The Business Case for Scrum

Peter Borsella, PMP
From what you have heard about Scrum, what are some of the reasons why an organization would choose Scrum as a way of delivering their product?

What benefits might be expected?
The Business Case for Scrum

Things we’ll talk about:
- The complexity of work
- A short review of Scrum
- The definition of “done”
- The business case: Delays and Risk
- The business case: Early Opportunity
- The business case: Earned Value

And, time permitting:
- A case study: ROI
- Prioritization Considerations
- The Agile Manifesto

Peter Borsella, PMP
The Complexity of Projects

Now let’s add people

- Culture
- Time Zones
- Language
- Personalities

And there’s more

Graph taken from Ralph Stacey’s “Complexity and Creativity in Organizations,” Copyright Ralph Stacey, 1996
See also "A Leader’s Framework for Decision Making," by David J. Snowden and Mary E. Boone, HBR reprint R0711c
Change & Unknowns

External Customers

Competition

Regulations

Internal Customers

Change & the Unknown
We are changing exponentially, not linearly

We are doubling the rate of technological progress every decade

At today’s rate of advance, this means that the past 100 years has actually seen the equivalent of 20,000 years of linear advances!

Taken from “The Law of Accelerating Returns” by Ray Kurzweil (www.kurzweilai.net); see also Moore’s Law
Now let’s add people
- Culture
- Time Zones
- Language
- Personalities

And there’s more
Scrum helps us take small pieces of upper-right and bring them towards lower-left.

And here’s what it looks like

Graph taken from Ralph Stacey’s “Complexity and Creativity in Organizations,” Copyright Ralph Stacey, 1996
See also "A Leader’s Framework for Decision Making," by David J. Snowden and Mary E. Boone, HBR reprint R0711c
Scrum Overview

The Scrum Team:
- Dev Team
- Product Owner
- ScrumMaster
Scrum Is Like

With your teammates choose ONE of the following pictures as your guide, and finish this sentence:
Scrum is like [photo of your choice] because __________________.
Let’s explore one of the specific expectations of working in a Scrum environment.
“Done”

Design
Coding
Unit Testing
Test Plans

Done

Done

Done
“Done”

Design
Coding
Unit Testing
Test Plans

“Done Done”

UAT
Separate the grain from the chaff

Planning
Analysis
Design
Coding
Unit Testing
System Testing
Performance Testing
User Acceptance
Production Implementation
Launch Pilot
Launch First Customer

Done
“Done”

Design
Coding
Unit Testing
Test Plans

“Done Done”

UAT
Performance Testing
Done

Planning
Analysis
Design
Coding
Unit Testing
System Testing
Performance Testing
User Acceptance
Production Implementation
Launch Pilot
Launch First Customer
Done

Planning
Analysis
Design
Coding
Unit Testing
System Testing
Performance Testing
User Acceptance
Production Implementation
Launch Pilot
Launch First Customer
“Done”

Design
 Coding
 Unit Testing
 Test Plans

“Done Done”

UAT
 Performance Testing
 Integration
 Regression

Separate the grain from the chaff
Why Change?

Traditional Approach

Built-in **DELAYS** associated with:
- Accumulation of new product needed to offset expense of a heavy implementation cost
- Lengthy implementation periods
- Required change difficulties in large plans

Built-in **RISK** associated with:
- Buildup of product functionality that has not been fully tested
- Big Bang approach to project funds allocations (ROI risk)
- Inflexibility in responding to beneficial changes
Why Change?

Traditional Approach

“building:” coding, unit testing, etc.

Agile Approach

production-ready
definition of “done?”

Implementation-ready at any time?

IF we can shrink this phase down to something less costly
Done

“Done”

Design
Coding
Unit Testing
Test Plans
~~~~~~~~

“Done Done”

UAT
Performance Testing
Integration
Regression

By moving more and more into our definition of “done.”
Why Change?

Traditional Approach

- “building:” coding, unit testing, etc.
- Integrate, Test, etc.

Agile Approach

- production-ready
- definition of “done”
- Implementation-ready at any time
Why Change?

We do not get this for FREE! There is a cost:
- the cost of organizational change, and
- the cost of technology (tools)

So, the cost of getting to this needs to be compared to the cost of operating with this.

Agile Approach

production-ready definition of “done”  Implementation-ready at any time
Why Change?

Traditional Approach

Built-in **DELAYS** associated with:
- Accumulation of new product needed to offset expense of a heavy implementation cost
- Lengthy implementation periods
- Required changes difficulties in large plans

Built-in **RISK** associated with:
- Buildup of product functionality that has not been fully tested
- Big Bang approach to project funds allocations (ROI risk)
- Inflexibility in responding to beneficial changes
Exercise!

When are we “done?”

-what does “done” mean on your current project?

-do you have any issues with this definition? Any engineering problems?

-How might you change things?

Planning
Analysis
Design
Coding
Unit Testing
System Testing
Performance Testing
User Acceptance
Production Implementation
Launch Pilot
Launch First Customer

Done

Exercise!

When are we “done?”

-what does “done” mean on your current project?

-do you have any issues with this definition? Any engineering problems?

-How might you change things?
### Return on Investment

<table>
<thead>
<tr>
<th>Budgetary allocation:</th>
<th>$820,000</th>
</tr>
</thead>
<tbody>
<tr>
<td>Project length (months):</td>
<td>18</td>
</tr>
<tr>
<td>Monthly budget:</td>
<td>$45,556</td>
</tr>
<tr>
<td>Monthly revenue expected:</td>
<td>$70,000</td>
</tr>
<tr>
<td>Monthly expenses:</td>
<td>$15,000</td>
</tr>
</tbody>
</table>

**Assumptions:**
- Projected monthly revenue and expenses are fixed
- Project is delivered successfully
Return on Investment

Budgetary allocation: $820,000
Project length (months): 18
Monthly budget: $45,556
Monthly revenue expected: $70,000
Monthly expenses: $15,000

Release after month 18, and then stop development
Return on Investment

Using the same assumptions, what if we could do the following:
- Prioritize the most valuable deliverables
- Phase releases beginning with an initial early release
- Verification checks of the market response after each release

<table>
<thead>
<tr>
<th>Budgetary allocation:</th>
<th>$820,000</th>
</tr>
</thead>
<tbody>
<tr>
<td>Project length (months):</td>
<td>18</td>
</tr>
<tr>
<td>Monthly budget:</td>
<td>$45,556</td>
</tr>
<tr>
<td>Monthly revenue expected:</td>
<td>$70,000</td>
</tr>
<tr>
<td>Monthly expenses:</td>
<td>$15,000</td>
</tr>
<tr>
<td>Progressive Revenue (&gt; 1st release)</td>
<td>$10,000</td>
</tr>
<tr>
<td>Progressive Revenue (&gt; 2nd release)</td>
<td>$20,000</td>
</tr>
<tr>
<td>Progressive Revenue (&gt; 3rd release)</td>
<td>$40,000</td>
</tr>
<tr>
<td>Progressive Revenue (&gt; 4th release)</td>
<td>$50,000</td>
</tr>
<tr>
<td>Progressive Expense (&gt; 1st release)</td>
<td>$5,000</td>
</tr>
<tr>
<td>Progressive Expense (&gt; 2nd release)</td>
<td>$10,000</td>
</tr>
<tr>
<td>Progressive Expense (&gt;3rd release)</td>
<td>$15,000</td>
</tr>
</tbody>
</table>
In addition to an earlier breakeven point, we could free up the team to work on more valuable endeavors.

**Return on Investment**

<table>
<thead>
<tr>
<th>Monthly Revenues</th>
<th>Monthly Expenses</th>
</tr>
</thead>
<tbody>
<tr>
<td>Progressive Revenue (&gt;1st release)</td>
<td>$10,000</td>
</tr>
<tr>
<td>Progressive Revenue (&gt;2nd release)</td>
<td>$20,000</td>
</tr>
<tr>
<td>Progressive Revenue (&gt;3rd release)</td>
<td>$40,000</td>
</tr>
<tr>
<td>Progressive Revenue (&gt;4th release)</td>
<td>$50,000</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Monthly Expenses</th>
<th>Budgetary Allocation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Progressive Expense (&gt;1st release)</td>
<td>$5,000</td>
</tr>
<tr>
<td>Progressive Expense (&gt;2nd release)</td>
<td>$10,000</td>
</tr>
<tr>
<td>Progressive Expense (&gt;3rd release)</td>
<td>$15,000</td>
</tr>
</tbody>
</table>

**Budgetary allocation:** $820,000

**Project length (months):** 18

**Monthly budget:** $45,556

**Monthly revenue expected:** $70,000

**Monthly expenses:** $15,000

**Progressive Revenue:**
- (>1st release) $10,000
- (>2nd release) $20,000
- (>3rd release) $40,000
- (>4th release) $50,000

**Progressive Expense:**
- (>1st release) $5,000
- (>2nd release) $10,000
- (>3rd release) $15,000

In addition to an earlier breakeven point, we could free up the team to work on more valuable endeavors.

Releases after months 4, 7, 10, 13, and 16, and then stop development.
Managing Earned Business Value, A New Metric?

Earned Business Value = Value of Completed Deliverables to date
Case Study! Documentation

After four years of use, an internal CRM system needs to be re-written. The original cost to write the system was $3MM, and the estimated cost of the re-write is $2MM. The reduced cost is based on an assumption that we should already be familiar with the system. However, there was no technical documentation created for the original system, so now the re-write is costing more than expected at an estimated additional cost of $1.5MM for a total of $3.5MM.

The CIO complains, “We should have spent the time writing the documentation! From now on, all systems development will include technical documentation.”

Discuss with a small group around you the merits of and/or flaws in the CIO’s new directive.
Prioritization
Prioritization Considerations

Questions to ask about each feature when assessing value:

- How much money will this feature make or save? 

- How much money will this feature cost (including ongoing support)?

- How much more will we understand about the product and the project by developing this feature?

- How much risk can we reduce by developing this feature?

(usually the driving force behind prioritization)

(usually modifiers)
Prioritization Considerations

How much money will this feature make or save?

Considerations:

- **New Business** → potential revenue
- **Incremental business** → potential revenue
- **Customer Retention** → potential loss
- **Operational Savings** → potential cost

A significant amount of time and research may be required to understand these factors.
How much money will this feature cost (including ongoing support)?

Considerations:

- Team Size
- Cost to Company
- Iteration Size
- Contribution Time Percentage

The end result will be a dollar cost per iteration. We could also use information to deduce the cost per unit of work (story point, e.g.).
Prioritization Considerations

How much more will we understand about the product and the project by developing this feature?

Traditional Approach

H Means
Uncertainty: Project Knowledge, or “How”

Agile Approach

H Means
Uncertainty: Project Knowledge, or “How”

Credit to Alexander Laufer, “Simultaneous Management”
Prioritization Considerations

How much risk can we reduce by developing this feature?

Specific credit to Mike Cohn, "Agile Estimating and Planning"
“We are uncovering better ways of developing software by doing it and helping others do it. Through this work we have come to value:

- **Individuals and interactions** over processes and tools
- **Working software** over comprehensive documentation
- **Customer collaboration** over contract negotiation
- **Responding to change** over following a plan

That is, while there is value in the items on the right, we value the items on the left more.**”

*(cont’d on last slides)*

* [www.agilemanifesto.org](http://www.agilemanifesto.org) : © 2001. This declaration may be freely copied in any form, but only in its entirety through this notice*
Principles behind the Agile Manifesto (cont’d from intro)

I. “We follow these principles: Our highest priority is to satisfy the customer through early and continuous delivery of valuable software.

II. Welcome changing requirements, even late in development. Agile processes harness change for the customer's competitive advantage.
Agile Thinking

III. Deliver working software frequently, from a couple of weeks to a couple of months, with a preference to the shorter timescale.

IV. Business people and developers must work together daily throughout the project.

V. Build projects around motivated individuals. Give them the environment and support they need, and trust them to get the job done.

www.agilemanifesto.org: © 2001. This declaration may be freely copied in any form, but only in its entirety through this notice
Agile Thinking

VI. The most efficient and effective method of conveying information to and within a development team is face-to-face conversation.

VII. Working software is the primary measure of progress.

VIII. Agile processes promote sustainable development. The sponsors, developers, and users should be able to maintain a constant pace indefinitely.

www.agilemanifesto.org: © 2001. This declaration may be freely copied in any form, but only in its entirety through this notice.
Agile Thinking

IX. Continuous attention to technical excellence and good design enhances agility.

X. Simplicity—the art of maximizing the amount of work not done—is essential.

XI. The best architectures, requirements, and designs emerge from self-organizing teams.

XII. At regular intervals, the team reflects on how to become more effective, then tunes and adjusts its behavior accordingly.”

www.agilemanifesto.org: © 2001. This declaration may be freely copied in any form, but only in its entirety through this notice.