December 2011

Scrum.org | Whitepapers

The Blending Philosophies of Lean and Agile

Gunther Verheyen

Some management or governance philosophies should not be mixed. Because the resulting mix will be a blurry amalgam and the unique flavor of the ingredients will get lost, as well as their benefits. In general, not only the flavor and the envisioned benefits get lost, the total 'product' may be well less effective than the sum would suggest.

Lean is a management or organizational model that thrives on a typical mindset, with powerful but distinct fundaments, principles and thinking. Does the assumption that such strategies are best not mixed however imply that only a 'clean' implementation of Lean will deliver good results? Should we conclude that there are no combinations imaginable?

I believe that **Agile** thinking, principles and methods are more than just not at odds with Lean. I see not only much common ground to Lean and Agile, I am even convinced that the combination of Lean management principles with Agile product development thinking, as a total outcome, will result in an even more powerful mix. I believe that Lean and Agile are truly *blending* philosophies.

In this paper I highlight some major aspects of the distinct views of Lean and Agile, and indicate the similar grounds to them. But I have also included the **Scrum** perspective to Agile to demonstrate how the tangible, yet open framework of Scrum aligns and blends the underlying thinking of Agile and Lean. Because Lean and Scrum are similar houses, only different materials.

Enjoy reading, and... keep Scrumming.

Kind regards

Gunther Verheyen

Professional Scrum Trainer for Scrum.org
Agile/Scrum Leader at Capgemini

22 December 2011

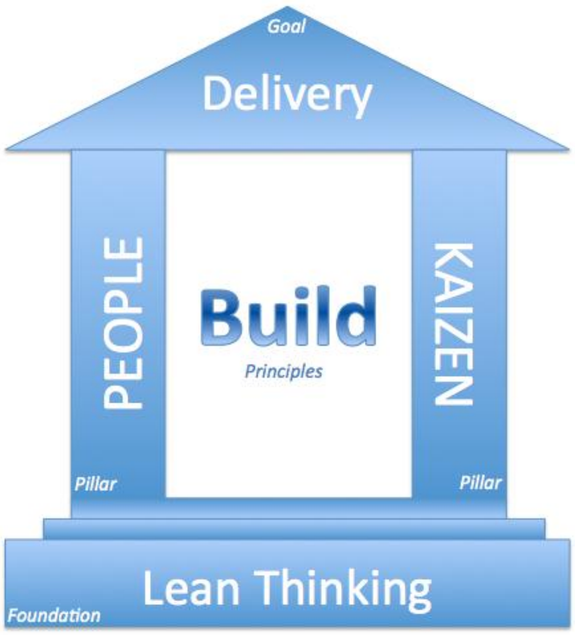
Part 1: Lean Thinking

Lean is a set of thinking tools, a collection of interwoven principles that educate, motivate, value and coach people to continuously optimize their work and the way they work. The principles form a toolkit of levers to install a system within which people are enabled to faster create better products in a sustainable and respectful way. It's a system that rewards people for doing the best they can.

But unfortunately Lean is often, and far too much, taken for or bluntly limited to just... eliminate waste. Just picking out that one element from the toolbox is already an undesirable over-focus on just one aspect of Lean, instead of looking at the whole. But it gets even worse when the principle itself is broken, when 'elimination' is applied to *people* and not as a mean to *improve*. The highly popular management sport of 'cost cutting' tends to twist this important Lean practice into denominating people's work as 'overhead', i.e. non-valuable. The devious idea is then to render the people doing that work as waste and... disposable.

From that popular misconception and it's all too limited perspective on Lean, it is a long journey to build up an understanding that Lean is primarily about respecting **PEOPLE** in order to optimize **VALUE** and **QUALITY**. It is more about creating a *context* in which people can prosper in order to perform, than about continuously over-stressing the need for results and performance. It invokes the difficult exercise of letting go of 'command and control', of big boss behavior, micro-management, over-allocating and nano-assignments. It is a long journey from this misconception to a deep understanding of the underlying ideas and principles of Lean, to looking beyond the formal practices. The goal is to build up a view on the whole and initiate the thinking that there is no definite goal. It is about people continuously reflecting on their daily work and self-improving. The 'end' state might even include the elimination of one's own job and look for another value-adding place in a Lean ecosystem.

The primary objective of embarking on the little Lean journey of this paper is to highlight the mindset and culture that enable and encourage people.



In the rest of this first part I will highlight no more than some major aspects of Lean. It will even be done in a somewhat condensed way. And still it will hopefully reveal the deeper impact of Lean, as well as the difference in the way that enterprises tend to work today or are being taught on how they 'should' operate.

I will present a minimal set of subjects that should be considered, included and answered when initiating, or even considering, a transition to Lean, whether it is Lean for production or a adopting a Lean management style. And it will become clear how a respectful implementation of Lean can produce remarkable improvements. And I would expect that to include even systematically preventing and eliminating waste.

People

The corner stone of any system that claims to be Lean are the **People**. And with 'people' I refer to every possible actor in the whole ecosystem of the Lean product development/build system: customers, workers, teams, suppliers, managers, internal, external.

All of these people contribute in their own way and by their own means to building or developing Product. They collaborate in a multi-skilled way to avoid hand-overs, delays and waiting time. They are empowered to take decisions. They have room to focus on knowledge gathering and constant learning. Managers act as **TEACHERS** with a **GO SEE** commitment of work-floor presence to promote the Lean thinking system, to help people understand how to reflect on their work, the products and how to build better products. The whole system embodies the spirit of **KAIZEN**, the attitude (!) of continuously minding the process, the product and possible improvements. Every member of the whole system can **STOP THE LINE** if a problem occurs. The root of the problem will be immediately detected and countermeasures are proposed or installed. It all serves to assure that **QUALITY** is built-in into the Product, acknowledging that quality cannot be simply added to a Product after it has been built.

A good hands-on practice for root-cause analysis when an error, a defect or a flaw occurs is the **5 WHYS**. Keep looking for the 'why' of every answer, the next-level cause. Think it through until the bottom is reached. *And it may require more than 5 attempts and the result might be a tree of branched answers...*

It has been said, everyone involved in the value chain works in an *integrated* way. This is also shown in the relationships with suppliers and external partners. These relationships are not based upon the traditional approach of large volume purchases, big negotiation rounds and pressuring one another. Those are procedures that are not only highly disrespectful with regards to the commitment and creativity of the involved people. It also typically ends with at least one strangled party. And that party will extend this price pressure to internal people, creating a highly undesirable and discouraging working environment that prevents ideas, improvement, creativity and learning.

It's all about building relationships on the *sharing* of profit (and risk!). Lean contracts incorporate mutual growth. It's about thinking in terms of communities and communities of practice, and exchanging information and knowledge not only cross-departmental, but even cross-company.

Waste

Before entering the subject of (eliminate) **WASTE**, let's mention that *avoiding* waste, via continuous improvement and small step optimizations, is a preferred option. *At least it emphasizes the subtle difference in timing on when to act...*

Furthermore, remember that 'Waste' refers to process steps, not to the disposal of people.

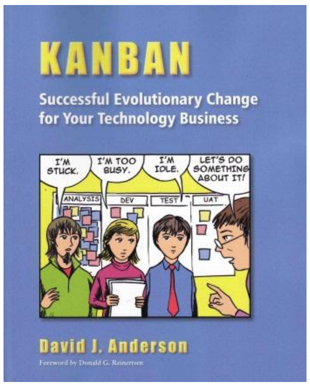
Obviously, no matter how much attention is paid to avoiding it, waste can (and will!) creep in. The **KAIZEN** spirit should drive all people to be committed, aware and critical in their daily work. It must have become a natural reflex. On top of that basic attitude, a good tool or practice to identify structural waste is **Value Stream Mapping**. All steps and phases in the process of going from 'idea' to 'build' are set out on a timeline. Activities may be labeled as valuable or non-value adding, but possibly also as necessary although not directly value-adding. The *Value Ratio* can be calculated as the ratio of time spent on Value-adding activities versus Waste activities. It's a figure that may serve as a baseline against which improvement can be measured. But, like in all improvement activities, there is no definite end goal, no final state. The improvement itself is the goal.

Inventory, WIP and Flow

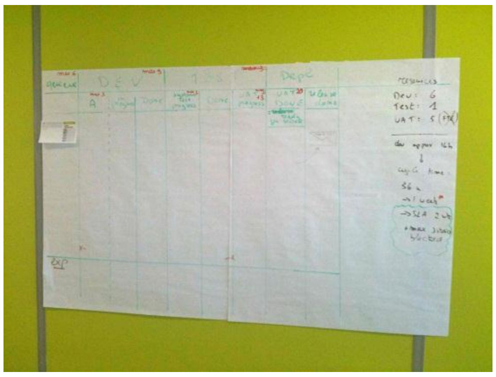
Lean strives for continuity and flow. Overproduction of materials disrupts flow and may delay the discovery of quality issues. But it is also disrespectful as it forces people to do work that may actually never be used. Lean says to limit 'Work in Progress' (and costly inventory) by producing only materials when there is a **Pull** signal from the next steps in the process in a 'Just in Time' mode. In an ideal world there wouldn't even be this type of signal.

But in the meantime, a **Kanban** is a physical signal card in manufacturing systems that is attached to an inventory of parts. It is linked to a level of stock. New parts are only produced when enough materials have been used and the signal card appears.

The Kanban software method as promoted by the Limited WIP Society revolves around the practice of visualizing and optimizing flow of software work. It uses Kanban cards to hold user requirements or software demands. The state of the cards is observed to optimize the regular production of working software. The physical cards are placed on a Kanban board. The board visualizes the process by showing the various states of the work items, with limits to the work-in-progress for every state. The limits prevent pushing work down the process and disrupting the flow. Work can only be taken in on a pull base.


KANBAN
Successful Evolutionary Change for Your Technology Business
I'M STUCK.
I'M TOO BUSY.
I'M IDLE.
LET'S DO SOMETHING ABOUT IT!
ANALYSIS
DEV
TEST
UAT
David J. Anderson
Foreword by Donald G. Reinertsen

Metrics are collected upon the flow of the work and the changes on the board. They are processed and visualized in diagrams and charts, like a *Cumulative Flow Diagram*. *Cycle Time* is the total time required to complete a work item after entering the process. It is tracked to guard continuity and for predictability reasons, as it is the average pass-through time for a request. A **Whole Team** optimizes flow and cycle time by removing piled up work first.

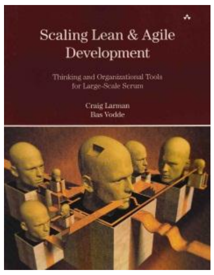


Design and tuning of a Kanban board incorporates engineering and adaptation of the process. It may include distinguishing work types or allow certain requirements to travel faster across the board.

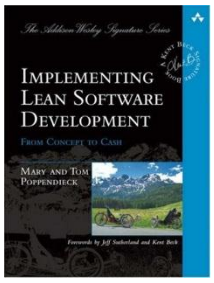
References

There is actually not one definite, full-blown, one-size-fits-all, unified Lean process with predefined and prescribed phases, roles, definitions, artifacts, deliverables, etc. A Lean process must be designed upon the principles and the thinking, and constantly tuned to the actual situation. It's about the power to adapt.

If there is no standard Lean process template to be copied, how should you start then? Well, coaching and training help. Getting informed is a good start.


Scaling Lean & Agile Development
Thinking and Organizational Tools for Large-Scale Scrum
Craig Larman
Bas Vodde

I personally advise reading the Lean Primer PDF, by Craig Larman and Bas Vodde, to have a quick, but still profound overview of Lean. In this paper the authors stick to the *thinking* without overloading the reader with details on business or industrial domains, areas and methods. Their paper always proves to be a great summary of the basics and has been an important source of inspiration for above descriptions. If you can spend more time, do check out their books on the subject.


The Addison-Wesley Signature Series
Implementing Lean Software Development
From Concept to Cash
Mary and Tom Poppendieck

With regards to software development, Mary and Tom Poppendieck have done a tremendous job in translating and applying the classic Lean manufacturing aspects (the Toyota way of Lean), definitions and concepts from industrial production processes to building software products.

And they often refer to Scrum as a quite extraordinary implementation of Lean thinking in software product development.

Part 2: Agile Spirit

In general, companies refer to *organizational* problems when expressing a desire for 'Lean'. If they want to become 'Agile' on the other hand, they are most likely referring to *software* development. Unfortunately, a majority of the desiring managers seem to hope for some magical, off-the-shelf (silver bullet?) Agile solution to all of their problems.

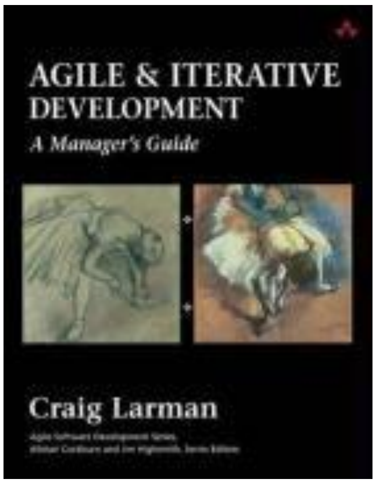
Responding that *"Agile in itself does not exist"* may be a good start to get their attention. To be continued by explaining that Agile, very similar to Lean, is in the first place a way of *thinking* and that a deep transformation to Agile is required for real and lasting benefits. In a software development context this does imply going beyond the borders of the software development departments. An entire organization will prosper from adopting the Agile mindset of *empiricism* and being *adaptive.* And the frequent **INSPECT & ADAPT** cycles of Agile will challenge large parts of the organization anyhow.

It might take some time for co-workers to see that the continuous learning that comes with Agile will help them do a great job amidst turbulent enterprise, business and market circumstances. Part of their uncertainty can be taken away as the active *Business Collaboration* optimizes the **BUSINESS VALUE** of the incremental outcomes.

Finally, Agile assumes the acknowledgement of software development as a creative and complex activity by and for **People**. The Agile views and approach at last allow us to finally stop trying to predict the unpredictable, as its practices incorporate dealing with answers, solutions and competing ideas that emerge while building software.

The Origins of Agile

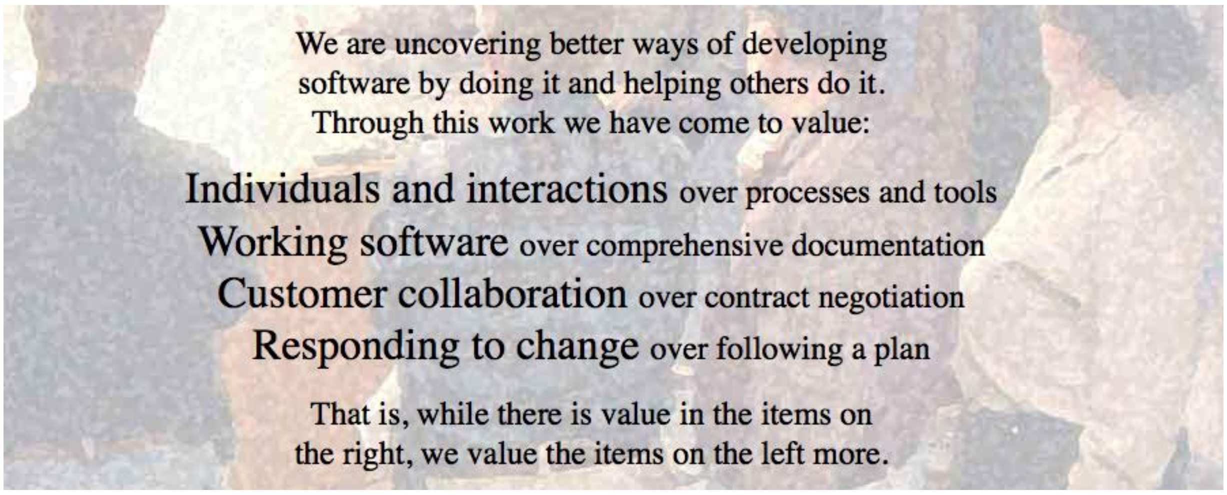
An evolutionary approach to software development is not new. Craig Larman has extensively described the historical predecessors of Agile in his book "Agile & Iterative Development - A manager's guide". But the label 'Agile' itself dates from early 2001, when 17 software development leaders gathered at the Snowbird ski resort in Utah. They assembled to discuss their views on software development in times that there was a tendency to replace failing traditional (waterfall) approaches with equally disastrous heavy-weight RUP implementations. While at that time these leaders were doing things differently with Scrum, eXtreme Programming, Crystal, DSDM, Adaptive Software Development, Feature Driven Development, etc.



The gathering resulted in assigning the label 'Agile' to the common principles, beliefs and thinking of these leaders and their methods. They were published as the Agile Manifesto. The most important conclusion to be drawn from this understanding is that Agile is just not one, single method. It is the common denominator of a number of methods.

Definition of Agile?

The Agile Manifesto certainly helps in understanding the Agile thinking:



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.

It is also worthwhile looking at the 12 principles for a deeper understanding!

But I like to describe 'Agile' as the following, somewhat more tangible, *key characteristics* of the portfolio of Agile methods:

Business & People. Agile is not driven by a predictive plan upon requirements that are created and thrown over the wall by requirements gatherers, but through *Business collaboration* with business people regularly expressing their functional intents and expectations from an overall Product Vision.

People are respected for their creativity, intelligence and self-organizing capabilities to understand and resolve a problem without overloading them with tons of ceremony that replace proper thinking;

Facilitation. Agile Teams are *facilitated* by servant-leadership. Overall objectives are set and a context for self-management and *subtle control* is created; rather than individual team members being assigned on a daily base to executable micro-tasks in a command-and-control style;

Iterative-Incremental. Agile processes are not free-play approaches. Agile processes are defined and even require high discipline in building products piece by piece ('incremental'), while frequently revisiting the built pieces and the product ('iterative') to expand, improve and adapt while assuring overall integrity upon technical excellence;

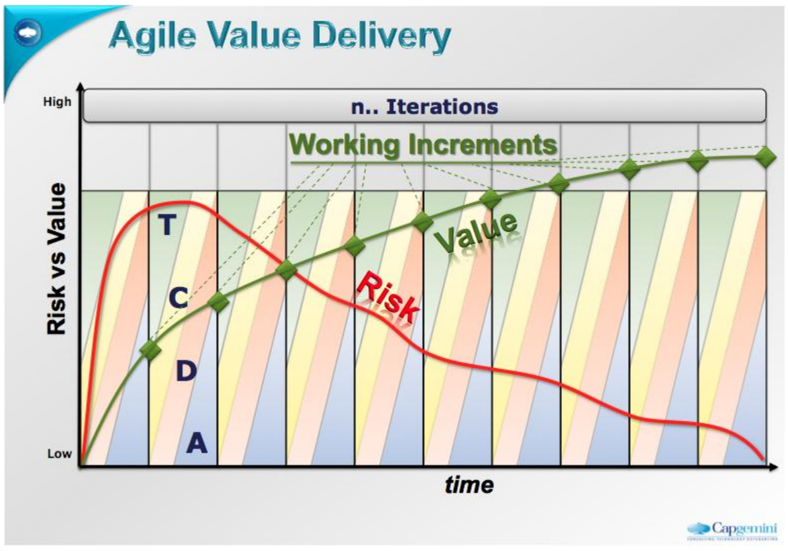
Success and **Progress** cannot be measured upon mere compliancy with predictive plans and milestones, documents, hand-overs, signatures, approvals or other ceremonies. Success and progress can only be determined by the actual *Business Value* of the *Working Software* that is delivered at the end of every iteration;

Change. New or evolving opinions and priorities form the living heart of Agile. Agile thrives upon **EMERGING REQUIREMENTS**. This is not disruptive because it forms a natural part of the process and is not excluded from it nor expelled to the ceremonial outskirts of development. So, what we used to know as 'change' has... evaporated.

The Iterative-Incremental Continuum

The heartbeat of an Agile approach to software development is the iterative-incremental continuum. Time is sliced into time-boxed iterations, periods having a fixed start and end date. Every time-boxed iteration results in an increment of working software. Value is continuously increased across iterations and risk is controlled by consecutively producing working increments upon defined engineering standards.

Figure 1: Agile Value Delivery



In Figure 1, it is essential to understand the __*business*__ *nature of 'risk'. Usually, in an IT context, risk is defined as something technical. (Will the system hold? Be performant? Scalable?) But the final goal of Agile is to provide more satisfaction to end-users and customers. Software must be useful. Software being usable is just the basic beginning, and needs to be assured by cross-skilled development Teams.*

'Risk' is here therefore defined as the business risk of not being able to capitalize on unforeseen opportunities, of not releasing fast enough, of being liable to customer dissatisfaction by releasing untested software, of releasing features that are not what users expect or appreciate, of lagging behind with regards to the competition.

To produce shippable Increments, Agile still requires the 'normal' IT activities (here conceptually represented as _A_nalysis, _D_esign, _C_oding and _T_esting). Agile does however require these activities to be fundamentally re-organized. All of these disciplines, required to deliver fully working software from an end-user perspective, are to be performed in a non-linear, incremental way but in parallel and on a daily basis.

The goal of such integrated, cross-functional approach is **BUILT-IN QUALITY**. It aims at the prevention of defects, instead of post-development bug-hunting and overdue finding of non-satisfactory quality.

Blending Philosophies

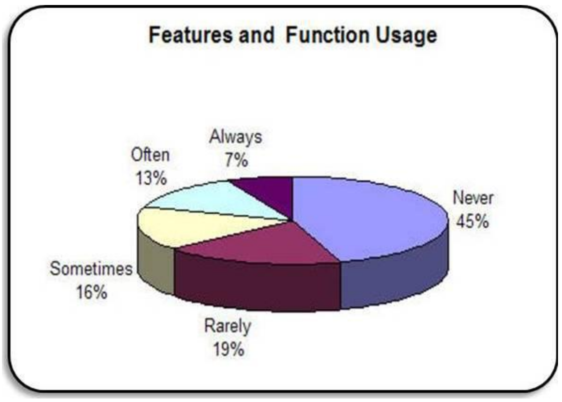
With our understanding of Lean thinking and the Agile spirit we can elaborate upon the obvious similarities. Agile has distinct practices that not only match the main Lean principles extremely well, but even form a very concrete implementation of them for software development.

There are no less than 3 major approaches in Agile to (prevent and) **ELIMINATE WASTE**;

Potentially unused inventories form a liability, and not an asset. Therefore detailed requirements, hard-coded plans, designs, etc. are not produced upfront. Because the chances are quite high that they will not be implemented, the exact expectations may vary or experience will result in insights that open up better ways of implementations. Only the next, highest ordered work is detailed appropriately as this is the work that will be worked on next. And even then a Team will **PULL** in only the amount of highest ordered work they deem feasible for an iteration, and start building it upon progressive learning and continuous improvement, even on a daily base.

Partially done work, another important type of waste, won't pile up as the goal of an iteration is to produce a working increment of product. No undone work is included at the end of an iteration. The overall **KAIZEN** thinking, and its explicit daily *Inspect & Adapt* implementation, mostly a stand-up meeting, prevents taking up new work while undone work remains in the iteration.

Active Business Collaboration prevents the production of unwanted or invaluable requirements. Only 20% of product functions are regularly used. Unused or underused functions thus represent an enormous waste of effort and budget. Keeping a focus on 'wanted' requirements saves not only development budget, it also assures that future maintenance and support costs can be kept much lower. And the iterative-incremental process allows Business to continuously adapt to new Value expectations.



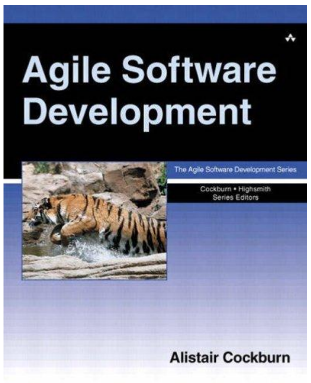
A **SHARED VISUAL WORKSPACE** facilitates fast decisions, high interaction and short communication lines. This workspace also contains *Information Radiators* as implementation of **VISUAL MANAGEMENT**. Task Board, Team definitions, agreements and process artifacts like Backlogs and progress trends are made visible within the shared workspace; preferably even posted on the room walls. Agile Teams publish this information to *share* it and use it to inspect and adapt. This transparency, unfortunately, can be abused. Regularly, in a situation where progress is less than expected or hoped for, for very legitimate reasons, tendencies will be to revert to old-style bashing. This undermines the foundation to being Agile. A 'popular' instruction is to enforce overtime. This is an infringement on **SUSTAINABLE PACE**, it destroys Agile metrics and it undercuts product quality.

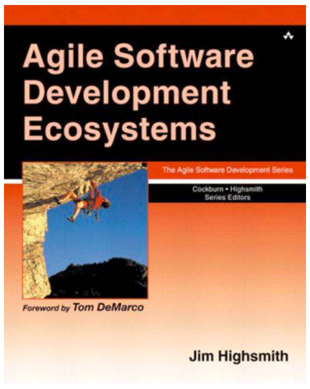
In order to **DELIVER FAST** every iteration is time-boxed and at the end of every iteration only usable, working software is shown, holding the option to launch it 'as is' into production. The software is reviewed with all stakeholders in order to gather feedback, remarks, improvements and enhancements. On top of this product inspection, an Agile Team will hold a retrospective meeting to look for improvements that can be introduced in the next iteration. Having this formal event at the end of an iteration may however not impede a **STOP THE LINE** call of each Team member *during* the iteration, not even waiting until the next stand-up meeting.

Agile **OPTIMIZES THE WHOLE** by demanding that Business expresses and orders work, and takes active part for clarifications and functional trade-offs during the technical build process. But it does require that all skills are onsite available to turn ideas, options and requirements into working software in one iteration. This automatically optimizes the *Value Stream* because traditional waiting activities like hand-overs and external decisions are eliminated. There are no macro hand-overs that are typical to a sequential organization of work with large blocks of specialized work packages. But there are also no micro hand-overs given the collective accountability of the Team. And, finally, the cross-functional eco-system includes a process master/mentor who will act as a **MANAGER-TEACHER**, i.e. manages the process (not the people), teaches the people and facilitates the Team by removing *impediments*.

References

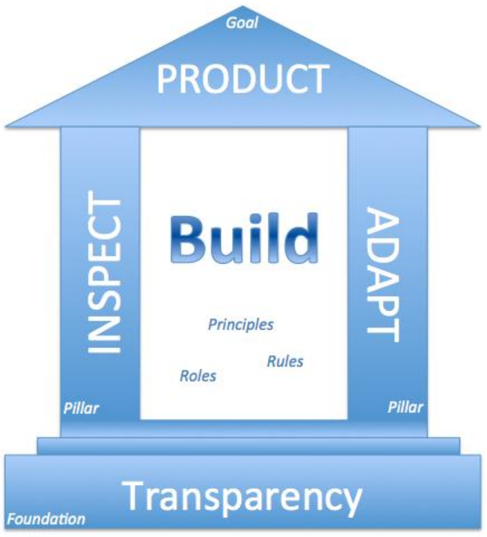
There seems to be no beginning and no end to the list of works on Agile, not to speak of method-specific books. I have found however 2 works in particular to be good and broad introductions in Agile in general with short summaries for different methods. These works are of biblical size, but also of comparable value and have been written by Alistair Cockburn and Jim Highsmith.


Agile Software Development
The Agile Software Development Series
Cockburn • Highsmith Series Editors
Alistair Cockburn


Agile Software Development Ecosystems
The Agile Software Development Series
Cockburn • Highsmith Series Editors
Foreword by Tom DeMarco
Jim Highsmith

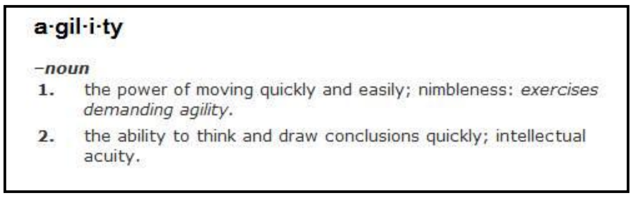
Part 3: The Scrum perspective to Agile

Scrum is an Agile framework for the development of complex products in complex circumstances. The framework, as described in the Scrum Guide by Scrum co-founders Ken Schwaber and Jeff Sutherland, describes the elementary roles, rules and principles of Scrum. But it doesn't prescribe the strategies to apply it.



It is obviously very important to read these game instructions to Scrum and understand how the game is played to fully enjoy the playing experience. The main game instructions demonstrate how Scrum implements the Agile thinking and principles, and help you judge whether it is a game suitable for you.

The goal of Scrum is to optimize and control the delivery of *Valuable* Software in turbulent enterprise, business and market circumstances. Control is achieved via frequent **INSPECT & ADAPT** cycles and the ability of the players to learn and improve, to become better players. Active collaboration with real business players is imperative in growing to **Business Agility**, and increasing the organization's competitiveness and market position.



a·gil·i·ty

–noun

1. the power of moving quickly and easily; nimbleness: *exercises demanding agility.*
2. the ability to think and draw conclusions quickly; intellectual acuity.

Scrum requires great discipline from the players, but still leaves much room for personal creativity. The rules of the game are based upon respect for the people-players through a subtle and balanced separation of responsibilities. And it is shown over and over again that respecting the rules of the game, not taking shortcuts on rules and roles or short-circuiting the empirical grounds of the game, deliver the most joys and highest benefits.

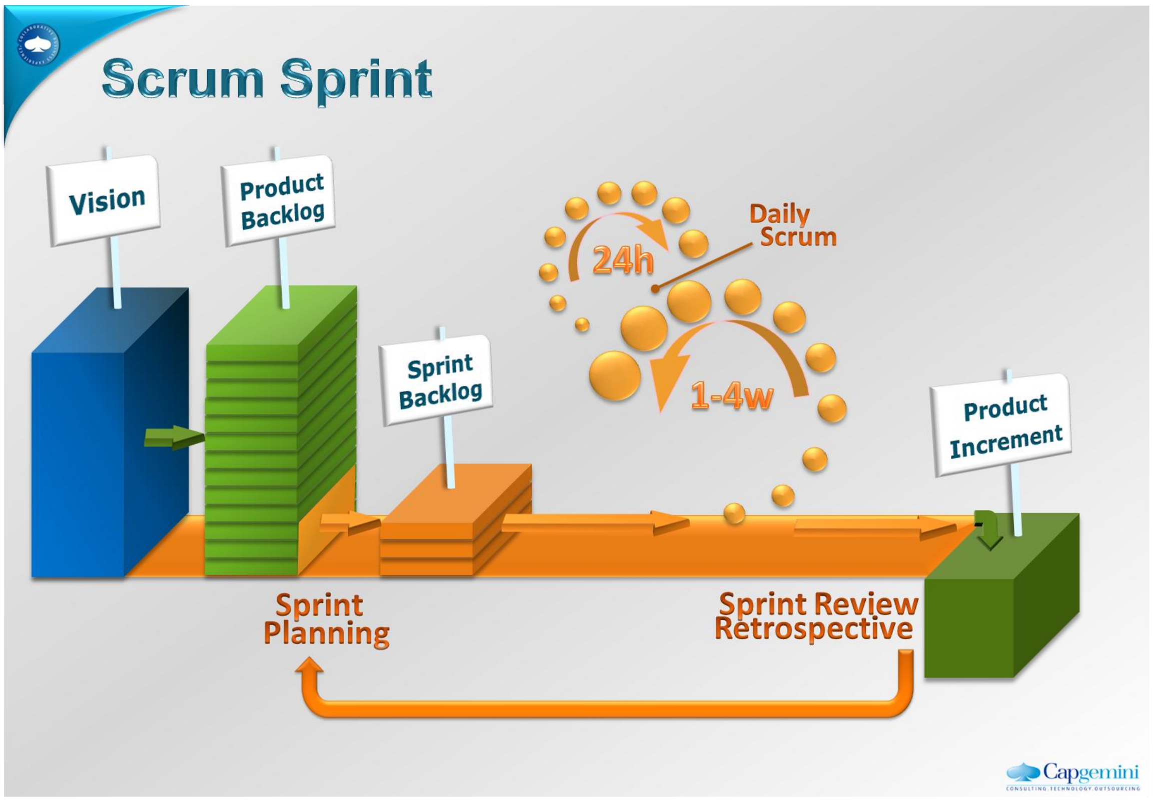
Players

Business is present on a Scrum Team as the **PRODUCT OWNER**, a one-person player role. The Product Owner represents all stakeholders, internal and external, to the **DEVELOPMENT TEAM**, a multi-person player role. Although a Product Owner has various strategic product management tasks outside of the Scrum Team, it is important that the Product Owner actively engages with the Development Team. The Product Owner is expected to structure and order the business expectations, functional requirements, non-functionals and other value-adding Product aspects. The Scrum Team gathers these in a list known as the **PRODUCT BACKLOG**. The Product Owner also manages the game budget to squeeze as much Value as possible out of it. The **SCRUM MASTER**, a one-person player role, facilitates the Scrum Team during the game by blasting any impediments that block the Team. The Scrum Master also teaches the Scrum Team and the organization in understanding, respecting and playing the game, making sure the rules of the game are well understood and inducing the continual desire to become better players.

Sprints

Time-boxed iterations in the game of Scrum are called **SPRINTS**. Sprints serve as containers that allow the Development Team to focus on achieving the next game level, the *Sprint Goal*, with minimal outside disruptions. The Development Team elaborates the selected Product Backlog Items (PBIs) it deems feasible for a Sprint in a list of actionable work, the **SPRINT BACKLOG**. To guarantee quality, the Development Team defines a set of 'Engineering Standards' and is empowered for technical implementation. To guarantee the "fitness for purpose" of the increment of software at the end of each Sprint, the Team is accountable for delivering work that complies with a 'Definition of Done' that needs to incorporate organizational standards.

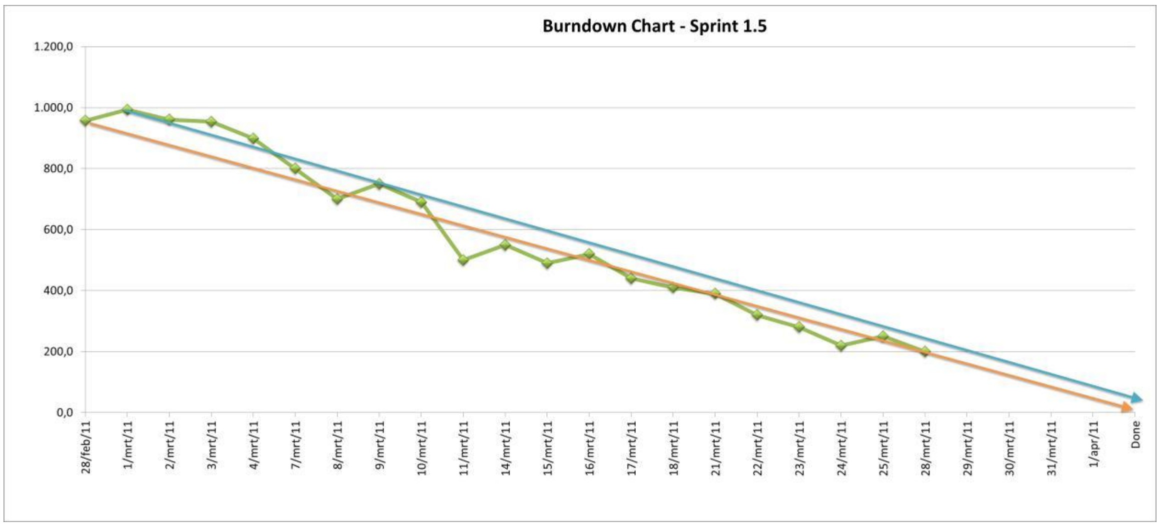
Sprint length is kept equal over a stage of the game, of multiple Sprints, for reasons of cadence. It is the heartbeat of development and helps the team in collecting effort calibration metrics like Velocity. The exact Sprint length should not only be chosen upon the team's need to be protected by a container, and on how long a Team can work without consulting the stakeholders but be right-sized to capitalize on emerging business opportunities. A Sprint typically takes 1-4 weeks.



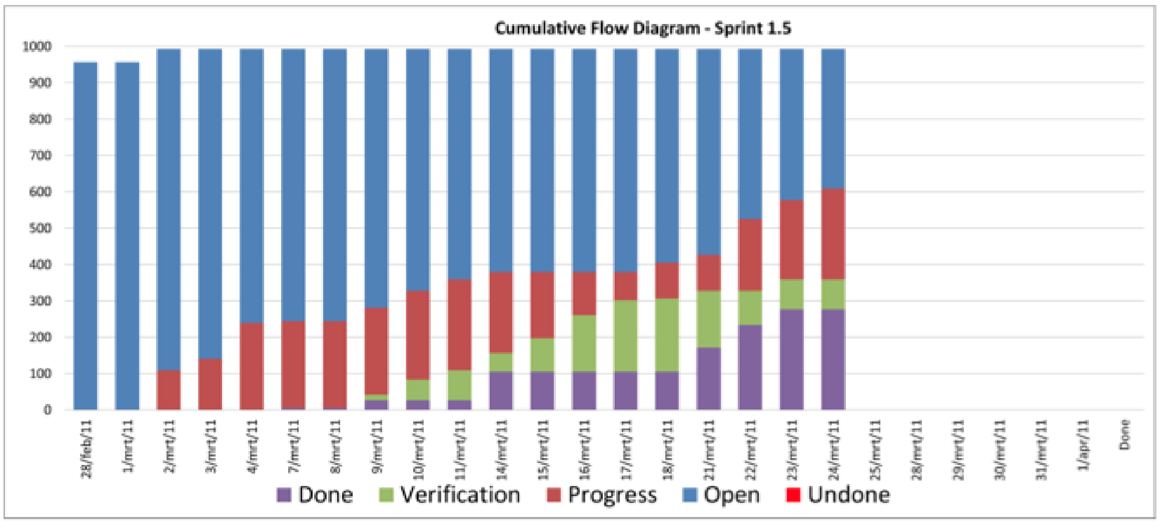
The Nature of the Game

The Development Team inspects itself, and adapts by sharing right-time planning information in a short, 15 minutes, daily meeting called the **DAILY SCRUM**. The Team acts as a self-organizing unit in performing all development activities required to turn items from the Product Backlog into 'Done' software. And 'development' applies to test cases, TDD tests, code, documentation, integration work, etc.

Overall progress of work, at Product (Release) and at Sprint level, is tracked and visualized, in order to create a **PROGRESS TREND** that adds predictability to uncertainty. The Scrum classic to do this is a *Burn-down chart*, i.e. a graph showing the overall downward evolution of remaining work. In order to continuously measure and adapt to reality and achieve the best predictability possible upon complexity, the remaining work is re-estimated regularly. Within a Sprint it is done at a daily frequency so the Sprint Backlog always represents the most realistic plan. At Product level progress should be reviewed at least every Sprint, at the **SPRINT REVIEW**. But I usually advise to do it even weekly.

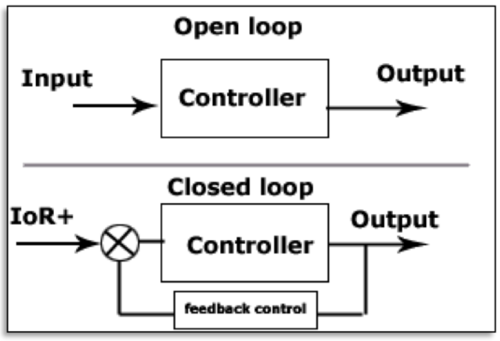


Worldwide implementations have also seen the rise of using Cumulative Flow Diagrams, visualizing the flow of work:



Tracking and forecasting

Dynamic behavior is not blocked but controlled via closed-loop feedback mechanisms. Feedback is used to compare the effective outcome of the system to the expected output, in order to perform adjustments. This inspection technique only works if the real situation is inspected, and not some cover-up, and that the inspectors have good common standards. Hence the need for transparency and visual management techniques for the Scrum artifacts that are inspected.



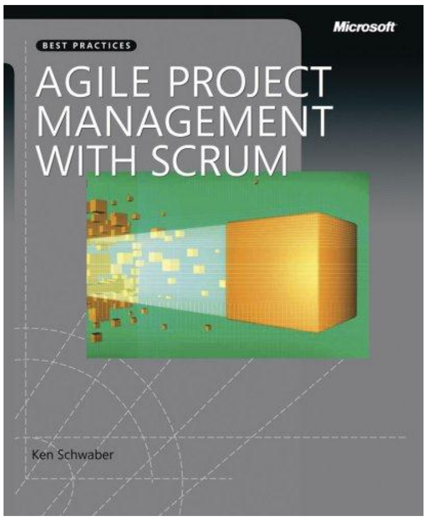
The Product Owner may package PBIs into tentative releases and monitor Release progress upon the remaining work over the open items. The measured progress of past Sprints gives the Product Owner a forecasted delivery date.

A Sprint forms an 'inspect & adapt' cycle that wraps the 24-hours 'inspect and adapt' of the Daily Scrum. Each Sprint cycle starts with a **SPRINT PLANNING** meeting. The Development Team inspects, upon a right-time planning principle, Product Backlog Items that are ordered and clarified by the Product Owner. They forecast the amount of Backlog they will take in upon the past performance and the Team's projected capacity. This becomes the forecasted functionality for a Sprint. The Development Team continues the Sprint Planning session with right-time analysis and design work on the selected items. The Product Owner adds right-time functional refinements that improve the Development Team's understanding and helps them making trade-offs. The result is the Sprint Backlog.

At the end of a Sprint, a collaborative **SPRINT REVIEW** session over the completed **INCREMENT** is organized with the Product's stakeholders. Input, feedback, remarks and comments are gathered. The Product Owner will assess it and order it in the Product Backlog. In the subsequent **SPRINT RETROSPECTIVE** session the Scrum Team looks back at all aspects of the past Sprint in order to adjust, experiment and improve their operations in the upcoming Sprint.

Game ends – You Win

The collaborative Sprint Review provides the Product Owner with the best possible information to decide whether the Product will be shipped, and how additional Sprints can further improve the Value of the Product upon a balance of risk, effort, budget and cohesion.



References

Check out the Scrum Guide, read the books by Ken Schwaber, but above all… play the game and share experiences with worldwide players.

About The Author

Gunther Verheyen is global Scrum leader of Capgemini with a career of over 8 years in Agile. He is occupied in rolling out, training, coaching, supporting and facilitating Scrum for organizations, Teams, stakeholders, coaches and trainers. Gunther is a Professional Scrum Trainer with Scrum.org for the *Professional Scrum Master* and the *Professional Scrum Product Owner* programs. He has developed several Scrum trainings for Capgemini and he is the author of the Capgemini “Agile/Scrum Transformation Playbook”, inspired by and based upon the “CxO Playbook” (Path to Agility) by Ken Schwaber and Jeff Sutherland.

Gunther is a contributor to the international and award-winning Capgemini Technology Blog. He represents Capgemini on the Board of the Agile Consortium Belgium.

About Scrum.org

Scrum.org is the home of Scrum, and is dedicated to improving the profession of software development. Scrum.org provides all of the tools and resources individuals, teams, and organizations need to leverage Scrum software development as a competitive advantage. For more information on Scrum.org, its global community of practitioners, or any of its training and certification programs for Scrum professionals, please visit www.scrum.org.

About Capgemini

Capgemini, one of the world's foremost providers of consulting, technology and outsourcing services, enables its clients to transform and perform through technologies. Capgemini provides its clients with insights and capabilities that boost their freedom to achieve superior results through a unique way of working, the Collaborative Business Experience^{TM}. The Group relies on its global delivery model called Rightshore®, which aims to get the right balance of the best talent from multiple locations, working as one team to create and deliver the optimum solution for clients. Present in more than 35 countries, Capgemini reported 2009 global revenues of EUR 8.4 billion and employs over 100,000 people worldwide. More information is available at www.capgemini.com.