Agile and the Seven Deadly Sins of Project Management

Mike Cohn
February 15, 2011
Mike Cohn - background

Agile coach and trainer

- Founding member and director of Agile Alliance and Scrum Alliance
- Founder of Mountain Goat Software
- Ran my first Scrum project in 1995
- Former VPE in four companies
A cornucopia of agile processes

**Agile Processes**
- Extreme Programming (XP)
- Scrum
- Crystal
- DSDM
- Lean software development
- Unbranded “agile”

**Semi-Agile Processes**
- Feature-Driven Development (FDD)
- Unified Process
- OpenUP
A closer look at Scrum

• No specific engineering practices prescribed
  • But many Scrum teams are adopting much of XP
• 2- to 4-week iterations called “sprints”
• Self-organizing, cross-functional teams
• Uses generative rules to create an agile environment
The Scrum project community

- Programmer
- Programmer
- Programmer
- DBA
- Tester
- Tester
- Analyst
- User Experience Designer

The team

Scrum Master
- Process coach
- Sheepdog for the team

Product Owner
- Defines what to build and in what order

© 2008–2011 Mountain Goat Software®
Scrum

Sprint goal

Return

Cancel

Coupons

Gift wrap

Product backlog

Sprint backlog

Sprint 2-4 weeks

24 hours

Potentially shippable product increment
Seven Sins of Project Management

1. Gluttony
2. Lust
3. Sloth
4. Opaqueness
5. Pride
6. Wastefulness
7. Myopia
Sin #1: Gluttony

• Definition
  • Fixing all dimensions (scope, schedule, resources, and quality) of a project
  • A project management sin of excess

• Experienced as
  • Impossible schedules
  • Death marches

• Leads to
  • Trying to do too much for the resources (time, people) available
  • Cutting quality to meet other goals
Timeboxes help avoid gluttony

- Agile iterations are timeboxed
  - So the schedule of an iteration is fixed
  - But the number of iterations is variable
  - Focus is always on “what can we accomplish next?”
Timeboxed iterations increase predictability

- Over time a team learns how much it can complete per iteration (its “velocity”)

- Knowing this prevents the temptation to overcommit
What expectation of future velocity should this team set?

<table>
<thead>
<tr>
<th>Feature</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Feature A</td>
<td>5</td>
</tr>
<tr>
<td>Feature B</td>
<td>3</td>
</tr>
<tr>
<td>Feature C</td>
<td>5</td>
</tr>
<tr>
<td>Feature F</td>
<td>3</td>
</tr>
<tr>
<td>Feature D</td>
<td>5</td>
</tr>
<tr>
<td>Feature E</td>
<td>5</td>
</tr>
<tr>
<td>Feature G</td>
<td>3</td>
</tr>
<tr>
<td>Feature I</td>
<td>3</td>
</tr>
<tr>
<td>Feature H</td>
<td>5</td>
</tr>
<tr>
<td>Feature J</td>
<td>2</td>
</tr>
<tr>
<td>Feature K</td>
<td>5</td>
</tr>
<tr>
<td>Feature L</td>
<td>3</td>
</tr>
</tbody>
</table>

✓✓

<table>
<thead>
<tr>
<th>Feature</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Feature A</td>
<td>5</td>
</tr>
<tr>
<td>Feature B</td>
<td>3</td>
</tr>
<tr>
<td>Feature C</td>
<td>5</td>
</tr>
<tr>
<td>Feature F</td>
<td>3</td>
</tr>
<tr>
<td>Feature D</td>
<td>5</td>
</tr>
<tr>
<td>Feature E</td>
<td>5</td>
</tr>
<tr>
<td>Feature G</td>
<td>3</td>
</tr>
<tr>
<td>Feature I</td>
<td>3</td>
</tr>
<tr>
<td>Feature H</td>
<td>5</td>
</tr>
<tr>
<td>Feature J</td>
<td>2</td>
</tr>
<tr>
<td>Feature K</td>
<td>5</td>
</tr>
<tr>
<td>Feature L</td>
<td>3</td>
</tr>
</tbody>
</table>

© 2008–2011 Mountain Goat Software®
Sin #2: Lust

- **Definition**
  - Intense or unrestrained craving for features

- **Experienced as**
  - Trying to put too many features into a product during the time allowed
  - Treating all features as “critical”

- **Leads to**
  - Overtime, reduced quality, surprises
Three ways agile deals with lust

1. Developing features in priority order

2. Incremental gratification
“Overtime is a symptom of a serious problem on the project. The XP rule is simple—you can’t work a second week of overtime. For one week, fine, crank and put in some extra hours. If you come in on Monday and say ‘To meet our goals, we’ll have to work late again,’ then you already have a problem that can’t be solved by working more hours.”

~Kent Beck
Old habits die hard

Normal Velocity

Overtime Velocity

Velocity

Iterations
Sin #3: Sloth

- **Definition**
  - Failing to do high quality work at all times

- **Experienced as**
  - Testing quality in at the end
  - Instability during development

- **Leads to**
  - Big delays
  - Unpredictable schedules
Agile practices related to quality

- Simple design
- Automated testing
- Test-driven development
- Continuous integration
- Pair programming
- Refactoring
## Case Study 1

### Cosmodemonic Biotech

<table>
<thead>
<tr>
<th></th>
<th>Waterfall</th>
<th>Scrum</th>
</tr>
</thead>
<tbody>
<tr>
<td>Use Case pages</td>
<td>3,000</td>
<td></td>
</tr>
<tr>
<td>User Stories</td>
<td></td>
<td>1,400</td>
</tr>
<tr>
<td>Calendar months</td>
<td>9</td>
<td>12</td>
</tr>
<tr>
<td>Person months</td>
<td>540</td>
<td>54</td>
</tr>
<tr>
<td>Lines of Java code</td>
<td>58,000</td>
<td>51,000</td>
</tr>
<tr>
<td>Lines of Java code per person-month</td>
<td>120</td>
<td>840</td>
</tr>
</tbody>
</table>
Case Study 2

ePS

Productivity (NCSS / month)

- US average: 270
- Three years prior to introducing agile: 389
- First nine months after starting agile: 1206

NCSS = Non-Comment Source Statement (Java)
ePS

Defects per KNCSS

Three years prior to introducing agile: 10
First nine months after starting agile: 2.9

But wait, there’s more...

• Results achieved without any targeted rewrite of existing (buggy) code
• Many of the post-agile defects continued to be in the old code
• True results would be even better (if we had measured them)
Sin #4: Opaqueness

• Definition
  • Obscuring the progress, quality or other attribute of a project

• Experienced as
  • Not knowing the true state of the project

• Leads to
  • Surprises
  • Poor decisions
Three types of opaqueness

1. How agile addresses Quality opaqueness

- Don’t let bugs accumulate
- Continuous integration
- Features are either “Done” or “Not Done”
  - Avoids the 90% Syndrome
How agile addresses Schedule opaqueness

A release burndown chart

- Story Points
- Iterations

© 2008–2011 Mountain Goat Software®
How agile addresses Scope opaqueness

A confidence interval (28–37 here)
Using a confidence interval to predict what will be delivered

Assume:
There are five sprints left.

We’ll almost certainly get here (5×28)

Based on our median, we’ll finish here (5×33)

This is the most we can realistically expect (5×37)
Sin #5: Pride

• Definition
  • Believing that we know everything to build the product

• Experienced as
  • A lack of stakeholder and user involvement

• Leads to
  • Failure to solicit feedback
  • Failure to learn
Where are opportunities for feedback?

- Sprint goal
- Return
- Cancel
- Coupons
- Gift wrap
- Product backlog

Sprint 2-4 weeks

Potentially shippable product increment

24 hours
Agile requirements

“User stories” facilitate working closely with users & customers

As a user, I want to reserve a hotel room.

As a vacation traveler, I want to see photos of the hotels so that I can choose the best one for me.

As a user, I want the site to be available 99.999% of the time I try to access it.
Sin #6: Wastefulfulness

- Definition
  - Misuse of critical resources
- Experienced as
  - Losses of creativity, motivation, and time
- Leads to
  - Project malaise
  - Delays
  - Doing it the same way (again)
How agile handles waste

1. Timeboxing
2. Daily standups
3. Iteration retrospectives
4. Self-organizing teams
5. Spreading intensity evenly

- **Waterfall**
- **Scrum**

Graph showing comparison between Waterfall and Scrum methods over time (months) with intensity on the y-axis.
Salesforce.com

- Improvement in mean time to release for major releases: +61%
- Increase in features delivered in major releases: +94%
- Increase in features delivered per developer: +38%
- Increase in major release cumulative value: +568%

Source: Scrum Gathering, April 16, 2008
Sin #7: Myopia (Shortsightedness)

• Definition
  • Not seeing beyond your own work

• Experienced as
  • Teams who don’t see the big picture
  • Individuals who work only within their roles

• Leads to
  • Unsuccessful products
  • Delays
Seeing the forest and the trees at the same time

**Release Plan**

<table>
<thead>
<tr>
<th>Sprint 1</th>
<th>Sprint 2</th>
<th>Sprint 3</th>
<th>Sprints 4–7</th>
</tr>
</thead>
</table>

**Iteration Plan**

<table>
<thead>
<tr>
<th>Task</th>
<th>Duration</th>
</tr>
</thead>
<tbody>
<tr>
<td>Task A</td>
<td>8 hours</td>
</tr>
<tr>
<td>Task B</td>
<td>16 hours</td>
</tr>
<tr>
<td>Task C</td>
<td>5 hours</td>
</tr>
<tr>
<td>Task D</td>
<td>8 hours</td>
</tr>
</tbody>
</table>
Cross-functional team

• All disciplines necessary to go from idea to implementation
• Improves creativity and ownership
• Whole team commitment

But does that make everyone a generalist?
Is everyone a generalist?
Is everyone a generalist?

Screw that; where's lunch?
Is everyone a generalist?

Key:
- Order-taker
- Floater
- Sandwich-maker

Meconi’s:
- Order-taker
- Floater
- Sandwich-maker

Bruno’s:
- Order-taker
- Sandwich-maker

Ferentino’s:
- Order-taker
- Floater
- Sandwich-maker

New that’s lunch?
Additional resources

Scrum Information
www.MountainGoatSoftware.com/scrum
www.MountainGoatSoftware.com/presentations
www.ScrumAlliance.org

Training
www.MountainGoatSoftware.com/training

Books
*Agile Software Development with Scrum*, Ken Schwaber
*Succeeding with Agile*, Mike Cohn
Mike Cohn
mike@mountaingoatsoftware.com
www.mountaingoatsoftware.com
twitter: mikewcohn
(888) 61-AGILE