Scrum Under a Waterfall

Dave Harrison, Columbia Sportswear
1. Align work to meet business goals and keep your promises
2. Eliminate unused features by failing with style
3. Are you really doing Agile? – the Tolstoy test
4. Sneaky guerilla tactics
5. More coding, less management - 2 hrs a week
6. Advanced capacity planning
7. (if time) the Harry Potter Hat of Excellence, and a WFH policy that might actually WORK
Just curious...

- Used agile project management in TFS 2012 or Team Foundation Service?
- Visual Studio 2012
- Automated stress testing?
- Find cloned code
- Code reviews
- Visualization of code dependencies?
- Windows Azure to spin up VM’s?
Are my teams reaching their true potential?
What is in our way?
The Tolstoy Test
“All happy families are alike; all unhappy families are unhappy in their own way.”

- I know who the product owner is.
- My backlog is prioritized by the business owner based on business value.
- My team does work in 1-3 week sprints.
- At the end of each sprint, our features are unit tested and functionally tested.
- My team estimates their tasks and wholeheartedly commits to doing them.
- Once committed, nothing disrupts the work of the team.
- I am transparent and accountable, so I send out honest retrospectives and the sprint plan to EVERYONE.
There is no one definition of Agile.
“All successful teams look alike?”

- Autonomy
- Mastery
- Purpose
Whether your team is pure “agile” is less important than if they feel engaged, productive, and have the tools and training they need to get their work done.

“We'd send out a list of three things each Monday that we wanted to accomplish that week. We'd follow up on Friday with what happened to those three things - what worked and what didn't.” — Scott Hanselman
The Symptoms of a Failed Project

- Lack of buy-in from the project team.
- Top down management style, no delegation, no rewards for initiative.
- Management isolated from daily realities.
- No incremental learning or honest mid-project retrospectives.
- Fixed, inflexible deliverables. Schedule and budget drawn up from thin air.
- No way of cutting off costs or evaluating progress.
- Excessively long meetings and reams of meaningless documentation.
The iron triangle ain’t so iron.

- What “Failure” or “Success” means will differ by the company culture and the constraints and resources of the project.
what I learned from doing it wrong...

Your team deserves isolation in sprint cycles, the business deserves accountability.

“IT makes a terrible servant and a worse master.”
developers

Mobile and hybrid

I can work in my bubble of love

I can get targeted feedback with one click
analysts and testers

Automated and code coverage testing rocks

Bugs get knocked down quickly

Awesome dashboards rule my world
managers and business owners

We’ll know right away if there’s a problem making our commitments

I’ll know if my team is overtasked or underutilized

We are more metrics driven
The business (your clients and customers) define what we do and in what order; we decide how that work should be done.

what I learned from doing it wrong...
Different perspectives

Developers
- Can I make my goals?
- Are the tools helping or hurting me?
- How can I get customer feedback and peer reviews done quickly?

Managers
- How much work is remaining?
- Do we need more resources?
- Are we off on our estimates?

Business analysts
- What is blocking progress?
- Are we on track?
- How is testing going?
One panel to rule them all

- Bug and backlog entry
- Key metrics
- Burndown
- Main activities
Different things to different people
What’s in it for me?

You can pull data into Excel and display pie charts for your customers.

You can see what is being done on a project across multiple teams.

You’ll know in real time if there’s a problem.
Transparency (and honesty) is a must in all relationships. So, the more dashboarding and visibility you provide your business partners and clients the better. (and it never hurts to say you’re sorry.)
How => What

Delivering Value
Metrics Driven
Business Partners
Value Driven Planning

Eliminating Waste
Self managed
Accountable
DevOps
Flow

Transparency
Show technical debt
Admits mistakes
Keeps their word
what I learned from doing it wrong...

Technical specs ain’t evil.*

*unless you overdo it.
An Agile calendar:

- **Jan:** Create and maintain a backlog to hold your technical debt, visible to EVERYBODY. (tfs.visualstudio.com, kanbanflow, Excel) Start a wiki and a blog.
- **Feb:** Begin daily standups, 4 days a week. 15 minutes or less.
- **March:** Begin planning sessions (2 hrs every 2 weeks).
- **April:** Transparency - begin spamming your retrospectives
- **May-July:** Begin folding in business partners, BA’s, testers
- **August:** Begin demonstration sessions
- **October:** Stop preassigning tasks (which creates siloes)
- **December:** Enforce coding standards, peer reviews – definition of “done”
# A week in Agile:

| Mon 9-11 a.m. Sprint Planning | 1. Retrospective (what worked, what didn’t)  
2. Mark work complete in TFS and copy undone items forward  
3. Time off entered into capacity planning  
4. Next batch of prioritized work in backlog added to current sprint and committed to by team  
5. Writeup sent of retrospective and planning (including burndown) to business partners |
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Tues 9:00-9:10</td>
<td>Daily Standup (every day developers mark off progress on board)</td>
</tr>
<tr>
<td>Weds 9:00-9:10</td>
<td>Daily Standup</td>
</tr>
<tr>
<td>Thurs 9:00-9:10</td>
<td>Daily Standup</td>
</tr>
<tr>
<td>Fri 9-10 a.m. Demos</td>
<td>5-10 minutes per developer. New discoveries should be followed with a (short) blog article.</td>
</tr>
</tbody>
</table>
If I knew then... 

Fail faster. 

You have to build your own Agile. 

Tech specs = safety. 

Your “team” is more than developers. 

Demos are fun. 

Bugs aren’t evil.
the harry potter hat of excellence
Can you revise your code?
- Source Control
- Reversion/Branching (TFS)
- Documentation (inline, blog/wiki, SharePoint)

Can you refactor your code without fear?
- Class Design / UML Layers
- Naming Conventions
- TDD
- Class Design / UML Layers

Are bugs VERIFIABLY fixable?
- Automated Unit Tests
  - Static Analysis (VS2010)
  - Dynamic Analysis (ReSharper)
- Design Review
- Peer Review

Can you deploy as easily as you can build?
- Continuous Integration
- Continuous Deployment w/rollbacks

Is there a WFH policy that works?

Many companies undervalue training

Not everybody enjoys or learns from conferences

WFH days alone won’t deliver the company a measurable ROI (i.e. the next gmail)
Bonuses – “work harder and we’ll pay you more!” – deincentive and demean (we are professionals and work for pride)
Work without measurable outcomes is discouraging and aimless

Demos are crowding sprint planning and are too infrequent

Asking for WFH days -> supervisor responds is clumsy and manual

• You take 1 day every week as 20% time. Schedule it in Outlook as OOTO, don’t ask for permission. Work on anything you want, where you want, on whatever schedule you like.

• We have a special demo session for 90-120 minutes every Wednesday morning. (not a part of sprint planning anymore)

• It must provide the company value and you are accountable for results.
  • Meaning, I want to see a demo and a blog article.