Does Agile Scale?
A Short Case Study
April 9, 2015
Agenda

- Introduction
- Agile Overview
- Agile Projects Analysis: A Quantitative Approach
- An Organizational Case Study
Introduction

During these times of economic austerity everyone is looking for a competitive edge. It’s not surprising then, that solutions which promote decreased time to market and increased productivity would be appealing.

Is Agile the “silver bullet” we’ve all been looking for?

The “Q” in QSM stands for Quantitative. So that’s what we did. We put numbers to assessments of software projects and estimates to see: “Does Agile Scale?”
What is Agile Software Development

• In the late 90’s several methodologies began getting increasing public attention
• Each had a different combination of old ideas, new ideas, and transmuted old ideas.
• They **ALL** agreed the following was **MOST** beneficial:
  – Close collaboration between IT and Business Experts
  – Face-to-Face communication (not written)
  – Frequent delivery of new deployable business value
  – Tight, self-organizing teams
  – Develop a “different perspective” to eliminate the **CRISIS** in inevitable requirements churn
The Agile Manifesto

Written in February 2001, at a summit of seventeen independent-minded practitioners of several programming methodologies who found consensus around four core values:

1. Individuals and interactions **over** processes and tools
2. Working software **over** comprehensive documentation
3. Customer collaboration **over** contract negotiation
4. 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.
Why Agile?

In a recent webinar “Getting Agile” presented by Fissure of Minneapolis MN, they presented some interesting survey results:

In a recent survey, almost half of respondents (43.3%) indicated their organizations had recently stopped an IT project:

**Top Reasons:**
1. Business needs had changed (29.9%)
2. The project didn’t deliver as promised (23.4%)
3. Project was no longer a priority (14.4%)
4. Project exceeded the budget (13.2%)
5. Project did not support the business strategy (6.6%)
<table>
<thead>
<tr>
<th>YES</th>
<th>NO</th>
</tr>
</thead>
<tbody>
<tr>
<td>Accelerate Time-to-Market</td>
<td>Lack of Up-front Planning</td>
</tr>
<tr>
<td>Manage Changing Priorities</td>
<td>Loss of Management Control</td>
</tr>
<tr>
<td>Increase Productivity</td>
<td>Lack of Documentation</td>
</tr>
<tr>
<td>Enhance Software Quality</td>
<td>Lack of Predictability</td>
</tr>
<tr>
<td>Improve IT/Business Alignment</td>
<td>Lack of Engineering Discipline</td>
</tr>
<tr>
<td>Improve Project Visibility</td>
<td>Inability to Scale</td>
</tr>
<tr>
<td>Reduce Risk</td>
<td>Projects are Too Big</td>
</tr>
<tr>
<td>Simplify Development Process</td>
<td>Too Much Organizational Change</td>
</tr>
<tr>
<td>Improve Engineering Discipline</td>
<td>Decreased Software Maintainability</td>
</tr>
<tr>
<td>Reduce Cost</td>
<td>Too Much Collaboration</td>
</tr>
<tr>
<td>Enhance Software Maint/Extensibility</td>
<td>Daily Status/Transparency</td>
</tr>
<tr>
<td>Improved Team Morale</td>
<td>Lack of “Big Picture”</td>
</tr>
<tr>
<td>Unrealistic Schedule Expectations</td>
<td>Requirements Volatility &amp; Growth</td>
</tr>
<tr>
<td>----------------------------------</td>
<td>----------------------------------</td>
</tr>
<tr>
<td>• Agile approach to release planning makes it possible to deliver highest priority functionality first, deferring other features to later releases.</td>
<td>• Agile embraces change. Scoping is managed via a backlog of required functionality that is quantified and prioritized.</td>
</tr>
<tr>
<td>• Agile provides early and frequent deliveries of threads of working software for stakeholder feedback, often via two week sprints.</td>
<td>• Agile provides stakeholders visibility into the project at regular intervals where requirements details can be clarified to avoid rework later in the project.</td>
</tr>
<tr>
<td>• Historical productivity from completed sprints is used to make more realistic plans for future sprints.</td>
<td>• If there is requirements growth in the backlog, future sprints can be re-planned.</td>
</tr>
</tbody>
</table>
Purpose of Analysis: Does Agile Scale?

Analyzed a sample of Agile projects from the QSM database containing over 10,000 projects of various development methods.

Research Questions:
• Does Agile truly differ from traditional non-Agile methods?
• Could Agile scale to larger functionalities and/or enterprise settings?
• If so, how could Agile be adopted and implemented most effectively?
• What resources would be needed?
Agile Projects Analysis:

A QUANTITATIVE APPROACH
First Study: Is Staffing-Up Necessary?

- As project size and complexity increases, there is a tendency to staff-up.
- However, using larger teams often negates the beneficial effects of any methodology because they:
  - Introduce communication complexities
  - Inject defects into the system
  - Result in higher costs

What We Did:
- Subdivided the Agile projects into “small” and “large” teams based on the number of Full Time Equivalents
- Compared team size categories in a variety of core metric areas
Scaling Agile May Not Require More Staff - Duration

Small teams can construct the same functionality as larger teams with only a **minimal schedule extension**.
Scaling Agile May Not Require More Staff – Effort/Cost

Size vs. Effort

Large teams can achieve minor schedule compression but at exponentially higher costs.
Scaling Agile May Not Require More Staff - Quality

Large teams can achieve minor schedule compression but at exponentially lower quality.
Time-Effort Tradeoff: Large Teams

- Tagged all the projects with staffing 1 standard deviation above the average.
- How much effort/cost was expended?
- How much longer did it take?

Overall, using more staff did not significantly shorten the schedule, but drastically increased the cost!
Time-Effort Tradeoff: Small Teams

- Tagged all the projects with staffing 1 standard deviation below the average.
- How did using fewer people affect schedule and cost?

Overall, using fewer staff saved money and did not drastically extend the schedule!
Organizational Case Study:

CONSIDERATIONS FOR ADOPTING AGILE
Adopting Agile Will Take Time and Resources

- Organizational support and buy-in is key
  - Included an initial assessment of its application development
  - Integrated tools and methods to support Agile and manage the backlog
  - Training component
- The integrated approach to adopting Agile yielded much better results.
- Adopting Agile, or any new development method, takes time!
Adopting Agile Means Higher Phase Overlap

- The software development lifecycles were modeled.
- The first glaring observation was that agile methods utilize a much higher degree of overlap between High Level Design and Construction Phases (97% versus 30%).
- Reasoning concluded agile methods leverage an iterative approach to release planning and delivery.
Adopting Agile Has a Steep Learning Curve

- When agile methods were first adopted, projects utilizing waterfall delivered **58% faster** and used **74% less effort**
- This equates to about **$550,000** in upfront costs for adopting agile using a normalized labor rate of $100/person hour
- In 2011, after the integrated agile adoption, agile methods achieved **34% faster deliveries** and utilized **27% less effort** than waterfall methods.
- Resulting in a cost savings of **$160,000 per project**

<table>
<thead>
<tr>
<th>Agile Methodology</th>
<th>2010</th>
<th>2011</th>
</tr>
</thead>
<tbody>
<tr>
<td>Duration (months)</td>
<td>15.2</td>
<td>5.1</td>
</tr>
<tr>
<td>Effort (PHRs)</td>
<td>12,900</td>
<td>4,317</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Waterfall Methodology</th>
<th>2010</th>
<th>2011</th>
</tr>
</thead>
<tbody>
<tr>
<td>Duration (months)</td>
<td>9.6</td>
<td>7.8</td>
</tr>
<tr>
<td>Effort (PHRs)</td>
<td>7,411</td>
<td>5,958</td>
</tr>
<tr>
<td>AGILE</td>
<td>Phased-Based</td>
<td></td>
</tr>
<tr>
<td>----------------------------------------------------------------------</td>
<td>------------------------------------------------------------------------</td>
<td></td>
</tr>
<tr>
<td>Learning Driven</td>
<td>Plan Driven</td>
<td></td>
</tr>
<tr>
<td>Continuous Client Communication</td>
<td>Infrequent Client Communication</td>
<td></td>
</tr>
<tr>
<td>Deliver in Short, Business-Focused Releases</td>
<td>Deliver Once in “Big Bang” Fashion Typically 9 – 12 Months</td>
<td></td>
</tr>
<tr>
<td>Typically 2 – 3 Months</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Develop in Sprints of consistent duration and Deliver Working Code</td>
<td>Develop in Distinct Phases with Interim Paper Deliverables</td>
<td></td>
</tr>
<tr>
<td>Develop in End-to-End Functional Slices</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Continuously Integrate Code Throughout (Hourly Builds)</td>
<td>Develop in Layers: Presentation, Persistence, Business, etc.</td>
<td></td>
</tr>
<tr>
<td>Fully-Automated, Continuous Testing at Both Functional and Unit Level</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Low Cost of Change</td>
<td>Integration of Different Layers Occurs at End of Build Phase</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Testing as Separate Phase at End of Project, Typically Emphasizing Functional Level</td>
<td></td>
</tr>
<tr>
<td></td>
<td>High Cost of Change</td>
<td></td>
</tr>
</tbody>
</table>

Source: Getting Agile – April 15
Does Agile Scale for Larger/More Complex Systems?

- Intersection points show when Agile yielded the greatest benefit.
- Agile is not a silver bullet.
  - The decision to use Agile should be situational.
- “Sweet-Spot” will vary by organization and scales as the average functionality increases.
Conclusions

**NOTE:** Before adopting Agile, or any new development method, obtain a quantitative baseline measurement.

- Appreciate and leverage the benefits of using small teams.
- Agile can provide benefits when used in the appropriate application.
- Agile must be scaled and realistically implemented in order to gain the greatest benefits.
  - Understand organizational needs, strengths, and identify areas for improvement.
  - Results can help determine future business decisions.
QUESTIONS?
Helping Clients Realize the Value of Better Software Estimation and Control

Quantitative Software Management, Inc. (QSM) is a family-owned and operated business started in 1978. Founded by Lawrence Putnam, Sr., we continue a proud and passionate legacy as a pioneer and leader in the software and systems estimation field.

Innovation. Integrity. Insight. We are: "The Intelligence Behind Successful Software Projects".

To learn more contact us at 1.800.424.6755, or visit our Web site at www.qsm.com.