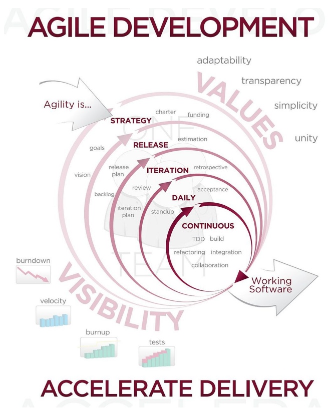
Refreshed Agile – Scrum’s foundation – between becoming an endangered mainstream buzzword and it’s evolution to THE management driver

ontheagilepath.net/2013/01/refreshed-agile-scrums-foundation-between-becoming-an-endangered-mainstream-buzzword-and-its-evolution-to-the-management-driver.html

By Sebastian Radics

Background



VersionOne, Inc. [CC-BY-SA-3.0 (http://creativecommons.org/licenses/by-sa/3.0)], via Wikimedia Commons

Today I face the word **AGILE** everywhere. Often it seems to me that its not clear what it means to be AGILE and to use agile development. With this post I try to contribute to refresh the meaning of agile.

- It starts with a broad definition of agile
- followed by why working agile is beneficial
- next with some agile myths uncovered
- combined with Scrum – as Scrum’s foundation
- a description of enablers for working agile
- it’s latest evolutions
- closed by a rich set of further readings.

Definition of AGILE

Lat. Agilis – nippy, flexible, agile

The Agile manifesto defines the base with the **Manifesto for Agile Software Development**. Let me shortly repeat it:

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 Agile manifesto is based on 12 principles of Agile Software.

Agile development **uses an empirical process based on empirical measurement** of the outcome produced in a defined interval of time. Its controlled by the manipulation of system constraints that:

- aims to have constant stabilization and optimization
- is done iteratively and incrementally.

It is **based on a pull system** (instead of a push system) – that means:

- **teams decide about its pace** – how much can be done in the next iteration. It pulls the next stories instead of getting it pushed through management.
- it is **respecting peoples own rhythm and capacity** – leading to more learning and creativity
- it **enables collective learning and collaboration** between team members as the teams decide how to work on the next functionality

Agile development:

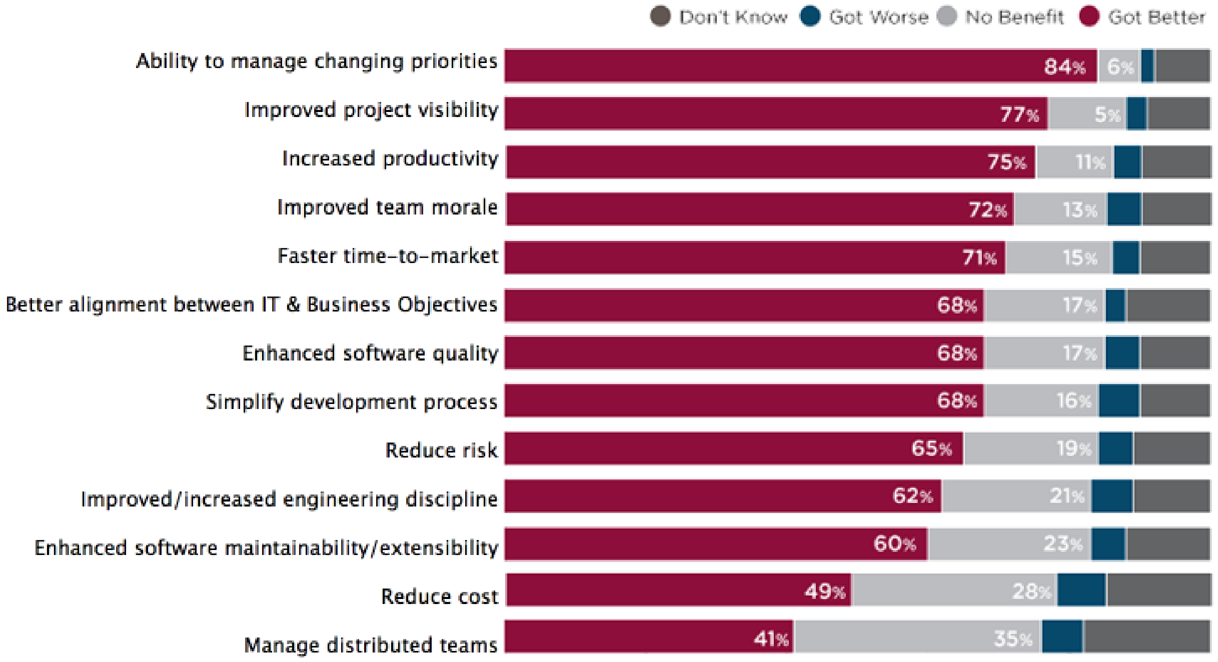
- **acknowledges that the knowledge and skills lie within the "worker"** and is not longer limited to decision makers.
- aims to **improve efficiency and effectivity** by:
  - focussing on value delivery and removing non valuable activities
  - working toward the business' goals – optimizing workflows and information flows
  - attention to symptoms of overburden to avoid later rework and dysfunctions
- recognizes that **work is fundamentally iterative learning**
- **shifts the management** model from being in charge **to being connected**.

It implies – **The discipline of making timely good decisions based on the shared understanding of a problem**.

It – **Expects performance and forces action if it does not occur**

Why working agile is beneficial

- Fosters managing changing priorities
- Improves project visibility
- Increases productivity
- Improves team morale
- Faster, accelerated time to market
- Better alignment of IT with business needs
- Enhanced software quality (maintainability/extensibility)
- Simplified development process
- Risk reduction
- Improves/Increases engineering discipline



Source: Version One survey from 07/2011 –
http://www.versionone.com/state_of_agile_development_survey/11/

Some agile myths uncovered

Agile is <u>not</u> the silver bullet

- you can fail as in any other project
- but you might fail sooner

Agile teams don’t do documentation

they eliminate waste and don’t write unnecessary documentation

Agile is anti planning

- in agile development you plan extensively on multiple levels
  - you create a release plan
  - an iteration plan
  - an plan you daily work in the daily standup
- you use different tools for planning – e.g. Scrum
  - the Product Backlog is your prioritized plan what to build
  - the Sprint Backlog is your plan what and how to build in your current iteration
  - the Sprint Burndown Chart and Release Burndown Chart provide you the progress overview
- all planning artefacts are highly visible

Agile is undisciplined

- in fact – agile development is extremely disciplined
  - you work with constant in time testing (using techniques like TDD and unit testing)
  - you seek for various ways of feedback – showing what and how you did something (Daily Scrum, Sprint Reviews, Retrospectives)
  - you optimize your continuous delivery with the aim to ship regularly
  - you update the plan regularly – every sprint and produce a high visibility allowing early intervention and adoption

Agile is anti-architecture

- you don't over-architecture and plan your architecture in time avoiding to much waste in the beginning
- this way you avoid over engineering
- KISS – as the driving principle (but still considering the constraints for your project – like scalability needs, security needs, performance needs)

Agile doesn’t scale

- like any other software process – scales not that great

Agile – the foundation of Scrum

The 12 principles of Agile Software build the foundation of Scrum. Each principle can be mapped to artefacts of Scrum. The following is a mapping of Scrum artefacts to the principles (headlines for principles are shortened):

Principle: Highest priority is to satisfy the customer

- Prioritized product backlog
- Sprint reviews – to enable fast feedback from customer to developers
- Incremental delivery – fast usage by customer and early feedback and learning
- Using user stories to describe business functionality in the domain language of the business expert

Principle: Welcome changing requirements

- No big up front planning, documentation and design
- Early feedback to spot need for change (through reviews and working software)
- Combined with agile development techniques known from XP (TDD, pair programming, clean code, refactorings)

Principle: Deliver frequently

- Stories as a small piece of business functionality
- Incremental and iterative development – Sprints & potential shippable products

Principle: Business people and developers work together daily

Product Owner involvement – as part of the Scrum team

Principle: Build around motivated individuals

- Empowered teams (decision on how to build it by the development team, decision what to build by the product owner as part of the Scrum team)
- Using retrospectives to clarify topics in time and welcome conflicts and their resolving

Principle: Face to face communication

- Daily Scrum meeting
- Sprint planning (including Product Owners, Customers and stakeholders)
- Retrospectives

Principle: Working software as the primary measure

- Sprint goals combined with sprint forecast and included user stories
- Velocity based on user stories delivered according to the Definition of Done

Principle: Sustainable development

- Incremental development that avoids having big bang releases and to heavy end of release workload
- Early feedback to discover bugs early and in smaller numbers

Principle: Technical excellence and good design

- Only scope is flexible. Time, cost and quality are fixed
- In combination with XP development techniques (TDD, refactorings, pair programming)
- Usage of the Definition of Done as quality gate

Principle: Self organizing teams

Scrum teams with a strong Scrum Master who ensures the self organization takes place

Principle: Reflection

Using the retrospective, sprint review and daily scrum

Enablers for AGILITY

Effective and efficient communication

Ask yourself: How long does it take that an information is passed from one employee to another including understanding? How many interactions should occur in a healthy environment?

- agile strives for collocated teams – not only onside but also enabling intensive communication between business and development. Teams working on a common project should be located near each other.
- you can expect a delay of >5 minutes if an employee needs to walk through your company
- collocations fosters the kind of information that is exchanged (assumption is replaced by face to face communication)

Consider communication modalities

- physical distance between each other
- 3D view – changes the way we communicate (and gets lost with remote communication)
- smell (e.g. we can smell fear)
- body movement
- touches
- sounds
- view distance

Much of these communication influencing parts get lost with remote communication

Choose proper information carriers

Strives for having the least effort to get an information

- see frequent changes
- use whiteboards and flipcharts
- place information on walls

Consider information cost aspects

- achieve cost reduction through osmotic communication – collocated you can hear project information with one ear (as a kind of ground noise)
- enable fast information detection and transformation (e.g. through pair programming or sitting beside each other – you already see if your team member starts searching for something and can offer help immediately)
- save costs by having questions raised instead of withholding information and building on assumptions (that could cause much higher costs at a later point in time)

Goal orientation

Goals provide the information in which direction to go. They present consequences of actions and enable course corrections. In additions goals provide the purpose for the direction.

7 underlaying principles

- **Interactive and direct communication** is the cheapest and fastes channel for information exchange
- **Unbalanced methodology is costly**
  - Consider what artefacts you really need for your development (what reports, documentation)
  - Define what is and scan for waste
  - Keep in mind the even little bureaucracy produces in sum high costs
- The **bigger the team size** the **heavier** the necessary **methodology**
- **Consider the level of criticality** to derive the necessary methodology and ceremonies – 4 levels
  - loss of comfort
  - loss of relative money (can be compensated by manual rework)
  - loss of relevant money (company can go bankrupt)
  - loss of life
- **Increasing feedback and communication** within the team reduces the need for intermediate results (like prototypes, reports, project plans)

  fostered by fast deliveries
- **Disciplin, skills and understanding** vs. process, formalities and documentation
  - process is not equal to discipline
  - formality is not equal to skill
  - documentation is not understanding
- For **activities without bottlenecks** you can dispense efficiency
  - team members that are bottlenecks need to work as efficient as possible
  - do everything to improve speed of finishing bottleneck activities
    - Can someone else take it?
    - Do more people help?
    - Can you use better/other tools?

Consequences

- Make teams better and not bigger
- Adding people to projects is costly

- Lightweight methodologies are better until the problem gets to big
- Stretch methodologies to fit – start with less and increase if necessary

Consider sweet spots

Sweet spots refer to the optimum you can reach for being agile.

- 2-8 people in one room (according to science the sweet spot is 5)
- having experts on side (fast feedback and more ideas)
- having automated regression testing
- incremental development
- having experienced developers in the team (are 2-10 time more productive)

Latest evolutions

Kent Beck provided some nice additions to the existing Agile manifesto related to working with start-ups but also relevant for standard projects.

Team vision and discipline over Individuals and interactions (over processes and tools)

Validated learning over Working software (over comprehensive documentation)

Customer discovery over Customer collaboration (over contract negotiation)

Initiating change over 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.

Watch live video from Startup Lessons Learned on Justin.tv

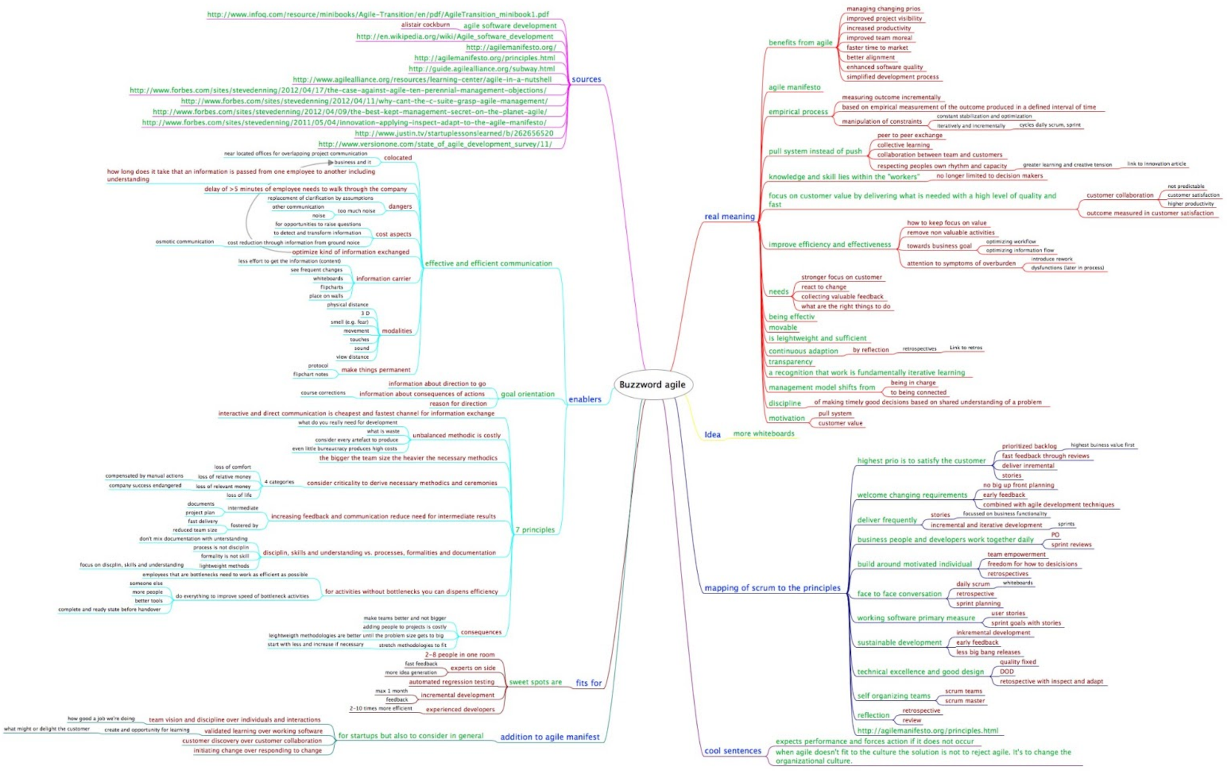
Further readings

Before you dive into the really interesting sources – what do you think about agile? Do you have additional points to add? I'm interested in your opinion!

- Agile transition minibook
- Agile Software Development: The Cooperative Game
- Agile software development on Wikipedia
- The Agile Manifesto
- 12 Principles for the Agile Manifesto
- Check your Agility with this nice Agility Guide
- Watch the Agile in a Nutshell presentation
- The Case Against Agile: Ten Perennial Management Objections
- Why Can't The C-Suite Grasp Agile Management?
- The Best-Kept Management Secret On The Planet: Agile
- Innovation: Applying "Inspect & Adapt" To The Agile Manifesto
- State of Agile Development Survey Results by VersionOne

Overview

Download MindMap (Freeplane format)


Buzzword agile
sources
http://www.infoq.com/resource/minibooks/Agile-Transition/en/pdf/AgileTransition_minibook1.pdf
agile software development
alistair cockburn
http://en.wikipedia.org/wiki/Agile_software_development
http://agilemanifesto.org/
http://agilemanifesto.org/principles.html
http://guide.agilealliance.org/subway.html
http://www.agilealliance.org/resources/learning-center/agile-in-a-nutshell
http://www.forbes.com/sites/stevedenning/2012/04/17/the-case-against-agile-ten-perennial-management-objections/
http://www.forbes.com/sites/stevedenning/2012/04/11/why-cant-the-c-suite-grasp-agile-management/
http://www.forbes.com/sites/stevedenning/2012/04/09/the-best-kept-management-secret-on-the-planet-agile/
http://www.forbes.com/sites/stevedenning/2011/05/04/innovation-applying-inspect-adapt-to-the-agile-manifesto/
http://www.justin.tv/startuplessonslearned/b/262656520
http://www.versionone.com/state_of_agile_development_survey/11/
enablers
effective and efficient communication
colocated
near located offices for overlapping project communication
business and it
how long does it take that an information is passed from one employee to another including understanding
delay of >5 minutes of employee needs to walk through the company
dangers
replacement of clarification by assumptions
too much noise
other communication
noise
cost aspects
for opportunities to raise questions
to detect and transform information
cost reduction through information from ground noice
osmotic communication
optimize kind of information exchanged
less effort to get the information (content)
information carrier
see frequent changes
whiteboards
flipcharts
place on walls
modalities
physical distance
3 D
smell (e.g. fear)
movement
touches
sound
view distance
make things permanent
protocol
flipchart notes
goal orientation
information about direction to go
information about consequences of actions
course corrections
reason for direction
7 principles
interactive and direct communication is cheapest and fastest channel for information exchange
unbalanced methodic is costly
what do you really need for development
what is waste
consider every artefact to produce
even little bureaucracy produces high costs
the bigger the team size the heavier the necessary methodics
consider criticality to derive necessary methodics and ceremonies
4 categories
loss of comfort
loss of relative money
compensated by manual actions
loss of relevant money
company success endangered
loss of life
increasing feedback and communication reduce need for intermediate results
intermediate
documents
project plan
fostered by
fast delivery
reduced team size
disciplin, skills and understanding vs. processes, formalities and documentation
don't mix documentation with unterstanding
process is not disciplin
formality is not skill
lightweight methods
focus on discplin, skills and understanding
for activities without bottlenecks you can dispens efficiency
employees that are bottlenecks need to work as efficient as possible
do everything to improve speed of bottleneck activities
someone else
more people
better tools
complete and ready state before handover
consequences
make teams better and not bigger
adding people to projects is costly
leightweigth methodologies are better until the problem size gets to big
stretch methodologies to fit
start with less and increase if necessary
fits for
sweet spots are
2-8 people in one room
experts on side
fast feedback
more idea generation
automated regression testing
incremental development
max 1 month
feedback
experienced developers
2-10 times more efficient
addition to agile manifest
for startups but also to consider in general
team vision and discipline over individuals and interactions
how good a job we're doing
validated learning over working software
create and opportunity for learning
what might or delight the customer
customer discovery over customer collaboration
initiating change over responding to change
real meaning
benefits from agile
managing changing prios
improved project visibility
increased productivity
improved team moreal
faster time to market
better alignment
enhanced software quality
simplified development process
agile manifesto
empirical process
measuring outcome incrementally
based on empirical measurement of the outcome produced in a defined interval of time
manipulation of constraints
constant stabilization and optimization
iteratively and incrementally
cycles daily scrum, sprint
pull system instead of push
peer to peer exchange
collective learning
collaboration between team and customers
respecting peoples own rhythm and capacity
greater learning and creative tension
link to innovation article
knowledge and skill lies within the "workers"
no longer limited to decision makers
focus on customer value by delivering what is needed with a high level of quality and fast
customer collaboration
not predictable
customer satisfaction
higher productivity
outcome measured in customer satisfaction
improve efficiency and effectiveness
how to keep focus on value
remove non valuable activities
towards business goal
optimizing workflow
optimizing information flow
attention to symptoms of overburden
introduce rework
dysfunctions (later in process)
needs
stronger focus on customer
react to change
collecting valuable feedback
what are the right things to do
being effectiv
movable
is leightweight and sufficient
continuous adaption
by reflection
retrospectives
Link to retros
transparency
a recognition that work is fundamentally iterative learning
management model shifts from
being in charge
to being connected
discipline
of making timely good decisions based on shared understanding of a problem
motivation
pull system
customer value
Idea
more whiteboards
mapping of scrum to the principles
highest prio is to satisfy the customer
prioritized backlog
highest buiness value first
fast feedback through reviews
deliver inremental
stories
welcome changing requirements
no big up front planning
early feedback
combined with agile development techniques
deliver frequently
stories
focussed on business functionality
incremental and iterative development
sprints
business people and developers work together daily
PO
sprint reviews
build around motivated individual
team empowerment
freedom for how to desicisions
retrospectives
face to face conversation
daily scrum
whiteboards
retrospective
sprint planning
working software primary measure
user stories
sprint goals with stories
sustainable development
inkremental development
early feedback
less big bang releases
technical excellence and good design
quality fixed
DOD
retospective with inspect and adapt
self organizing teams
scrum teams
scrum master
reflection
retrospective
review
http://agilemanifesto.org/principles.html
cool sentences
expects performance and forces action if it does not occur
when agile doesn't fit to the culture the solution is not to reject agile. It's to change the organizational culture.