I Tried to Avoid Computers
People will pay me to do this?!!

The IMSAI 8080 is a microcomputer built by IMSAI Computers in 1975. It was one of the first microcomputers to be widely available to hobbyists. It used the Intel 8080 microprocessor, which was a significant departure from the Intel 4004, which was the first microprocessor to be widely available. The IMSAI 8080 was a popular choice for hobbyists because it was relatively inexpensive and easy to assemble. It was also one of the first computers to be used for educational purposes, and it helped to popularize the concept of personal computing.
Did that work?

The industry suffered from defects, delays and frustration.

We Make Our Problems

Vague Requirements  
Unrealistic Plan  
Missed Customer Expectations

Unplanned Activities

Unpredictable Activities

Poor Quality

Positive Feedback

Unstable System

Vague Requirements

Unrealistic Plan

Unplanned Activities

Unpredictable Activities

Missed Requirements

Late Project

Schedule Pressure

Long Hours

Bugs

Poor Quality

Manifesto for Agile Software Development

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 processes 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 in the items on the right, we value the items on the left more.
Agile is Designed to Help Break this Vicious Cycle

Can Projects be Managed Better?

From Winston Royce’s original paper on Waterfall

Figure out everything, then do what you figured
Steve McConnell from Rapid Development

“Software projects contain too many variables to be able to set schedules with 100-percent accuracy. Far from having one particular date when a project would finish, for any given project there is a range of completion dates, of which some are more likely and some are less.”
What is Agile?

• Agile software development is a conceptual framework for undertaking software engineering projects.
  -- wikipedia
• a.k.a. Extreme Programming, Scrum, Feature Driven Development, DSDM, Crystal Clear, Agile Unified Process

Agile methods are Designed to...

• Manage with Data
• Improve Visibility
• Improve Predictability
• Improve Quality
• Improve Productivity
• Reduce Waste
**Manifesto for Agile Software Development**

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 processes 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 in the items on the right, we value the items on the left more.

---

**Skilled Self-Organizing Teams**

- Developers work together to organize the work
- Customer or Product Owner works with the teams to define the product and establish priorities
- Managers usually take an outward focus, removing roadblocks, rather than managing day-to-day tasks and schedules.

---

**Collaboration**

- Daily standup meeting
- Pair programming, Mob programming or Daily reviews
- Shared code ownership
- Team room

---

**Individuals and Interactions over Processes and Tools**
Working Software over Comprehensive Documentation

- Each team has different needs
- Less formal documentation might work.
- Prefer executable Documentation

Agile is Mostly Silent about Documentation

- Your needs are special
- There is no single prescription

Customer Collaboration over Contract Negotiation

Responding to Change over following a plan

Drive:

1,121 mi (about 17 hours 43 mins)
Agile Approach is more...

- Visible
- Predictable
- Productive

- With a focus on
  - High Quality Work
  - Reduced Waste

Agile Principles

- Plans
  - Visible
  - Realistic

- People
  - Communications
  - Courage
  - Respect
  - Honesty

- Product
  - Customer Focus
  - High Quality
  - Simple
  - Responsive

- Feedback

Iterative and Incremental Development

Projects end, products don’t (hopefully)

Requirements analysis is never done
Design is never done
Why Iterative?

• A system's users seldom know exactly what they want and cannot articulate all they know
• … There are many details we can only discover once we are well into implementation
• … as humans we can only master only so much complexity
• … external forces lead to changes in requirements…

[LARMAN]  

Project Progress is Measurable
Functionality Built and Tested

Complete Stories

Sprints or Iterations (2-4 weeks)

Measure Development Velocity
Estimated work per Iteration
Product Burn Down Chart
Work to be Completed

New features added—what happens?
The Backlog is Made up of Stories

- The short term plan is more detailed.
- Work on it, buying time to refine longer term plan.
- Generally stories are in the order set by customer.
- Engineers can ask to move up stories to reduce risk.
- Stories are tested in the iteration they are implemented; story tests are automated.
- A story is done when it passes its tests.

Introducing the User Story

- The name of a feature.
- A promise for a conversation. (Ron Jeffries)
- Like the name of a use case, or extension.
  - Acceptance tests provide the details.
- Fine grains help make visible progress and avoid gold plating.
- I call them Product Stories
High Level Requirements to Stories

The product must:
- Secure password
- Schedule lights
- Schedule blinds
- Security camera
- Perimeter alarm
- Phone police
- Call broker-in-law
- Entry delay
- Party mode
- Software updates
- Network config

Epic stories (story areas)

Focused stories

Stories to Stories

Deliver the most bang for the buck first.

Invest in User Stories

Source: Bill Wake

- Independent
- Negotiable
- Valuable
- Estimate-able
- Small
- Testable

Valuable stories

- The ideal story delivers value to the customer.
- May be unrealistic for some embedded stories.
- Sometimes Visible is all you can get.

Why Small and Visible?

- One big story, with greater risk and no feedback
- Four smaller stories, with less risk and more feedback
- Value realized
What if Part of the Work is not Valuable or Urgent?

Stories

Consider the Customer

Who benefits from the story

Hardware engineer

Identify Flash Device

To verify address logic

Why is it valuable

Stories are a Vertical Slice of Functionality

• They cut across subsystem boundaries
• They can include
  – UI (graphics, preliminary or polished)
  – System behavior
• They have an agreed upon definition of DONE.
• Prefer visible work over engineering tasks
• But sometimes a story is more horizontal, or isolated to an layer

Make Work Visible

Find a Natural Order

• USB device detected
• Device identified as a flash drive
• Erase flash memory device
• Read from flash memory device
• Write to flash memory device
• Open an existing file
• Read from an existing file
• Open a file for writing
• Write to the file
Story Mapping Helps Discover the Natural Delivery Order

Stories and Acceptance Tests

- Stories lack detail
- Details are provided in automated acceptance tests
- The test are like executable use cases
- Test either pass or fail
Completed Stories Have Passing Acceptance Tests

<table>
<thead>
<tr>
<th>Test Suites</th>
<th>Test Pages</th>
<th>Right</th>
<th>Wrong</th>
<th>Ignored</th>
<th>Exceptions</th>
</tr>
</thead>
<tbody>
<tr>
<td>TestSuite</td>
<td>14</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Tests Passed</td>
<td>156</td>
<td>0</td>
<td>0</td>
<td>7</td>
<td>0</td>
</tr>
</tbody>
</table>

When the Story Test is Ready Before the Development Finishes

<table>
<thead>
<tr>
<th>Test Suites</th>
<th>Test Pages</th>
<th>Right</th>
<th>Wrong</th>
<th>Ignored</th>
<th>Exceptions</th>
</tr>
</thead>
<tbody>
<tr>
<td>TestSuite</td>
<td>13</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Tests Passed</td>
<td>154</td>
<td>2</td>
<td>7</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

When Things Break

<table>
<thead>
<tr>
<th>Test Suites</th>
<th>Test Pages</th>
<th>Right</th>
<th>Wrong</th>
<th>Ignored</th>
<th>Exceptions</th>
</tr>
</thead>
<tbody>
<tr>
<td>TestSuite</td>
<td>5</td>
<td>11</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Tests Passed</td>
<td>129</td>
<td>27</td>
<td>7</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

How are We Doing in Agile’s 16th Year?

<table>
<thead>
<tr>
<th>Test Suites</th>
<th>Test Pages</th>
<th>Right</th>
<th>Wrong</th>
<th>Ignored</th>
<th>Exceptions</th>
</tr>
</thead>
<tbody>
<tr>
<td>TestSuite</td>
<td>3</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Tests Passed</td>
<td>69</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

- Jama Software Webinar — Overview and Pitfalls
- Agile for Embedded — June 15, 2017
- www.wingman-sw.com
- jama@wingman-sw.com

Copyright © 2008-2017 James W. Grenning
All Rights Reserved  @jwgrenning
What is Scrum?

Scrum exposes every inadequacy or dysfunction within an organization’s product and system development practices. The intention of Scrum is to make them transparent so the organization can fix them.

From an interview with Ken Schwaber on agilecollab - http://www.agilecollab.com/interview-with-ken-schwaber

What is Scrum?

Certified Scrum Masters

Like in the 80s! Short Cycle Improvement

Except without the Difficult Improvement Part
Agile’s Two Halves
Miss Any, Miss a Lot

Iterative planning
Technical Excellence

Agile’s Three Halves
Miss Any, Miss a Lot

Iterative planning
Respect for people
Technical Excellence

There are too many Dogma followers!
And not enough problems solvers

Remember what Ken Said?

"The intention of Scrum is to make them [the problems] transparent so the organization can fix them."

Unfortunately, many organizations change Scrum to accommodate the inadequacies or dysfunctions instead of solving them.
Did that work?

The industry STILL suffers from defects, delays and frustration.

So Far the Marriage is Unequal

Does this mean it takes six CSMs to master one CSD?
Quora Question:
In a nutshell, why do a lot of developers dislike Agile?

I'm a programmer.
Written Jul 5, 2016 - Upvoted by 15+ years as a software developer, and
CEO of Software (2006-present)
This story is 100% true.

Omitted: Rant on something called Agile

146.4k Views · View Upvotes

James Grenning, Tried to avoid programming in the early 70's, then got
hooked. Loving it since.
Updated Nov 5 - Upvoted by Marcelo De Zen, 15+ years as a software developer., Magnus
Falk, 10 years a programmer & tester in large distributed systems, and Kenrick Chien

Omitted: Explanation of how Agile is misunderstood

137.4k Views · View Upvotes
A Bug’s Life

From http://www.softwaretestinghelp.com/bug-life-cycle/

Do Your Work Degrade into a Fire Fight?

Why Does This Happen??

This Work Flow is Designed to Allow Defects

The Physics of Debug Later Programming (DLP)

- As \( T_d \) increases, \( T_{\text{find}} \) increases dramatically
- \( T_{\text{fix}} \) is usually short, but can increase with \( T_d \)

http://blog.wingman-sw.com/archives/16
Edsger Dijkstra

*If you want more effective programmers, you will discover that they should not waste their time debugging, they should not introduce the bugs to start with.*

XP Identifies values and engineering practices needed for growing computer software

The Marriage of Scrum and XP

Yeah but I’m an engineer
Quality is important  ➔ Focus on it every day
Testing is good  ➔ Test all the time
Reviews are good  ➔ Review all the time
Customer input is good  ➔ Talk to your customer every day
Planning is important  ➔ Keep the plan alive, plan to replan

If test moved upstream, could we get working features to flow?
Development and Test are a Continuum

Preventing Defects

The Physics of Test Driven Development

- When $T_d$ approaches zero, $T_{find}$ approaches zero
- In many cases, bugs are not around long enough to be considered bugs.
- See: https://wingman-sw.com/articles/the-physics-of-test-driven-development

Manual Test is Unsustainable

Risk Accumulates in the Untested Code Gap
Why does code quality matter?

The Lawyers are Coming

Toyota settles acceleration case after $3 million jury verdict

By Chris Isidore @CNNMoney October 25, 2013: 2:27 PM ET

Toyota has already agreed to pay $1.1 billion to settle a class-action suit by owners who saw the resale value of their cars decline. A little more than half of that settlement went toward installing a "brake override" system in affected cars. The settlement does not cover cases in which personal injury or death occurred.

According to financial filings, the automaker still faces more than 700 acceleration cases.
We’ve demonstrated how as little as a single bit flip can cause the driver to lose control of the engine speed in real cars due to software malfunction that is not reliably detected by any fail-safe.

- Michael Barr, CTO, co-founder of Barr Group
Why doesn’t your team use TDD and Refactoring and the other technical practices of Extreme Programming?
We’re different!

We can’t use Agile!

Agile’s Three Halves
Miss Any, Miss a Lot

Challenges for Agile and Embedded

- Stories and incremental scope control
- Breaking dependencies on hardware
- Applying outside of software
  - Mechanics, hardware, ASIC development
- Not unique to embedded, though prevalent
  - Your own preconceived notions
  - Organizational resistance
"We are what we repeatedly do. Excellence, then, is not an act, but a habit.”

Aristotle

https://hakanforss.wordpress.com/2014/03/10/are-you-too-busy-to-improve/

Questions?

Talk to me on Twitter
@jwgrenning

Find my book at
http://wingman-sw.com/tddec

Find me on LinkedIn
http://www.linkedin.com/in/jwgrenning
Please remind me how we met.

http://facebook.com/wingman-sw
http://www.wingman-sw.com
http://www.wingman-sw.com/blog

Coming Soon

Webinar
Test-Driven Development and Engineering Practices for Embedded

Web Delivered Training — July
TDD for Embedded C — wingman-sw.com