Why Does Agile Software Development Not Require the TSP Disciplines?

Yoshihiro Akiyama
Next Process Institute (NPI)

September 18th, 2012
Contents

1. Why the Agility is Needed
2. Review on Agile Method by Waterloo Univ. and ACM
3. Problems in Agile Methods
4. Two Principles for Making the Agility Meaningful
   • Isolation of “process to develop a solution” from “process to seeking a solution”
   • Prediction capability at any time during project
5. TSP principles and practices for improving the agility
6. Summary

Needs for the Agility – Open system Evolution

Closed/Open Architecture
Mult. System & Appl. Integration.
Needs for Agility - Knowledge Work

- “Invisible from outside,” “complex,” and “distributed,”
- Directly related to customer, end-user, and market
- Presented the “goals” with numbers.

"Man-Month is not equivalent to Calendar-Month."
Ref. The Mythical Man-Month, Frederick P. Brooks, 1974

Why “Agile” started

Claimed problems in the current development methodology

- Low Interaction in team
- Working software at the project end
- Low listening to Customer Voice
- Slow Responding to Changes
Challenges of Migrating to Agile Methodologies

- CACM May 2005

<table>
<thead>
<tr>
<th>Traditional</th>
<th>Agile</th>
</tr>
</thead>
<tbody>
<tr>
<td>Functions</td>
<td>High quality, adaptive software can be developed by small teams by testing with rapid feedback and change,</td>
</tr>
<tr>
<td>Control</td>
<td>People centric</td>
</tr>
<tr>
<td>Management style</td>
<td>Leadership-and-collaboration</td>
</tr>
<tr>
<td>Knowledge management</td>
<td>Tacit</td>
</tr>
<tr>
<td>Role Assignment</td>
<td>Self-organizing teams – encourages role interchangeability</td>
</tr>
<tr>
<td>Communication</td>
<td>Informal</td>
</tr>
<tr>
<td>Customer’s role</td>
<td>Critical</td>
</tr>
<tr>
<td>Project cycle</td>
<td>Guided by product features</td>
</tr>
<tr>
<td>Development model</td>
<td>Evolutionary-delivery model</td>
</tr>
<tr>
<td>Technology</td>
<td>Favors object-oriented technology</td>
</tr>
</tbody>
</table>

1. Advantages
   - notable increase in productivity, teamwork, and knowledge sharing

2. Disadvantages
   - more time needed in design discussions/documentation
   - lack of time for thorough testing of the product before deployed
   - difficult to prioritize stories (product owner)
   - the process regarding defects/bugs is not clearly defined

3. Recommendations
   - should have more design stories in their sprints with clear deliverables and the documentation done together
   - invite others for daily standup meetings to show how Agile does and works
Agility Impediments
From “Balancing Agility and Disciplines” by James Over 2012

1. Impediments to Agility in Software
   - Lack of “done” at the end of an iteration
   - Lack of teamwork
   - Lack of good design
   - Lack of tolerating defects

2. Avoiding the Impediments
   - Build high-performance teams
   - Plan all development work
   - Use a measured process
   - Design before you build
   - Make quality the top priority.

REQ defects brake on Agility – from PSP

The design and unit test times are extensively spread respectively
if defect is injected during the requirement phase.
Incorrect Understanding leads Upstream Defect

- longer test time per module leads higher defect density

Assumptions:

Actual/Plan(DLD, Code, UT, IT, ST) =

\[
\begin{cases}
1 & \text{when quality work is done} \\
3,1,2,3,3 & \text{when low quality work is done}
\end{cases}
\]

Life Cycle Increase Ratio estimation with a mixture of quality and low quality work

<table>
<thead>
<tr>
<th>Mixing Ratio</th>
<th>Case1</th>
<th>Case2</th>
<th>Case3</th>
<th>Case4</th>
<th>Case5</th>
</tr>
</thead>
<tbody>
<tr>
<td>Quality work</td>
<td>0.95</td>
<td>0.9</td>
<td>0.7</td>
<td>0.5</td>
<td>0.3</td>
</tr>
<tr>
<td>Low quality work</td>
<td>0.05</td>
<td>0.1</td>
<td>0.3</td>
<td>0.5</td>
<td>0.7</td>
</tr>
<tr>
<td>Increase</td>
<td>1.06</td>
<td>1.12</td>
<td>1.27</td>
<td>1.59</td>
<td>1.82</td>
</tr>
</tbody>
</table>

⇒ 6 – 82% increase in life cycle is expected for the mixture of 5 – 70% of low quality work.
The PSP Conceptual Design and the PROXY

Proxy based planning is the key for small & simple, high quality, understanding, and faster.

LOC = Added & Modified Code LOC

The PSP PROXY Based Planning is Key

Using the part concept as proxy ensures in early phase
- Thinking more clearly on what to build
- Planning with simplicity and reuse
- Avoiding potential low-quality and ill-enlarged code
The TSP Solution PROXY Based Planning

1. Creating/searching a new solution may results in
   - Unlimited time required because of the complexity
   - More defects injected in the upper stream
2. Using the solution proxy concept ensures
   - Thinking more clearly on what to build
   - Avoiding potential defect injected in early phases
   - Communicating clearly with clients and team members
   - Estimating for accuracy with simplicity and reuse
   - Being helped with architecture or framework centered

TSP Balanced Plan Gives Shortest Cycle Time

TSP gives the shortest life cycle time with
- Estimating with process data
- Tracking daily, weekly, to_date, Individual, and team
- Ordering tasks properly
- Launch, relaunch, checkpoint, and weekly meeting to
  - maximize utilizing all team talents and resources
  - ask management support properly

Replanned Balanced Plan
The TSP Launch Discipline Supports the Agility

<table>
<thead>
<tr>
<th>TSP Launch Meeting</th>
<th>M1 Goals by management and customers</th>
<th>M2 Team Goals and Roles</th>
<th>M3 Dev Strategy/Process definition</th>
<th>M4 Top Down plans</th>
<th>M5 Quality Plan</th>
<th>M6 Individual Plan</th>
<th>M7 Risk Planning Proposal/Approval</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Supports for Agility</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Understand all requirements from management and customer</td>
</tr>
<tr>
<td></td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Define the team goals and roles</td>
</tr>
<tr>
<td></td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Establish strategy with solution and multi cycles</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Establish accurate estimate</td>
</tr>
<tr>
<td></td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Establish quality reliable by solution/part</td>
</tr>
<tr>
<td></td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Building balanced plan (shortest life cycle)</td>
</tr>
<tr>
<td></td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Eliminating potential variation</td>
</tr>
<tr>
<td></td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Ensuring management agreement/approval</td>
</tr>
</tbody>
</table>

Conclusions

1. Demanding for the open infrastructure (Internet or TCP/IP) and the knowledge work enabled agility in development,
2. The TSP disciplines ensures the agility for shorter development life cycle because it filters out the “seeking for a solution” complexity during the planning or the launch and the upper stream phases.
3. Also the residual cycle time of a project is “balanced,” i.e., planned for shortest, by fully utilizing the abilities and resources of the team and management.
4. As Agile methods may allow more defects injected during upper stream, the braking is estimated by somewhere 6% to 80% longer than its planned time.
Thank you for your patience.

Contact: Yoshihiro Akiyama
Next Process Institute Ltd.
102-3-3024 Nogawa Miyamae
Kawasaki Kanagawa 216-0001, Japan
Email: y.akiyama@next-process.com
HomePage: http://www.next-process.com
Tel: +81-44-751-1360

TSP Strategic partner
CMU/Software Engineering Institute

Note: The NPI home page will be opened by Oct. 3rd, 2012.

(c) 2012 Next Process Institute 2012/09/18 19