services to wireless companies and mobile phone manufacturers. Due to the fast growing nature of the wireless industry, the majority of our customers are extremely sensitive to delivery dates and are struggling to keep up in a rapidly changing market place. Our services include custom built and hosted web sites for mobile phone setup, customer service and self-help applications.

2.3 The nature of projects
Our typical projects are multi-release web sites where each release is typically one or two weeks. We have three project teams around the globe with each XP team consisting of 6-10 developers and our customers are usually off-
In any given two week iteration, each team is developing for multiple projects with different customers.

2.4 Extreme Programming applied

We have employed the Extreme Programming methodology and practices [1] for over two years. These practices include:

- user stories
- the planning game
- short iterations
- test driven development
- refactoring
- pair programming
- continuous integration
- collective code ownership

We have had great results from these practices; we have achieved regular iterative delivery, usually pushing features to production each week, providing our customers with quick time to market, low defect rates, and a rapid response to changes.

3. Seeking the next level

With the development process under our belt, we started to look at other areas around development that needed improvement, and found the following problems.

3.1 Balancing and prioritizing across multiple customer needs

Serving multiple customers simultaneously left our company with no objective way to balance and prioritize across multiple customers’ needs. When there were simultaneous requests from multiple customers, the development 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 business value potential. For example, if customer X is a critical customer, we may develop for customer X’s project with an expectation of 50% profitability even though customer Y’s project was expected to generate a much higher 70% profitability.

3.2 Doing all that’s been requested

We would typically do whatever the important customer specified, including unnecessary or low value features, even though we are practicing the planning game with cross-project prioritization. This extra work caused project overruns and an ongoing maintenance burden. For example, customer X may ask for UI features that have no contribution to the overall functionality of the web tool.

3.3 Changing scope

We did not always meet our commitments due to unclear scope and fuzzy requirements. Slipped commitments caused other projects to be delayed and other customers to become unhappy. Meeting commitments and delivery dates was the responsibility of the development team, and the customers were not accountable for the impact of drastic scope changes on the delivery date. 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?”

3.4 Feeling responsible

With collective code ownership and other team practices, individual developers were not motivated to have personal involvement in the projects, and lacked responsibility and
accountability for hitting delivery dates and scope commitments.

4. What should be added or strengthen

I have seen these problems in previous Extreme Programming teams as well. Therefore, I started to search for commonalities and root causes and discovered the elements that needed to be added or strengthened:

4.1 Help prioritize stories across multiple projects

Our project sponsor should make priority decision on total project value including development effort. We applied value-based investment decision making (section 5.1).

4.2 Ensure commitments are met and changes are controlled

I’ve frequently seen iterations where stories were signed-up for and not completed. Mainly due to two reasons:

a) The scope of a particular story was changed drastically by the customer. It was impossible to complete the story within the iteration.

b) The story was under estimated.

So both the customer and development should work together to determine the scope of the story and be accountable for any drastic changes in scope. Whenever scope change is proposed, both parties should make trade-offs based on the agreed time commitment and budget. See section 5.4 Story Ownership

Development should give an estimate based on their confidence level, and both parties should restructure the stories accordingly. See section 5.2 High Confidence Stories First

4.3 Eliminate excess features from the beginning

The Extreme Programming’s YAGNI principle (You An’t going to Need It) [1] relating to software design, the same principle should be applied to story scope as well.

Focus on delivering in the shortest time possible with no excess or low value scope. Both parties should be committed to delivering only the most valuable portions of the agreed upon features within the allotted time. See section 5.3 Incremental Story Delivery

4.4 Encourage personal involvement

We can encourage more individual team member involvement by allowing them to take ownership of the stories. See section 5.4 Story Ownership

5. The practices we used

Over time, we have found a family of four agile practices that merge the Extreme Programming principles of implementing the “highest value features first”, and “don’t do anything extra”, with Lean Software principles such as “eliminate waste” [2] to address the highlighted issues: Value-based Investment Decisions, High Confidence Stories First, Incremental Story Delivery, and Story Ownership.

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

5.1 Value-based Investment Decisions

Name: Value-based Investment Decisions

Category: Project and Feature Selection

Problem: Development resources are often wasted on low value projects. Much of the value gained from completed projects is lost due to project schedule overruns and the accompanying lost opportunity 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.

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

Conclusions:
Incrementally paying for resources as they are used provides a natural self-limiting effect that leads to optimized usage of scarce and costly development time.

Value-based reviews expose poor investments and help project sponsors make better decisions on future development investments.

5.2 High Confidence Stories First
Name: High Confidence Stories First
Category: Iteration Story Prioritization

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

Context: Common wisdom suggests that high risk work items be attacked first as they are most likely to blow the time schedule. However, the blown schedule means we get less work into production. Agile teams often run out of time and are forced to drop stories that they have agreed to complete. Dropped stories wind-up in the next iteration thereby displacing other stories the ripple effect causes whole features to never be implemented due to schedule constraints.

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.

a. Separate stories into two piles: known and risky. Known stories are those we have a high confidence in completing within the estimated time. Risky stories are those whose estimates are less likely to be accurate as they contain uncertainty.

b. For the stories in the known pile, offer fixed-quote estimates based on "yesterday’s weather" and past performance.

c. Implement and deliver the fixed-quote stories first before beginning any stories in the risky pile.

d. If possible, split risky stories into a known story and a smaller risky story whose estimate will be less uncertain and then do the known part first.

Value-based reviews expose poor investments and help project sponsors make better decisions on future development investments.

e. For the stories remaining in the risky pile, use incremental story delivery to further lower the risk.
The following diagram demonstrates stories within a iteration applying this practice.

**Iteration with Velocity 30**

<table>
<thead>
<tr>
<th>P1</th>
<th>P2</th>
<th>P3</th>
</tr>
</thead>
<tbody>
<tr>
<td>Estimate 15</td>
<td>Estimate 10</td>
<td>Estimate 5</td>
</tr>
<tr>
<td>some unknowns</td>
<td>all known</td>
<td>all unknowns</td>
</tr>
</tbody>
</table>

**Using “High Confidence First”**

<table>
<thead>
<tr>
<th>P1</th>
<th>P2</th>
<th>P3</th>
</tr>
</thead>
<tbody>
<tr>
<td>Estimate 12</td>
<td>Estimate 10</td>
<td>Estimate 3</td>
</tr>
<tr>
<td>fixed-quote</td>
<td>fixed-quoted</td>
<td>incremental delivery (timebox)</td>
</tr>
</tbody>
</table>

**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 likely run over time and cause less risky stories to be dropped from the Iteration. So do the less risky stories first to maximize value delivered.

### 5.3 Incremental Story Delivery

**Name:** Incremental Story Delivery  
**Category:** Story Implementation  
**Problem:** Lack of commitment to deliver signed-up for stories in each iteration.

**Context:** During planning, developers commit to achieving story goals within an estimated time and signup for an iteration's worth of stories. However, 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. Much of the loss could be prevented if a finer focus was placed on delivering all or most of the story goals within the estimated time. Both the customer and the team must stay focused on delivering value by actively managing story scope.

**Solution:**
- Commit to completing all stories signed up for in an iteration.
- Prioritize estimated stories with high confidence estimates over risky stories with low confidence estimates.
- For the risky stories, use the story’s estimated hours as a time-box, continuously refocusing effort and reducing scope to deliver all or a useful subpart of the story objectives within the estimated time frame. This fine grained scope management requires frequent customer interaction, as scope is as much a part of the commitment as is the time allocated.
- Deliver the story within the target time and put any leftover objectives on a new story to be scheduled in a future iteration.
- When the budgeted time runs out for a risky Story place it on hold and reprioritize 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 and stay focused on delivering the greatest number of stories within the Iteration.

**Conclusion:** The delivering 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 time box. Delivering on commitments builds and keeps customer trust.

5.4 Story Ownership
Name: Story Ownership
Category: Individual involvement and low-level scope management
Problem: Stories are not being delivered within the targeted time.

Context: Agile teams are missing both a feedback loop and an attitude of individual responsibility to help the team put forward its best effort by focusing on the goals of meeting the customer commitment and following the Agile practice of avoiding unnecessary effort.

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

- When the project is proposed, development team members assist in breaking the project down into stories. Stories are initially used for time estimating, and predicting the cost of the project.
- As the stories are created, one of the development team members signs up to be the Story Owner.
- The Story Owner communicates with the Customer to understand the story and to derive an estimate.
- The Story Owner negotiates with the customer on the scope of the story in order to reduce the overall costs of the project.
- Part of this negotiation involves selecting the correct pricing model for the story. Predictable stories with a known cost use a fixed pricing model.
- Uncertain stories with a variable cost are more likely to use a target cost or time and materials pricing model.
- Customers are encouraged to prefer the fixed pricing model and will assist in shaping stories so that the story costs can be fixed.
- 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 when the story goes off track and cannot be completed.
- During the Iteration the team reviews each Story Owners’ effectiveness by comparing time and scope delivered to original expectations.

Conclusion: To meet the overall objective of delivering the project on time, 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 cost in time, thereby insuring the highest overall value delivered.

6. Results

When we first introduced the budget and pricing model changes, a lot of concerns were raised, but we stuck with it as we knew the changes could help solve our problems. Over a period of time, we discovered the other practices and put them in-place incrementally. Even though we have only implemented these strategies for a few months, the combination of these practices address the worst of our problems and have had a clear positive impact. Now, 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 that occurred 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.

6.1 Measure on value  
We enabled development to start managing its own budget and had them 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.

6.2 Team member empowerment  
Our development team member acting as story owner has empowered us to assist in managing the budget for the project, determine pricing model, and ensure the successfulness of the stories.

6.3 Eliminate waste from the beginning  
Our team member acting as story owner works with customer at the beginning of proposal and pre-planning stage. 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, transferred into stories, development is able to deliver them in shortest time possible to market.

6.4 Deliver on time  
We have been able to meet commitment with a few exceptions; 47% of iterations are delivered on time, 35% delivered more than what was committed, 18% commitment not fully met.

Lean and Extreme Programming both focus on providing value to the customer and on eliminating waste. They compliment each other by appealing to different audiences within an
organization while providing everyone with tools for achieving the highest possible business value creation. Using Extreme Programming helps development to implement engineering practices for ensuring a high quality product. Employing lean principles [2] helps development to translate estimates into terms such as cost and value, which are understood by the business. The cost and value exposure helps guide our customers toward avoiding the production of unused product features.

7. Acknowledgements

I would like to thank all the team members for their cooperation and hard working on the projects, and various people in WDSGlobal for giving advice on this paper. I also thank Giovanni Asproni for providing valuable guidance and encouragement in shepherding this paper.

8. Speaker Qualifications

Monica Yap is the Extreme Programming Coach and Head of Group Development for WDS Global. Monica has designed and developed software for the past 15 years on a wide variety of projects from on-line eCommerce sites to property tax management systems. She has over six years of experience with agile processes. In particular, Monica has focused on building agile teams and merging business concerns with an agile process. She has presented in Agile2005, local Extreme Programming groups, and taught classes in agile processes.

9. References
