An Introduction to Agile/Scrum

Jeff Pulcini
Cyber Security Workshop
October 19, 2019

Co sponsored by The Department of Defense, Indiana University of Pennsylvania and Westmoreland Community College
CREDITS

Base presentation was done by:
Mike Cohn (circa 2007)
mike@mountaingoatsoftware.com
www.mountaingoatsoftware.com
(720) 890-6110 (office)

Other portions are credited as appropriate.

Additional material by Jeff Pulcini
Goals and Ground Rules

• Introduce the concepts of Agile/Scrum
• Contrast with another methodology, i.e. waterfall
• Why we should care?

Get the big picture, concepts, and keywords

I’m not going to be rigorous in history or definitions
I’m likely to make extreme statements to make the point.

We will talk software development, but Agile can be used for anything.
Problem Statement

What we have been doing is not working.

We are faced with…

• Producing more and better output with shrinking budgets
• Maximize the actual and perceived value we deliver
• Keep up with the pace of change in the market

How did we get here?

Adapted from a presentation by Evan Campbell, Slide 32
http://agilecommons.org/files/4e9f355ad9/Agile_Fncl_Impact_Slides.pdf
Project Management - (a VERY brief overview)

- People have ALWAYS organized work

- Management of process and projects became more codified as we developed “Scientific Management” principles in the early 1900’s

- Henry Gantt developed the Gantt chart in 1903

- In the 1950’s the PERT chart (1958) (program evaluation and review technique) and critical path management were dominant

- In the 60’s more formalized “waterfall” techniques were used

- In 1969, the Project Management Institute was incorporated

- in 1985 the DOD issued DOD-STD-2167A that was their standards for working with software development contractors
Waterfall Project Management Methodology

• A step-wise approach to product delivery

• It is a or relay race of analysis, requirements definition, design, Implementation (code and test) and then delivery and maintenance.

• Product delivery is a “big bang”/“all or nothing”
Downside of This Approach

- **Focus** is primarily **on process**, not people.
- **Long development cycles**
- **Unresponsive** to changing markets (and thus users needs)
- **Formal communications**, Lots of words on paper
- Each step fosters a **them vs. us** mentality
- **Plan based** - We measure if we are on track, not what we have done
- **You know the least when you start**
Cone of Uncertainty

Example: Boeing’s 787 Dreamliner was 5x more expensive and took twice as long to deliver than original estimates.

As described by Steve McConnel, Software Estimation: Demystifying the Black Art
We’re losing the relay race

“The... ‘relay race’ approach to product development...may conflict with the goals of maximum speed and flexibility. Instead a holistic or ‘rugby’ approach—where a team tries to go the distance as a unit, passing the ball back and forth—may better serve today’s competitive requirements.”

Waterfall vs. Agile

THE WATERFALL PROCESS

‘This project has got so big, I’m not sure I’ll be able to deliver it!’

THE AGILE PROCESS

‘It’s so much better delivering this project in bite-sized sections’

From the Slide set Lecture 1 - What is Agile Dr. Imran Ghani, As transmitted September 2019
- Why the Neanderthals became extinct -

Yes, but Og assures me that this will improve our efficiency and keep us ahead of those Cro-magnons in the valley.

I don't know, it seemed easier when we just went hunting.
Agile – some definitions, changing our mindset
“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 **Process 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 on the items on the right, we value the items on the left more.”

Source: www.agilemanifesto.org
1. **Our highest priority is to satisfy the customer through early and continuous delivery of valuable software.**

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

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

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

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

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

7. **Working software is the primary measure of progress.**

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

9. Continuous attention to **technical excellence and good design enhances agility.**

10. **Simplicity**--the art of **maximizing the amount of work not done**--is essential.

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

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

Source: http://agilemanifesto.org/principles.html
Sequential vs. overlapping development

Rather than doing all of one thing at a time...

...Scrum teams do a little of everything all the time

Scrum – The Big Picture
The Big Picture

- **Daily Scrum Meeting**
- **Product Backlog**
- **Sprint Backlog**
- **2-4 Weeks**
- **24 Hours**
- **Potentially Shippable Product Increment**

Image available at www.mountaingoatsoftware.com/scrum
Characteristics

- Requirements are captured as items in a list of “product backlog”
- Product progresses in a series of short “sprints”
- Self-organizing teams
- No specific engineering practices prescribed
- Uses generative rules to create an agile environment for delivering projects
Scrum Details
Scrum Framework

Team Roles
• Product owner
• ScrumMaster
• Team

Ceremonies
• Sprint planning
• Sprint review
• Sprint retrospective
• Daily scrum meeting

Now they are called EVENTS!

Artifacts
• Product backlog
• Sprint backlog
• Burndown charts
Scrum Framework

Team Roles
- Product owner
- ScrumMaster
- Team

Events
- Sprint planning
- Sprint review
- Sprint retrospective
- Daily scrum meeting

Artifacts
- Product backlog
- Sprint backlog
- Burndown charts
* Define the features of the product
* Decide on release date and content
* Be responsible for the profitability of the product (ROI)
* Prioritize features according to market value
* Adjust features and priority every iteration, as needed
* Accept or reject work results
The Scrum Master

- Represents management to the project
- Responsible for enacting Scrum values and practices
- Removes impediments
- Ensure that the team is fully functional and productive
- Enable close cooperation across all roles and functions
- Shield the team from external interferences
- The ScrumMaster differs from a Project Manager in that he does not exercise command and control
The Team

- Typically 5-9 people
- Cross-functional:
  - Programmers, testers, user experience designers, etc.
- Members should be full-time (no multi-tasking!)
  - May be exceptions (e.g., database administrator)
- Teams are self-organizing
  - Ideally, no titles but rarely a possibility
- Membership should change only between sprints
The Myth of Multitasking

from a 1990’s Harvard Study by Steven C.Wheelwright and Kim B.Clark
Scrum framework

Team Roles
- Product owner
- ScrumMaster
- Team

Events
- Sprint planning
- Sprint review
- Sprint retrospective
- Daily scrum meeting

Artifacts
- Product backlog
- Sprint backlog
- Burndown charts
Sprint Planning

**Sprint planning meeting**

**Sprint prioritization**
- Analyze and evaluate product backlog
- Select sprint goal

**Sprint planning**
- Decide how to achieve sprint goal (design)
- Create sprint backlog (tasks) from product backlog items (user stories / features)
- Estimate sprint backlog in hours

**Team capacity**

**Product backlog**

**Business conditions**

**Current product**

**Technology**

**Sprint goal**

**Sprint backlog**
The sprint goal

• A short statement of what the work will be focused on during the sprint

Database Application

Make the application run on SQL Server in addition to Oracle.

Life Sciences

Support features necessary for population genetics studies.

Financial services

Support more technical indicators than company ABC with real-time, streaming data.
Sprint Planning

• Team selects User Stories from the product backlog they can commit to completing
• Sprint backlog is created
  • Tasks are identified and each is estimated (1-16 hours)
  • Done Collaboratively, not alone by the Scrum Master/Product Owner
• High-level design is considered

User Story

As a vacation planner, I want to see photos of the hotels.
(The who, what and why of what is to be done)

Code the middle tier (8 hours)
Code the user interface (4)
Write test fixtures (4)
Code the foo class (6)
Update performance tests (4)
Sprints

• Scrum projects make progress in a series of “sprints”

• Typical duration is 2–4 weeks or a calendar month at most

• A constant duration leads to a better rhythm

• Product is designed, coded, and tested during the sprint
No changes during a sprint

- Plan sprint durations around how long you can commit to keeping change out of the sprint
The Daily Scrum

- Parameters
  - Daily
  - 15-minutes
  - Stand-up
- Not for problem solving
  - Whole world is invited
  - Only team members, ScrumMaster, product owner, can talk
- Helps avoid other unnecessary meetings
Everyone Answers Three Questions

1. What did you do yesterday?
2. What will you do today?
3. Is anything in your way?

- These are *not* status for the ScrumMaster
- They are commitments in front of peers
The Sprint Review

- Invite the world
- Whole team participates
- Informal
  - 2-hour prep time rule
- No slides
- Team presents what it accomplished during the sprint
- Typically takes the form of a demo of new features or underlying architecture
Sprint Retrospective

• Periodically take a look at what is and is not working
• Typically 15–30 minutes
• Done after every sprint
• Whole team participates
  • ScrumMaster
  • Product owner
  • Team
  • Possibly customers and others
Start / Stop / Continue

• Whole team gathers and discusses what they’d like to:

  - Start doing
  - Stop doing
  - Continue doing

This is just one of many ways to do a sprint retrospective.
**Scrum Framework**

**Team Roles**
- Product owner
- ScrumMaster
- Team

**Events**
- Sprint planning
- Sprint review
- Sprint retrospective
- Daily scrum meeting

**Artifacts**
- Product backlog
- Sprint backlog
- Burndown charts
Product Backlog

- The requirements
- A list of all desired work on the project
- Ideally expressed such that each item has value to the users or customers of the product
- Prioritized by the product owner
- Reprioritized at the start of each sprint
## A Sample Product Backlog

<table>
<thead>
<tr>
<th>Backlog item</th>
<th>Estimate</th>
</tr>
</thead>
<tbody>
<tr>
<td>Allow a guest to make a reservation</td>
<td>3</td>
</tr>
<tr>
<td>As a guest, I want to cancel a reservation.</td>
<td>5</td>
</tr>
<tr>
<td>As a guest, I want to change the dates of a reservation.</td>
<td>3</td>
</tr>
<tr>
<td>As a hotel employee, I can run RevPAR reports (revenue-per-available-room)</td>
<td>8</td>
</tr>
<tr>
<td>Improve exception handling</td>
<td>8</td>
</tr>
<tr>
<td>...</td>
<td>30</td>
</tr>
<tr>
<td>...</td>
<td>50</td>
</tr>
</tbody>
</table>
Sprint backlog

- One or more backlog items make up the backlog.

- The work is decomposed into tasks and hours.

- If work is unclear, define a sprint backlog item with a larger amount of time and break it down later.

- Individuals sign up for work of their own choosing - work is never assigned.

- Estimated work remaining is updated daily as more becomes known.

- Any team member can add, delete or change the sprint backlog.
# A Sprint Backlog

<table>
<thead>
<tr>
<th>Tasks</th>
<th>Mon</th>
<th>Tues</th>
<th>Wed</th>
<th>Thur</th>
<th>Fri</th>
</tr>
</thead>
<tbody>
<tr>
<td>Code the user interface</td>
<td>8</td>
<td>4</td>
<td>8</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Code the middle tier</td>
<td>16</td>
<td>12</td>
<td>10</td>
<td>4</td>
<td></td>
</tr>
<tr>
<td>Test the middle tier</td>
<td>8</td>
<td>16</td>
<td>16</td>
<td>11</td>
<td>8</td>
</tr>
<tr>
<td>Write online help</td>
<td>12</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Write the foo class</td>
<td>8</td>
<td>8</td>
<td>8</td>
<td>8</td>
<td>8</td>
</tr>
<tr>
<td>Add error logging</td>
<td></td>
<td></td>
<td>8</td>
<td>4</td>
<td></td>
</tr>
</tbody>
</table>
# Burndown Example

<table>
<thead>
<tr>
<th>Tasks</th>
<th>Mon</th>
<th>Tues</th>
<th>Wed</th>
<th>Thur</th>
<th>Fri</th>
</tr>
</thead>
<tbody>
<tr>
<td>Code the user interface</td>
<td>8</td>
<td>4</td>
<td>8</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Code the middle tier</td>
<td>16</td>
<td>12</td>
<td>10</td>
<td>7</td>
<td></td>
</tr>
<tr>
<td>Test the middle tier</td>
<td>8</td>
<td>16</td>
<td>16</td>
<td>11</td>
<td>8</td>
</tr>
<tr>
<td>Write online help</td>
<td>12</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Hours</th>
</tr>
</thead>
<tbody>
<tr>
<td>44</td>
</tr>
<tr>
<td>32</td>
</tr>
<tr>
<td>34</td>
</tr>
<tr>
<td>18</td>
</tr>
<tr>
<td>8</td>
</tr>
</tbody>
</table>

Ideal
Why do this?
Reasons for Adopting Agile

Success Rates

![Project Success Rates: Agile vs Waterfall](https://vitalitychicago.com/blog/agile-projects-are-more-successful-traditional-projects/)

April 1, 2018

<table>
<thead>
<tr>
<th>METHOD</th>
<th>SUCCESSFUL</th>
<th>CHALLENGED</th>
<th>FAILED</th>
</tr>
</thead>
<tbody>
<tr>
<td>AGILE</td>
<td>42%</td>
<td>50%</td>
<td>8%</td>
</tr>
<tr>
<td>WATERFALL</td>
<td>26%</td>
<td>53%</td>
<td>21%</td>
</tr>
</tbody>
</table>

SOURCE: STANDISH GROUP CHAOS STUDIES 2013-2017

https://vitalitychicago.com/blog/agile-projects-are-more-successful-traditional-projects/
Reasons for Adopting Agile
Success Rates

PROJECT SUCCESS RATES BY PROJECT SIZE
AGILE VS WATERFALL
FOR LARGE PROJECTS, AGILE APPROACHES ARE 2X MORE LIKELY TO SUCCEED

Source: Standish Group, Chaos Studies 2013-2017
WWW.VITALITYCHICAGO.COM

18% AGILE 9% WATERFALL
31% AGILE 19% WATERFALL
59% AGILE 56% WATERFALL

https://vitalitychicago.com/blog/agile-projects-are-more-successful-traditional-projects/
Reasons for Adopting Agile
Shifts in Industry Attitudes

Changes from 2018 to 2019

• Less about increasing productivity (51% compared to 55%)
• More about improving team morale (34% compared to 28%)
• Less about reducing project risk (28% compared to 37%)
• More about reducing project costs (41% compared to 24%)

Benefits of Adopting Agile

- Ability to manage changing priorities: 69%
- Project visibility: 65%
- Business/IT alignment: 64%
- Team morale: 64%
- Delivery speed/time to market: 63%
- Increased team productivity: 61%
- Project predictability: 52%
- Project risk reduction: 50%
- Software quality: 47%
- Engineering discipline: 42%
- Managing distributed teams: 39%
- Software maintainability: 34%
- Project cost reduction: 28%

Bottom Line

• Agile projects are 2X more likely to succeed

• Agile projects are 1/3 less likely to fail than waterfall projects

The Standish Group has conducted surveys of IT project success and failure rates every 2 years since 1994.

Source: Standish Group Chaos Study 2018
Scrum has been used for:

- Commercial software
- In-house development
- Contract development
- Fixed-price projects
- Financial applications
- ISO 9001-certified applications
- Embedded systems
- 24x7 systems with 99.999% uptime requirements
- The Joint Strike Fighter
- Video game development
- FDA-approved, life-critical systems
- Satellite-control software
- Websites
- Handheld software
- Mobile phones
- Network switching applications
- ISV applications
- Some of the largest applications in use