Yes, Hardware Can Be Agile!

Boston SPIN, 15 Mar 2016

Nancy Van Schooenderwoert

http://www.leanagilepartners.com/
NancyV@LeanAgilePartners.com

© 2015-16 Lean-Agile Partners Inc. All rights reserved.

Nancy V’ s Background

- Electrical Engineering and Software Engineering for embedded systems
- 15 years safety-critical systems development experience
- Since 2002: Agile coaching of teams and managers in regulated industries
- Industries: Aerospace, Medical Devices, Sonar Weaponry, Scientific Instruments, Industrial Controls, Financial Services
- Author of papers and articles on Lean and Agile methods for engineering work
- Frequent presenter at Agile-related conferences worldwide
- Past president of Agile New England, current ANE board member
Topics

- Hardware roots of “Agile”
- Agile Engineering: examples
- Team composition mirrors workflows
- Want predictability? Or fast learning?

In the beginning…

- 1977 Ward Cunningham testing early microprocessors – forerunner of TDD (Test-driven development)
- 1980 Kent began working with Ward at Tektronix
- 1999 ‘Extreme Programming Explained’ by Kent Beck

I came to a conclusion that you had to get the machines to tell you what they were doing. And that if you wanted to trust what you got the machines to tell you, that part had to be very simple.

Topics

- Hardware roots of “Agile”
- Agile Engineering: examples
- Team composition mirrors workflows
- Want predictability? Or fast learning?

FPGA example

- Field Programmable Gate Array

FPGA chip on ckt board
Inside the chip
Inside each circuit
FPGA circuits ‘wired’ by s/w

- Separate circuits on the wafer are connected by statements in HDL, a type of s/w

```
const logic [1:0] N_ENABLE = 1;
const logic [1:0] S_ENABLE = 2;
// SETUP -> ENABLE
always @(negedge rst_n or posedge clk) begin
  if (rst_n == 0) begin
    apb_st <= 0;
pdata <= 0;
  end
  else begin
    case (apb_st)
      SETUP : Begin
        // clear the pdata
        pdata <= 0;
    endcase
  end
end
```

- Changes to the FPGA can take a few minutes to a few hours

- FPGA Reprogrammed - new bitfile downloaded in seconds…
- Bitfile is rebuilt: this changes the circuit connections, takes time
- Result: h/w can do the work faster than s/w – by orders of magnitude
Writing a unit of HDL code…

**Before Agile**

- START
- Write HDL
- Download bitfile to FPGA circuits
- Check circuit behavior manually
- OK?
  - Y: Done with this Unit test
  - N: Keep testing

**After Agile (TDD)**

- START
- Write HDL unit test that fails
- Write HDL (tiny)
- Download bitfile to FPGA circuits
  - Y: OK?
  - N: Keep testing
- Check circuit behavior manually
  - OK?
    - Y: Done with this Unit test
    - N: Keep testing

Red lines indicate "high traffic" flows.

HDL testing…

**Before Agile**

- Write the HDL code
  - (optional) simulate the HDL
- Deploy to the FPGA circuitry
- Check the circuitry behavior manually
- Very time-consuming

**Result**: have code

**After Agile (TDD)**

- Write the HDL code
  - Write a unit test in HDL
  - Ensure unit test fails
  - Write HDL code
  - Ensure unit test passes
- Deploy to the FPGA circuitry
- Check the circuitry behavior manually
- Much faster

**Result**: have code and tests

Using SVUnit TDD framework

Lean-Agile Partners
Agile Car example

- Wikispeed 100 mpg car built by volunteers
- Agile ‘software’ practices used:
  - Pairing
  - Swarming
  - TDD (Test-driven Development)

Source: www.wikispeed.com

Modularity is key

- Modularity → dependency management
- Car made of 8 modules
- New release weekly

Source: www.wikispeed.com
Example: Grain Monitor System

- Measures protein, oil in corn, wheat, etc. in seconds, in the field
- Evaluated for medical application
- New science, new CPU, new OS port, new NIR sensor, new algorithm…
- Agile team delivered 1st field units in 6 months
- In 3 years – 60+ s/w iterations,
  - approx. 9 electronic iterations
  - approx. 5 mechanical iterations
  - 51 s/w defects post-unit-tests, 3 yr total

Spectrometer system

Iterations work differently

- Less frequent iterations for hard-to-change items
- Aim for working hardware at each iteration boundary
- Misconception: To be Agile, h/w dev has to fit inside of 2-wk or 4- wk iterations

<table>
<thead>
<tr>
<th>Flexibility</th>
<th>Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>Software</td>
<td>User settings</td>
</tr>
<tr>
<td></td>
<td>App s/w</td>
</tr>
<tr>
<td></td>
<td>Operating Sys</td>
</tr>
<tr>
<td></td>
<td>Reprogram’ble (PLDs)</td>
</tr>
<tr>
<td></td>
<td>On-site reconfigurable</td>
</tr>
<tr>
<td></td>
<td>In-house reconfigurable</td>
</tr>
<tr>
<td>Electronic H/W</td>
<td>Components</td>
</tr>
<tr>
<td></td>
<td>Housings</td>
</tr>
<tr>
<td>Mechanical H/W</td>
<td>Special materials</td>
</tr>
</tbody>
</table>
Iterations work differently

- Each junction gives tangible baseline each person sees

<table>
<thead>
<tr>
<th>Flexibility</th>
<th>Time</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td>Software</td>
<td></td>
</tr>
<tr>
<td>Electronic H/W</td>
<td></td>
</tr>
<tr>
<td>Mechanical H/W</td>
<td></td>
</tr>
<tr>
<td>User settings</td>
<td></td>
</tr>
<tr>
<td>App s/w</td>
<td></td>
</tr>
<tr>
<td>Operating Sys</td>
<td></td>
</tr>
<tr>
<td>Reprogram'ble (PLDs)</td>
<td></td>
</tr>
<tr>
<td>On-site reconfigurable</td>
<td></td>
</tr>
<tr>
<td>In-house reconfig'ble</td>
<td></td>
</tr>
<tr>
<td>Components</td>
<td></td>
</tr>
<tr>
<td>Housings</td>
<td></td>
</tr>
<tr>
<td>Special materials</td>
<td></td>
</tr>
</tbody>
</table>

GMS Agile s/w helped h/w

- Only the s/w team was using Agile practices, but…
- Frequent s/w releases created many more opportunities to improve h/w-s/w interaction
  - Some measurements inconclusive due to voltages out of range – so added s/w monitoring of h/w key areas
  - Field problems that could not be isolated to one area (opto, sensor, electronics) could be investigated thru special s/w releases for troubleshooting
  - Hand assembly of field units improved by downloadable collection of s/w drivers with command-line menu
- Result was h/w became more Agile “without trying”
Topics

- Hardware roots of “Agile”
- Agile Engineering: examples
- Team composition mirrors workflows
- Want predictability? Or fast learning?

Example: Ops + Software team

- Ops people could only do ops stories
- IT people could only do IT stories

Should they cross-skill? That would be usual coach advice.
In this case – NO!
Ex.: Physicists-turned-C++ coders

- Each member is physicist or mathematician
- Each member is also a C++ experienced coder

Should they have cross-skilled?
That would be usual coach advice.
In this case – absolutely YES!

Hardware + Software team(s)

- Story board is divided…

But we’re still one team, now with different workstreams visible

This same mechanism can work for two cooperating teams
Lanes not independent

- Keep focus on whole features; don’t merely fit work to skill siloes

People pair to do their parts of features that span disciplines

Topics

- Hardware roots of “Agile”
- Agile Engineering: examples
- Team composition mirrors workflows
- Want predictability? Or fast learning?
Predictability or Fast Learning

- FPGA example: Predictability
- Wikispeed Car example: Fast learning
- Most projects: Predictability for coordination, Fast learning to handle unknowns

Everyone thinks they want predictability
- Increasingly, I believe they all really need fast learning
- “Instead of freezing the ocean, learn to ride the waves”
Contact info

Nancy Van Schooenderwoert
LeanAgilePartners
nancyv@leanagilepartners.com
@vanschoo

Lean-Agile coaching for software and hardware teams
- Safety-critical, regulated coaching is our specialty
- Lean-Agile coaching for stakeholders and senior managers

References & Further Info

REFERENCES:
- Wikispeed Project: www.wikispeed.com

FURTHER INFO & Credits:


Surfer photo: By The Last Minute (Flickr) [CC BY 2.0 (http://creativecommons.org/licenses/by/2.0)], via Wikimedia Commons; used unchanged.
Wikispeed photos used with permission of the Wikispeed Project.
Snail drawing by W. H. Dall (1845-1927)