Is Agile just a Software thing?

www.agilemanifesto.org

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.
<table>
<thead>
<tr>
<th>To do</th>
<th>Going on</th>
<th>Done!</th>
</tr>
</thead>
<tbody>
<tr>
<td>Drink table</td>
<td>Dip snacks</td>
<td>Light the BBQ</td>
</tr>
<tr>
<td>Orange Juice</td>
<td>Vegan sticks</td>
<td>Salads</td>
</tr>
<tr>
<td>Food serving pack</td>
<td>BBQ</td>
<td>Ice cream</td>
</tr>
<tr>
<td>BBQ</td>
<td>Grapes + cheese + crackers</td>
<td></td>
</tr>
</tbody>
</table>
Recruitment team

Henrik Kniberg
Recruitment team
Agile practices implemented at every level and in every discipline: software, hardware and fuselage design.

Pilots on the same site as development teams. Direct feedback provided every sprint.

1500 people, all co-located in Linköping, Sweden.

Compared to F35 joint strike fighter, Gripen 39E has:
- 50x lower development cost!
- 10x lower unit cost!

World’s most cost-effective military aircraft ($4700 Cost per Flight Hour)

Sources:
- Personal visit to SAAB Linköping
- Research paper "Owning the Sky with Agile"

Henrik Kniberg
Henrik Kniberg

Tools
“anything used as a means of accomplishing a task or purpose.”
- dictionary.com

Physical tools

Thinking tools
a.k.a. “mindsets” or “philosophies”

Lean  Agile  Systems Thinking  Queuing theory

Toolkit
a.k.a. “frameworks”

Scrum  Kanban  XP  SAFe

Process tools
a.k.a. “organizational patterns”

Product Owner role

Pair programming

Visualize management

FLOW

- To do
- Dev
- Test
- Release
- Done

- 1
- 2
- 3
- 4
- 5

Henrik Kniberg
Lean

Agile
Lean  Agile

Henrik Kniberg
**Toyota Production System**

Best Quality - Lowest Cost - Shortest Lead Time
Best Safety - High Morale

through shortening the production flow by eliminating waste

**Just-In-Time**
- right part, right amount, right time
  - Takt Time Planning
  - Continuous Flow
  - Pull System
  - Quick Changeover
  - Integrated Logistic

**People & Teamwork**
- Selection
- Common Goals
- Ring Decision Making
- Cross-trained

**Continuous Improvement**

**Waste Reduction**
- Gendoch Gembutsu
- 8 Whys
- Eyes for Waste
- Problem Solving

**Jidoka**
- (In-station Quality), make problems visible
  - Automatic Stop
  - Arden
  - Person-Machine Separation
  - Error Proofing
  - In-Station Quality Control
  - Solve Root Cause of Problems (8 Whys)

**Leveled Production (heijunka)**

Stable and Standardized Processes

Visual Management

Toyota Way Philosophy

---

**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 process 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.

---

**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.

---

Henrik Kniberg
Agile Manifesto

www.agilemanifesto.org

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 solutions** over **comprehensive documentation**

**Customer collaboration** over **contract negotiation**

**Responding to feedback** 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 not new

Buzzwords will come and go, but the underlying ideas and principles are timeless
• 2.3 million blocks
• 6 million tons
• 140 meters high
• Tallest man-made structure for 3800 years
Iterations, Continuous Improvement, Pull, Single-piece flow 4500 years ago

Velocity of Khufu’s pyramid construction: 1 block every 2.5 minutes ... for 30 years!!!

Henrik Kniberg
Beware of Tool Misuse

If all you have is a hammer, everything looks like a nail

Abraham Maslow
Misguided Lean

Solving the wrong problem

Revealing the right problem

Henrik Kniberg

Photo: http://leanactionplan.pl/o-nas/artykuly_lean/Lean-Office;183.html
The Agile "umbrella"

Using an agile framework doesn’t automatically make you agile!

You can be agile without using ANY of these frameworks.
Henrik Kniberg
Early on, **all of our movies suck.**

That’s a blunt assessment, I know, but I choose that phrasing because saying it in a softer way fails to convey how bad the first versions really are.

Our job is to make them go from **Suck to Not-Suck.**

---

**Example: Pixar**

Ed Catmull  
President of Pixar & Disney Animation Studios
In the early stage of making a movie, we draw storyboards (a comic-book version of the story) and then edit them together with dialogue and temporary music. The first versions are very rough, but they give a sense of what the problems are, which in the beginning of all productions are many.

We then iterate, and each version typically gets better and better.
The further you are from software development, the less likely that any of the popular frameworks will fit 100%.

Henrik Kniberg
Understand the Why of each tool

- Story points
- Definition of Done
- Cross-functional team
- User stories
- Daily standup
- Continuous Integration
- Cadence
- WIP limits
- Sprint
- Velocity
- Value stream mapping
- TDD
- Pair programming
- Scrum Master
- Daily standup
- Retrospective
- XP

Henrik Kniberg
Example: Why Sprints?
Compromise between stability & flexibility.

Too much “stability”

Too much “flexibility”

Sprint = stability + flexibility

Henrik Kniberg
When is Agile most needed?

- **When** to deliver:
  - Unclear/unstable
  - Clear & stable

- **How to deliver it**:
  - Clear & stable (Any process works here)
  - Unclear/unstable (Predictive process doesn’t work here. Need an adaptive process. (feedback loops rather than detailed plans))

Henrik Kniberg
Why is Agile spreading so fast?

Henrik Kniberg

Unclear/unstable

Clear & stable

What to deliver

How to deliver it

Agile is optimized for this

Predictive process doesn’t work here.
Need an adaptive process.
(feedback loops rather than detailed plans)

The world is moving this way
The role of copy-paste

Scrum and XP from the Trenches

Spotify Engineering Culture

Copy → Paste

Copy → Paste → Adapt

Henrik Kniberg
Strategies for applying agile in other contexts

- Implement method X “by the book”, and follow the rules religiously
- Implement method X “by the book”, then customize it
- Cherry-pick specific practices
- Apply agile ideas directly, without using any specific framework
Strategies for applying agile in other contexts

1. Implement method X “by the book”, and follow the rules religiously

2. Implement method X “by the book”, then customize it

3. Cherry-pick specific practices

4. Apply agile ideas directly, without using any specific framework

Henrik Kniberg
Example: Big Family Trip
**Departure Date:** Oct 1

**Preparations**

- Att göra
- Nästa
- Pågjorde
- Klart!

**Motive**

- Vision
- Drömmar
- Bild
- Vision
- Förädla

**Must Do**

- Förädla
- Förändra
- Förändra
- Förändra

**Should Do**

- Förändra
- Förändra
- Förändra
- Förändra

**Could Do**

- Förändra
- Förändra
- Förändra
- Förändra

**Henrik Kniberg**
Travel “spike”

Small Family Trip
London, 4 days

Big Family Trip
Round the world, 6 months

Henrik Kniberg
On-the-road schooling using velocity, cadence, and burnup chart
“School” is every day after breakfast, regardless of location
On-the-road schooling using velocity, cadence, and burnup chart

“On track” = bottom bar is ahead of top bar
Back home from the trip...

Why is the kitchen always such a mess suddenly?

We didn’t have that problem when travelling. Why?

Henrik Kniberg
Henrik Kniberg

1-2 days!

Root cause: Dishwasher!
< 1 hour!
Worked like a charm!
but did we keep doing it?

Henrik Kniberg
Sometimes agile practices don’t stick. That’s Fine.

Explanations:
• The practice was only needed for a specific situation
• The practice didn’t work too well
• The practice was a stepping stone until a better practice was found
• The practice was only needed to learn & internalize a new behaviour
Pattern: Go all-in first, then go pragmatic

**Batching**

1. Dishwasher

**Kitchen**

2. No batch testing
3. No dishwasher, WIP limit, personal sets

**Test automation**

1. No tests
3. “Good enough” test coverage.
   Tests & code in the same commit
   TDD when needed

**Lean**

2. Full TDD
Example: Using a practice only when needed

Agile Product Ownership in a Nutshell
- Production time: 2 days

Takes a couple of days to make a cool animated video

Spotify Engineering Culture video – part 1
- Expected production time: A few days
- Actual production time: Several weeks!

Whoa! That took MUCH longer than I expected!

How can I avoid the same problem for Part 2?

Henrik Kniberg
Video storyboard (rough sketches)

Henrik Kniberg
"Pointifying" the work

<table>
<thead>
<tr>
<th></th>
<th>Drawing</th>
<th>Voice</th>
<th>Flow</th>
<th>Currently done</th>
<th>Max</th>
<th>Remaining</th>
</tr>
</thead>
<tbody>
<tr>
<td>Intro</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>6</td>
<td>6</td>
<td>0</td>
</tr>
<tr>
<td>Recap</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>6</td>
<td>6</td>
<td>0</td>
</tr>
<tr>
<td>Fail fast</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>6</td>
<td>6</td>
<td>0</td>
</tr>
<tr>
<td>Retrospectives</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>6</td>
<td>6</td>
<td>0</td>
</tr>
<tr>
<td>Limited blast radius</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>6</td>
<td>6</td>
<td>0</td>
</tr>
<tr>
<td>Lean startup</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>6</td>
<td>6</td>
<td>0</td>
</tr>
<tr>
<td>Innovation</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>6</td>
<td>6</td>
<td>0</td>
</tr>
<tr>
<td>Hack time</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>6</td>
<td>6</td>
<td>0</td>
</tr>
<tr>
<td>Experiment friendly</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>6</td>
<td>6</td>
<td>0</td>
</tr>
<tr>
<td>Waste repellant</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>6</td>
<td>6</td>
<td>0</td>
</tr>
<tr>
<td>Big projects</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>6</td>
<td>6</td>
<td>0</td>
</tr>
<tr>
<td>Growth pain</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>6</td>
<td>6</td>
<td>0</td>
</tr>
<tr>
<td>Improvement board</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>6</td>
<td>6</td>
<td>0</td>
</tr>
<tr>
<td>Healthy culture</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>6</td>
<td>6</td>
<td>0</td>
</tr>
<tr>
<td>Spreading</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>6</td>
<td>6</td>
<td>0</td>
</tr>
<tr>
<td>You are the culture</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>6</td>
<td>6</td>
<td>0</td>
</tr>
</tbody>
</table>

Total: 96 / 96 / 0

Henrik Kniberg
Pomodoro Technique
Personal Scrum with 30 minute sprints

Decide what to focus on

100% Focus! No interruptions!

25 minutes

Pomodoro 1

5 minute break

Decide what to focus on

100% Focus! No interruptions!

25 minutes

Pomodoro 2

Measure:
- How much can I get done in one Pomodoro?
- How many Pomodoros can I execute per day / week?

Henrik Kniberg
Used Yesterday’s Weather and burndown chart to reliably forecast when the video would be done.
Strategies for applying agile in other contexts

- Implement method X “by the book”, and follow the rules religiously
- Cherry-pick specific practices
- Implement method X “by the book”, then customize it
- Apply agile ideas directly, without using any specific framework

Henrik Kniberg
Robit

Henrik Kniberg
LEGO® MINDSTORMS® COMPETITION

Do you have what it takes to fight and win the competition of the future? - Then sign up and enter the LEGO® MINDSTORMS® Robotic competition at GOTO Copenhagen 2015 and win fabulous prizes on top of the fame and glory!

How to enter the game?

1. Form a team of 2-5 members (NB: Only conference attendees can join the competition)
2. Build your own intelligent, autonomous robot before the conference (use your own LEGO® MINDSTORMS® Robotic Toolkit or borrow one for free when registering to the competition)
3. Pitch it against the robots from other teams at the GOTO Conference Dinner, Monday October 5, 19:30-22:30
4. There will be prizes for the winning team

Henrik Kniberg
2 kids & rookies with very little robot experience...

... vs ten teams of adult geeks and programmers
Step 1: Set a clear goal (define “success”)

Let’s build a robot that at least can put a fight....

No! We’re going to WIN!

Henrik Kniberg
Agile

I'm going to have to science the shit out of this

Henrik Kniberg
Step 2: Build a Minimum Viable Robot (Earliest Testable Robot)
Aim for the clouds, but deliver and test in small steps

Can stay in the ring

Can find opponent

Henrik Kniberg
Step 3: Build an opponent to practice against
Henrik Kniberg
Aim for the clouds, but deliver and test in small steps

Can stay in the ring

Can find opponent

Can get to opponent

Can budge opponent

Can win match against a static opponent

Henrik Kniberg
Lifter? Or no lifter?

Hypothesis:
- Mechanical Lifter can help us win

Experiment:
- Build a simple lifter and try

Learning:
- Works as designed...
- But too weak to lift opponent
- ... so it doesn’t help us win!

Options:
- Keep it cuz it’s cool (who needs to win anyway)
- Improve it
- **Remove it, try a different approach**

Henrik Kniberg
Simpler was better
Field testing = Success by 100 failures

Henrik Kniberg
Henrik Kniberg
How could they win?

Building skill? No.
Programming skills? No.
Luck? Partly, but not entirely.

1) Clear goal
2) Low self-confidence
3) Emergent design
4) LOTS of field testing!

Henrik Kniberg
Some tips when applying agile in <insert domain here>
Don’t inflict help on people. Pull works better than Push.

**Push**

Hey we’re using Scrum here, you should too!

Invites resistance and “not invented here” syndrome

**Pull**

Hey, your way of working looks interesting. Think something similar might work for us?

I like it! Can you help us get started?

Sure. Tell me more about your domain, and we’ll figure it out together.

Sure, I’ll show you how we work, and why.

Invites collaboration

Henrik Kniberg
WARNING

2 slides full of bullet points coming up

sorry...
Agile in Domain X requires a collaboration between people who understand Domain X, and people who understand Agile.

Step 1: Understand the context
- What do you do?
- Who are your stakeholders?
- What is a unit of work?
- What does Done mean?
- What does Success look like?
- Who is need to get things to Done?
- What do you want to improve, and why?
- How will you know if you’ve improved?

Step 2: Understand the tools
- What is Agile? Scrum? Kanban? XYZ?
- Which principles and practices are most applicable in your context?

Step 3: Get Buy-in
- Who needs to be involved to make the change happen?
- What’s in it for them?

Step 4: Start experimenting
- When in doubt, start by making work visible
- Find some early wins to build trust

Henrik Kniberg
Take-aways

- Agile is not new, and not going away
  - The word may go out of fashion, but the ideas are timeless
- Agile can be useful in just about any context, not just software
  - But Agile or <insert framework here> is only a means, never a goal
- Distinguish between Principles and Practices
  - Practices are more domain-specific and need to be adapted or replaced
- Copy & Paste & Evolve
  - No need to reinvent the wheel
- Use the appropriate language for the domain
  - Don’t unnecessarily alienate people with strange words
- Don’t inflict help on people
  - If they are happy with their current way of working, then don’t bother trying to change it.

Henrik Kniberg