Learning is not compulsory... neither is survival.
- W. Edwards Deming
The Agile Dysfunction Spiral

Coaching to help improve sprint planning

Detailed sprint planning

Sprint planning takes longer

We need to get better at planning sprints

Formal sprint commitment

Culture of Blame creeps back in...

Unfinished work

Management Applies pressure

Work the sprint

Team works overtime

Are we really just after better sprints?
The Agile Dysfunction Spiral

- Coaching to help improve sprint planning
- Detailed sprint planning
- Sprint planning takes longer

“That’s my life on a slide”

- Culture of Blame creeps back in...
- Management Applies pressure
- Unfinished work
- Work the sprint
- Team works overtime

Are we really just after better sprints?
Which is why we see…

(Actual Google searches, BTW)
Minimally Viable Agile
What is it?

Borrowing from the Product Development concept of *Minimum Viable Product*.

"The *minimum viable product* is that version of a new product which allows a team to collect the maximum amount of validated learning about customers with the least effort."
Minimally Viable Agile
What is it?

Minimally Viable Agile is a search for Agile practices and ceremonies, informed by Lean and Agile theory, that produces the maximum amount of customer value, with the least amount of effort.

Minimally Viable Agile is a mindset where decentralized Agile innovation is cultivated.

Or (Just Enough practices and ceremonies to be effective).
Many early Agile attempts simply installed practices (recipes).

**Warning!**
Following recipes leads to Cargo Cult Agile
A DAY OF MOBBING
(whole team programming)

Hour 1 - Skills Learning Session

-- Aaron Griffith, Woody Zuill, Llewellyn Falco
Mob Programming: An original MVA?

**Reduces or Eliminates:**
1. Standup or other alignment meetings
2. Detailed planning-only sessions
3. The need for estimates
4. Context Switching
5. Burn-down charts for performance tracking
6. Code Reviews

**Increases:**
1. Quality
2. Collaboration
3. Productivity
4. Satisfaction
Mob Programming: Why does it work?

Sample timeline of a user story or requirement.

- Customer Request
- Prioritized in Product Backlog
- Backlog Grooming
- Planning Into a Sprint
- Story Completed

Time:
- 2 weeks = wait time (tacit knowledge decay)
- 4 - 10 weeks = touch time
- 2 weeks
Minimally Viable Agile Canvas

**Focus**
- Work Visualizations
- WIP limits / smaller batches
- Cadence / Time-boxes

**Learning**
- Retrospectives / Improvement Kata
- Acceptance Test Driven Development
- Continuous Integration/Delivery

**Simplicity**
- Emergent Architecture
- Art of the work not done

**Customer Value**
- Definition of Done
- Progress measured in working software

**Respect for People**
- Sustainable pace
- Purpose, Autonomy, Mastery
Visceral Reaction Disclaimer

We might cover some “controversial” topics.
Minimum Viable Agile

Let’s take a closer look at some popular Agile practices.

1. Story Point based Velocity
2. Burn Down charts

This is my journey, your mileage may vary!
Story Points based Velocity
Story Point Predictability?

![Graph showing cycle time versus estimate with expected and actual distributions.]

- **Expected Distribution**
- **Actual Distribution**

Courtesy of Bennet Vallet
Director of Development
Siemens Health Services
Minimalist Planning

Option #1
Hours-long traditional sprint planning meetings spent playing planning poker and a team arguing about the difference between 2’s and 3’s.

Option #2
Short story slicing meeting where the focus is on understanding and slicing large stories.

“Do we understand this story? If not, let’s keep slicing.”

Right-sizing the work

1  2  3  5  8  13  21  40  100
Minimalist Planning

<table>
<thead>
<tr>
<th>Input Q</th>
<th>Design</th>
<th>Develop</th>
<th>Test</th>
<th>Done</th>
</tr>
</thead>
<tbody>
<tr>
<td>5</td>
<td>4</td>
<td>2</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>6</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>7</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>8</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>9</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>10</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Throughput
5 stories per week

Cycle Time

14 days from this point

Probabilistic Forecasting is preferred!

For Example:
- 85% certain one card delivered <20 days
- 75% certain 50 stories delivered in 15 weeks
Burn Down Charts

Why do we assume this is the wrong place for this team to be?
Nature of the Work

Manufacturing = repeatable processes
Product Development = one time processes

One-time processes are non-deterministic in nature
Nature of the Work

**Deterministic Approach**

- Knowledge
- Project Start
- Planned Activity
- "Complete"

Project Managers in the middle, pulling out their hair

**Non-deterministic Reality**

- Knowledge
- Better Knowledge
- Project Start
- ?
- "Complete"

What Customer Really Wants
Burn Down Charts

Even if this is a problem, what exactly do we do now?

It’s not actionable!
Burn Down Charts (refactored)

Actionable Information!
Burn Down Charts (refactored)

Actionable Information!

(from ActionableAgile.com)
Minimally Viable Agile

While there is value in the items on the right, we value the items on the left more.

Flow-based metrics and visualizations over burn down charts

Story slicing skills over estimation schemes

( . . . )
Applying Minimally Viable Agile

We can apply MVA thinking in two forms:

Training and education

How we transition

“We’re going to stop creating burn down charts and stop using velocity.”

“Are you insane?”
Improvement Kata

Steps: PDCA

Current Condition

Next Target Condition

Our Vision

Awesome
## Improvement Kata

<table>
<thead>
<tr>
<th>Improvement Theme</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Now / Current Condition</strong></td>
</tr>
<tr>
<td>Definition of Awesome (vision)</td>
</tr>
</tbody>
</table>

--Jimmy Janien, Crisp Blog
Improvement Kata

**Now/Problem**

Releases are chaotic and cost too much in multiple ways. (time, stress, $, customer impact)

**Next Target**

Establish daily integration environment for faster feedback.

1. Merge when the feature is done.

**Definition of Awesome**

Ability to release value as quickly and frequently as needed, in a sustainable manner.

**First Steps**

<table>
<thead>
<tr>
<th>Plan</th>
<th>Do</th>
</tr>
</thead>
<tbody>
<tr>
<td>Check</td>
<td>Act</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Issues</th>
</tr>
</thead>
<tbody>
<tr>
<td>Code complete</td>
</tr>
<tr>
<td>UAT complete</td>
</tr>
<tr>
<td>QA complete</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>User Story</th>
</tr>
</thead>
<tbody>
<tr>
<td>Merge when the feature is done.</td>
</tr>
</tbody>
</table>
Improvement Kata

Plan:

1. Do:
   - Obstacle: Yes DB deploy
     Step: Automate
   - Obstacle: No DB
     Step: Automate
   - Obstacle: Index re-calc
     Issue

2. Do:
   - Obstacle: NS Build out of date
     Step: Automate
   - Obstacle: Testing (Zeenyx)
     Run(s)
     Measure: Does it work? (Yes)
     Expect: None against Zeenyx

3. Do:
   - Obstacle: Shared DB
     Step: Create Service Stub
   - Obstacle: Service not available in trunk

Check:

- Obstacle: Deployment to CI environment
  Step: Automate
  Measure: Dependency on code
  Expect: One click deploy

- Obstacle: Incomplete knowledge across teams, team process
  Step: Branch Best Practice Forum
  Measure: 10 cards done, but not merged
  Expect: More frequent merges to trunk.
Improvement Kata

Merge Day

Continuous Integration

<table>
<thead>
<tr>
<th>Now/Problem</th>
<th>Releases</th>
</tr>
</thead>
<tbody>
<tr>
<td>Releases are chaotic and cost too much in multiple ways (time, stress, &amp; customer impact)</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Next Target Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td>Establish daily integration environment for faster feedback</td>
</tr>
<tr>
<td>Merge, when the req is done</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Definition of Awesome</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ability to release value as quickly and frequently as needed, in a sustainable manner</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>First Steps</th>
</tr>
</thead>
<tbody>
<tr>
<td>Plan</td>
</tr>
<tr>
<td>Check</td>
</tr>
</tbody>
</table>
Minimally Viable Agile Summary

Understand the Why behind Lean and Agile, as your inoculation against Cargo Cult Agile.

Decentralize the Agile Innovation.

Focus on Awesome, and deliberately work towards that vision, and the process cruft will fall away.

Minimally Viable Agile Canvas:

Focus | Learning | Simplicity

Respect for People | Customer Value
Learning is not compulsory... neither is survival.
- W. Edwards Deming
References


*Toyota Kata*, Mike Rother

*Drive*, Daniel Pink

*Lean Startup*, Eric Reis

*Kanban at Scale – A Siemens Success Story*, Bennet Vallet

*CFD generated from ActionableAgile.com*