Agile Practices
For everybody - not just developers
• Pragmatism
• Values to Support Pragmatism
• Practical Habits to Support Values
• Reduce Stress with Action
The Values of Extreme Programming

Extreme Programming (XP) is based on feedback: We will take every iteration seriously by delivering working natural, extensible and comprehensive software. We demonstrate our software every iteration maximizing our values. XP isn’t really a set of rules but rather a way to work in harmony with changes needed. We will talk about the rules but rather a way to work in harmony with changes needed. XP isn’t really a set of rules but rather a way to work in harmony with changes needed. We will talk about the rules but rather a way to work in harmony with changes needed.

Respect: Everyone gives and gets respect they deserve as a valued team member reflecting them in the changes you make to the rules.

Simplicity: We will do what is needed and everyone, contributes value of their work. This will minimize the enthusiasm and value we create for the investment made in code. In general, we will take small, simple steps to our goals and respect our right to mitigate failure as we happen. We will create positive authority over the things we are proud of and maintain it long-term for reasonability.

Courage: We will do what is needed and everyone, contributes value of their work. This will minimize the enthusiasm and value we create for the investment made in code. In general, we will take small, simple steps to our goals and respect our right to mitigate failure as we happen. We will create positive authority over the things we are proud of and maintain it long-term for reasonability.

Communication: Everyone is part of the team, failure because we communicate face to face daily and we communicate face to face daily. We fear anything we will work together on everything from We will solve the problem that we can together.

The Scrum Guide™

The Definitive Guide to Scrum

The Rules of the Game

Ken Schwaber

July 2013
Time Yourselves!

WOOT

WOOT!!
How did you do?
This time, the rightmost column MUST sum to 15.

<p>| | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>15</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
The top row MUST ALSO sum to 9.

WOOT WOOT!!
The diagonal MUST ALSO sum to 18.

WOOT WOOT!!
Time to Solve

Interactive Rule Complexity

0 1 2 3
Still just 9 digits and 4 lines, folks.

Why does it take longer now?
Estimate how long this will take you.
What does this have to do with Agile practices?
SIP #1:

Software Development is all about making decisions.
SIP #2:

Software Development is a flow, subject to turbulence.
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.
the right, we value the items on the left more.

Kent Beck  James Grenning  Robert C. Martin
Mike Beedle  Jim Highsmith  Steve Mellor
Arie van Bennekum  Andrew Hunt  Ken Schwaber
Alistair Cockburn  Ron Jeffries  Jeff Sutherland
Ward Cunningham  Jon Kern  Dave Thomas
Martin Fowler  Brian Marick

© 2001, the above authors. This declaration may be freely copied in any form, but only in its entirety through this notice.

Twelve Principles of Agile Software

Become a Signatory
Principles behind the Agile Manifesto

We follow these principles:

Our highest priority is to satisfy the customer through early and continuous delivery of valuable software.

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

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

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

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

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

Working software is the primary measure of progress.

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

Continuous attention to technical excellence and good design enhances agility.

Simplicity—the art of maximizing the amount of work not done—is essential.

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

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

http://agilemanifesto.org/principles/
Working software is the primary measure of progress.

Our highest priority is to satisfy the customer through early and continuous delivery of valuable software.
SIP #3: Agile is about delivering value.
Deliver working software frequently, from a couple of weeks to a couple of months, with a preference to the shorter timescale.
So, what do you really do?
FASTER

SOONER

FASTER
Practice:

**Incremental Delivery**
Planning Stress

Deferral of crucial activities

Internal Consequences

External Consequences

Release
Pick some small period of time.
Pick only two weeks worth of work everyone understands
About 2 Weeks

Two weeks worth of stuff actually works here.
100% of system
30% done
No testing yet

30% of the system
100% done
With known quality
Practice:

“Whole Team”
Build projects around motivated individuals.
Give them the environment and support they need, and trust them to get the job done.
Manager’s View

About 2 Weeks

HANDS OFF
No one (not even the Scrum Master) tells the Development Team how to turn Product Backlog into increments of Potentially releasable functionality

-- The Scrum Guide
What is this?
Communism?
Faith-based management?
It's Official: Micromanaging Kills Productivity

BY JILL KRASNY @JILLKRASNY

New research suggests the more you try to dictate how and when employees work, the less they will accomplish.
3 FACTORS LEAD TO BETTER PERFORMANCE & PERSONAL SATISFACTION...

AUTONOMY

MASTERY

PURPOSE
work. We start from the presumption that our people are talented and want to contribute. We accept that, without meaning to, our company is stifling that talent in myriad unseen ways. Finally, we try to identify those impediments and fix them.
Practice:

Sustainable Pace
Agile processes promote sustainable development. The sponsors, developers, and users should be able to maintain a constant pace indefinitely.
Manager’s View

About 2 Weeks

If you are only working in 2 week periods...

Only plan 2 weeks worth.

Only deliver 2 weeks worth.

See what can happen in 2 weeks.
Easily said, but how do I decide how much the team can do in two weeks?
What about estimating, planning poker, story points?
Practice:

Planning
Game
Most important
<table>
<thead>
<tr>
<th>To Increase</th>
<th>You Must:</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Velocity</strong></td>
<td>Increase Effort</td>
</tr>
<tr>
<td></td>
<td>Cut corners</td>
</tr>
<tr>
<td></td>
<td>Take chances</td>
</tr>
<tr>
<td><strong>Capacity</strong></td>
<td>Develop skills</td>
</tr>
<tr>
<td></td>
<td>Increase Knowledge</td>
</tr>
<tr>
<td></td>
<td>Improve tools</td>
</tr>
<tr>
<td></td>
<td>Share work efficiently</td>
</tr>
<tr>
<td></td>
<td>Reduce waste</td>
</tr>
</tbody>
</table>
SIP #4:

Capacity (velocity) is a consequence, not a choice.
Omigersh! Unclaimed Capacity!
How likely is it that someone will be sick?

...that training will be scheduled over your work time?

...that you will hit a technical snag?

...that your estimates are right to begin with?
Good use of unclaimed slack!
<table>
<thead>
<tr>
<th>Practice:</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Backlog Grooming</strong></td>
</tr>
</tbody>
</table>
Shrink XL Stories to Fit

» Defer alternate paths, edge cases, or error cases
» Defer supporting fields
» Defer side effects
» Stub dependencies
» Split operationally (for example, CRUD)
» Defer nonfunctional aspects
» Verify against audit trail
» Defer variant data cases
» Inject dummy data
» Ask the customer
Stories will compete for priority
SIP #5:
The backlog is an optional set of things we *may* do, not the set of things we *must* do.
Advanced Topics

- Minimal Work In Progress (WIP)
- Minimal Backlog (approaching none)
- Value Stream Mapping
- NoEstimates
<table>
<thead>
<tr>
<th>Practice:</th>
</tr>
</thead>
<tbody>
<tr>
<td>Continuous Integration</td>
</tr>
</tbody>
</table>
Practice:

Evolutionary Design
"It is not the strongest of the species that survives, nor the most intelligent that survives. It is the one that is most adaptable to change."

Charles Darwin
Welcome changing requirements, even late in development. Agile processes harness change for the customer's competitive advantage.
What I’ve built but don’t need is a lost investment.

What I need but don’t have is a sure investment.
We can shrink or drop stories, but can we also increase capacity?
At regular intervals, the team reflects on how to become more effective, then tunes and adjusts its behavior accordingly.
<table>
<thead>
<tr>
<th>Practice:</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Retrospective</strong></td>
</tr>
</tbody>
</table>
Retrospectives

- **Set the stage**: Get everyone to speak. Agree on rules. Use a *safety exercise*.
- **Gather data**: Feelings are legitimate data!
- **Generate insights**: “Why?” Begin discussing how to do things differently.
- **Decide what to do**: Commit to one to two action items or experiments.
- **Close the retrospective**: Review the retrospective itself. Capture info. Thank all.
SIP #6:

Agile methods are empirical
Could you write a spreadsheet that auto-summed?

Could you learn all the ways unique add up to a given sum?

Could you write a solver program?

Could you write a spreadsheet that auto-summed?

How else would you "Adapt"?
import itertools

combinations = (
    (sum(x), x)
    for x in itertools.combinations(range(1,10), 3)
)

for total, sequence in sorted(combinations):
    print total, sequence
Practice:

Continuous Improvement
Learning & experimenting takes *time*, and we are always in a time crunch!
I was told to expect hyper-productivity.
Our highest priority is to satisfy the customer through early and continuous delivery of valuable software.
SIP #7:

Agile is sooner, not faster.
Cheat
SIP #8:

Agile is an alternative, advantageous ruleset (AKA: “cheating”).
Simplicity--the art of maximizing the amount of work not done--is essential.
Cheat at an Exam
Early Testing
Few implementations of Scrum achieve the hyperproductive state for which Scrum was designed (5-10 times normal performance).

Those that do *all implement variations on XP engineering practices* and many, like the CMMI Level 5 implementation of Scrum (which has not achieved the hyperproductive state yet) drive the whole implementation from a lean perspective.

http://www.scruminc.com/origins-of-scrum/
IF you see the first scrum at Easel Corporation implemented by Jeff Sutherland, *it incorporated all the activities of extreme programming’s engineering practices*.

All the most high-performance teams use Scrum and XP together. It is hard to get a Scrum with extreme velocity without XP engineering practices.

Privacy of error

Colleague in Team

Other Department

Other Org (etc.)

World

Twin → Y

Person
Pen Testing & Perf Testing

Regression Testing

TDD

BDD

- TEST
- Standards
- Code
- PARTNER
SIP #9:

If something is hard, we do it more often and automate it to death.
Practice:

Pair Programming
Continuous attention to technical excellence and good design enhances agility.
SIP #10:

We avoid failure by making safer bets.
Continuous Integration
Automated Testing
Incremental Releases
Planning Game
Retrospectives
Pair Programming
Continuous Improvement
When do I install the agile life cycle management tools?
Do I have to hire a consultant?