

HUGO MESSER



START AGILE

Ignite agile transformation
within your organization

START AGILE

Ignite Agile Transformation Within Your Organization

Introduction	4
What is This 'Agile' All About?	5
Three Pillars of Transformation	9
Agile Way of Working	9
Digital Roadmap	10
Innovation Engine	10
5 Leadership Capabilities	11
Embracing Change	12
Accountability and Structure	13
Vision, Strategy and Execution	15
Entrepreneurship	15
Innovation	16
5 Leadership Shifts	17
From Long Term Planning to Iterative Work	18
From Silos/departments to Cross Functional Teams	20
From Command and Control to Self-Organization	22
Decision Making	24
From Inside Out to Customer Drives Innovation	25
From Output 'Doing My Job' to End to End Accountability	27
Agile Maturity Assessment (AMA)	30
Agile Fundamentals	32
Why Agile?	32
The Agile Manifesto	36
Iteration and Sprint	41
Waterfall Vs Agile	44
What Is Better: Waterfall or Agile?	48
	2

Agile: Risk & Budget	53
Agile and ROI	55
Scope in Agile Versus Scope in Waterfall	58
How to Combine Waterfall and Agile?	59
Being Agile Vs Doing Agile	62
Roles in An Agile Organization	66
The Scrum Framework	72
The Scrum Roles	73
The Scrum Master	74
The Product Owner	78
Development Team	80
Scrum Events	82
Sprint Planning	83
Daily Scrum	88
Sprint Review	90
Sprint Retrospective	95
Scrum Artifacts	97
Product Backlog	98
Sprint Backlog	102
The Product Increment	104
Kanban	106
Making the Whole Enterprise Agile	110
How to Create an Innovation Engine?	115
What is the Startup Way?	124
5 Laps to Kick Start an Innovation Program	132
What's Next?	140

Introduction

Transformation is hot. Everybody is talking about Agile and digital transformations. Startups disrupt traditional business models, causing enterprises to wonder 'how can we stay ahead and innovate before we get disrupted'.

The good thing about transformation being hot is: it gets many people 'into the forest'. The challenge however is to see the trees from the forest.

There are so many ways to drive transformation, to become agile, to innovate and launch digital products, it becomes confusing.

The aim of this book is to help you START Agile. We show you what the forest looks like. And we explain the individual trees. So you can start your transformation with a clear purpose, roadmap and a starting point.

What is This 'Agile' All About?

The word Agile gained popularity in the '90s when the Agile manifesto was created. A group of IT professionals got together to combine their wisdom on building software.

The general consensus was that the profession needed both a 'mindshift' and a new way of building software products. One of the central tenets of this group was 'self-organization'.

To stay true to that, they described their manifesto as a set of values (abstract, high level), made specific by 12 principles. There was no 'how to guide'.

From that, scrum evolved as the most popular 'how to guide'. Since then, many other 'how to's' have been developed, most labeled 'frameworks' (e.g. kanban, Safe).

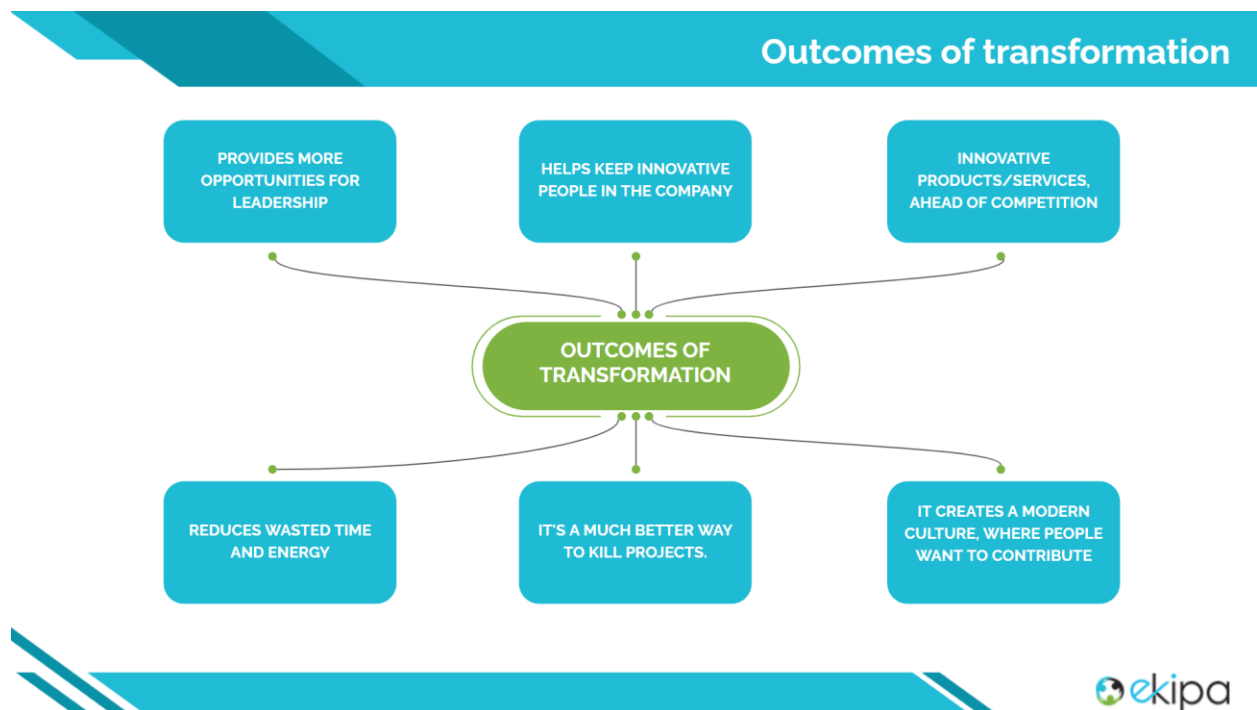
Hence: the traditional notion of Agile stems from the Agile manifesto, scrum and other IT-induced frameworks and movements.

Today, we also speak about business agility. Some of the concepts from the IT agile 'movement', combined with general management and leadership 'theory' sparked a much bigger movement.

Instead of making IT departments agile, we aim for the entire enterprise. This is often called transformation.

For us at Ekipa, when we speak about Agile and transformation, we adhere to the second story. Agile is a means to an end.

We create business outcomes **through** Agile. The outcome of a transformation is much wider than 'using scrum in an IT team'. Transformation leads to ([*courtesy Eric Ries, The startup way*](#)):



A transformation is about changing the 'operating model' of a company, along with its culture.

Transformation isn't a new phrase for 'project' (which has a beginning and an end). It's a never ending process in which we 'reinvent' ourselves.

One of the most inspiring books describing this process is 'Reinventing organizations', by Frederik Laloux. Another

groundbreaking book is the aforementioned 'The startup way' by Eric Ries. Both are described in this book.

Having said this, the Agile manifesto and scrum are important enablers for any transformation.

Therefore, we have described them in detail in this book, along with the enterprise transformation enablers. In true agile fashion, take what is valuable to you and ignore the rest.

Agile is not 'following a how to framework or book'. Agile is 'iteratively finding out what works' and 'continuous learning and change'. If there were blueprints for creating innovation, everyone would be rich.

EXPLORE AND GET YOUR QUALITY COURSE NOW!

AGILE FUNDAMENTAL

BEGINNER

SPECIAL OFFER

**60%
OFF**

~~1,25 JT~~

500 K

**FOR 25 FIRST!
PURCHASES**



What will you learn:

- Understanding of what it means to be 'Agile'
- Understanding iterative vs continuous workflow
- Practical experience in applying Agile Methods via stimulations & workshopping client scenarios
- Benefits of implementing Agile frameworks & practices
- Knowledge of Kanban, Scrum, Lean, Principles
- Action plan to implement Agile frameworks & practices

What will you get:

- 24 Amazing Lectures
- Access on tablet and phone
- Certificate of completion
- 1:55:06 On-demand video

Student Feedback:

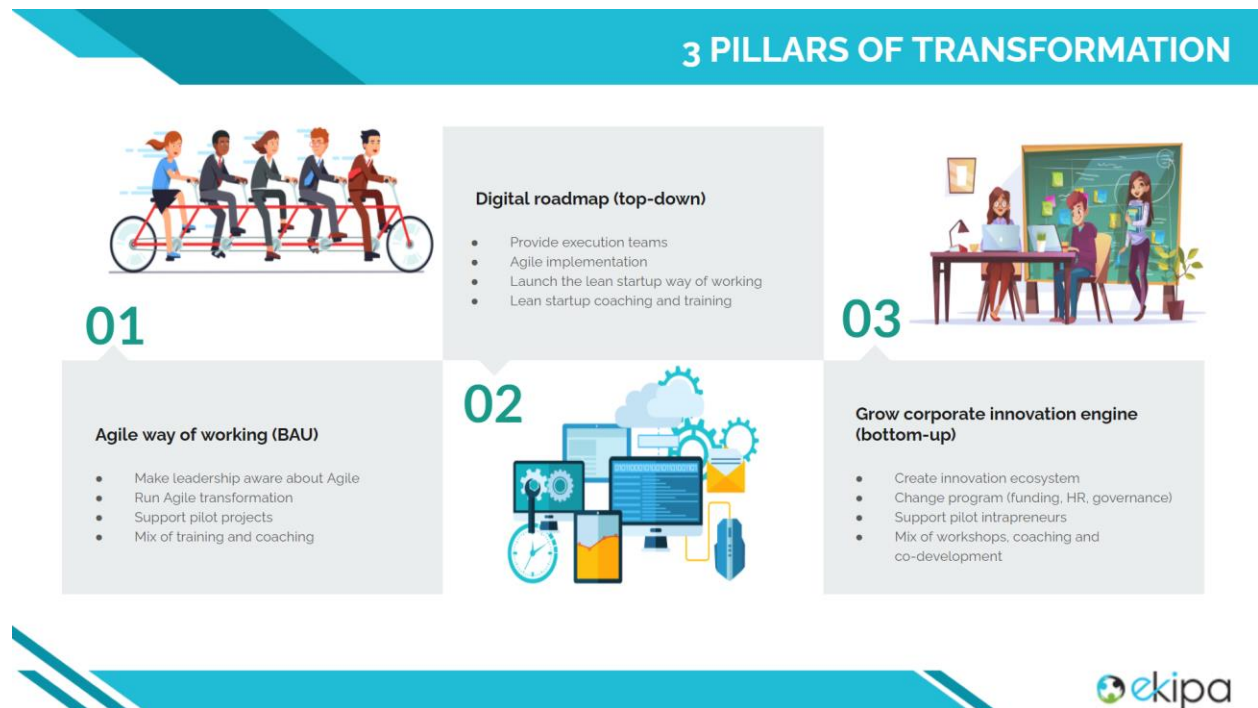


Rating scale: 5.0
"Great to start Agile journey"

CLICK HERE TO GET IT NOW

Three Pillars of Transformation

In the world of Ekipa, transformation today consists of 3 'pillars':



Agile Way of Working

This is where we change the way people approach work. It's about habits, mindsets, structure to enable speed and flexibility. We change the IT approach from long term, waterfall based planning to iterative work.

We adopt kanban in the finance department in order to visualize work and work on the highest value items first. It's about changing business as usual. This lays a foundation for the other pillars.

Digital Roadmap

Enterprises want to develop the digital products of the future. We want to be ahead of competition and fend off startup disruption. The roadmap is developed by senior leaders, often with the help of a large consulting firm.

Once the products and roadmap are clear, the execution is delegated to the organization. This can be done top-down or ideally, co-created with the execution level to make it more agile.

Innovation Engine

The third pillar looks at bottom-up innovation. We identify, empower and provide resources to the latent intrapreneurs. This can be done through internal accelerator programs, specific innovation units or any variant.

The key ingredient is entrepreneurship, which is often the missing 'function' in a large enterprise. You can base your transformation on one or all three pillars.

You can also start at any of the three or ignite all three at the same time. The most important thing is to get started and iteratively find your 'way of transforming'.

5 Leadership Capabilities

In order for a transformation to succeed, we need to develop leadership capabilities (at all levels). Some organizations use this as their starting point.

They invest time in supporting their leaders in their personal transformation, before inviting the wider organization.

Other organizations start pilots on the team level and later down the road, transform their leadership.



Embracing Change

The outside world changes fast and continuously. If you were not convinced yet, Covid probably showed that change is real and instant.

In order for people to transform organizations, we need to first transform ourselves. To do that, we need to adopt a growth mindset. We look at the world as full of possibilities and embrace them.

The infographic is titled "Accomplish BIG Things With a GROWTH MINDSET!" and features a subtitle "Success Begins With Believing You Can". It is divided into two columns. The left column, titled "Instead of Thinking...", shows a sad, pink brain character with glasses and lists 10 negative thought patterns. The right column, titled "Think This...", shows a happy, green brain character with glasses and lists 10 positive, growth-oriented thought patterns. Each thought pattern in the left column is connected to its corresponding positive alternative in the right column by a right-pointing arrow.

Instead of Thinking...	Think This...
I can't do it.	I'm still learning. I'll keep trying!
I'm not good at this.	What can I learn to get better at this?
It's good enough.	Is this the best I can do?
It's too hard.	With more practice it will get easier!
I'm afraid of making a mistake.	Mistakes are how I learn & get better!
They are better at it than I am.	What can I learn from them?
I don't know how.	I can learn how!
I can't make this any better.	I can always find ways to improve!
I don't like challenges.	Challenges make me better!
I give up.	I'll try a different way!

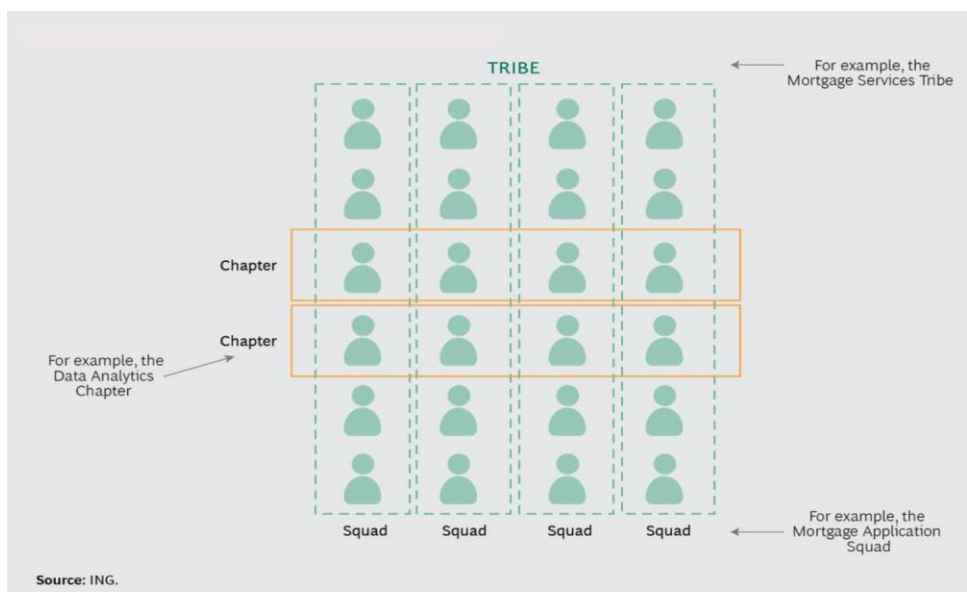
[Source: Jenneke van der Wal](#)

Accountability and Structure

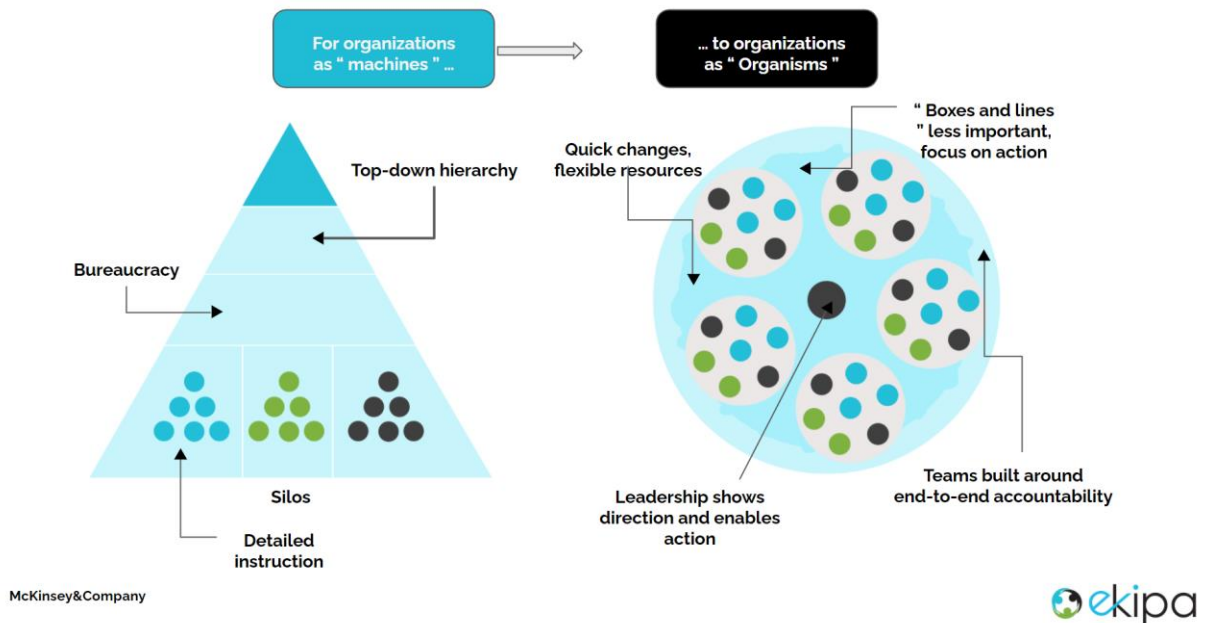
Most organizations look like a pyramid. Somehow we came to believe that 'this is the way to organize'. Agile tries to destroy the pyramid, in which one person reports to the other. In which we have a manager and a subordinate.

There is guidance as to what an agile structure could look like. Since 2019, the leading guidance is 'the Spotify model', partly because many transformations are initiated by some of the big 4 strategy consultants.

The Spotify model speaks about tribes, squads and chapters. A famous example of an enterprise using (elements of) the tribe and squad model is INK bank. You can see the picture below:



Most agile organizations change their pyramid into a circle. The below image, taken from McKinsey, shows this change:



Our belief is that the above examples can be used as guidance, but each company must search its own structure. And we should experiment with that structure.

It's not a 'reorganization' in which we go from pyramid to circle; it's a continuous process of adjusting the structure to serve our objectives.

That takes us to the next part: accountability. Objectives are where we want to get. To get there, we need the right people contributing to those objectives.

And our structure (the way in which the right people organize their work) facilitates those people reaching their objectives. Roles become fluid.

Vision, Strategy and Execution

This capability is the 'middle': this is where things get shaped + done. Everyone knows these three terms are described in any management book, at nausea.

What's important to note here is that transformational leadership requires a strong story. The Agile entrepreneur, driving the transformation, must show where we're going. To get where we're going, we have a strategy and a roadmap. From that, execution follows.

In practice, we have seen that what most leaders lack in getting the three elements right is reflection. They are 'in the dance', doing their business as usual, running around to get things done.

We forget that in order to see the forest from the trees, we need to look from above: [from the balcony](#). From the balcony, we see what's going on, where we're going and whether we're prioritizing correctly.

Entrepreneurship

When people think of entrepreneurship, images of Mark Zuckerberg and Bill Gates come to mind. People who come up with a brilliant idea, get a large investment and change their industries. But entrepreneurs are everywhere, also inside organizations. Call them intrapreneurs or innovators, these are the people who drive change and create new businesses.

As a leader involved in a transformation, entrepreneurs can serve as role models. They can inspire us to challenge assumptions, break or change rules, see opportunities and start new ventures.

Innovation

Innovation is the outcome of entrepreneurship. It's where strong vision, strategy and execution show results. Embracing change leads to innovation. Innovation might even be the main objective of transformation.

We want to move from a 'traditional organisation' to a modern organisation. Leaders need to find ways to get there. This can be through the other 4 capabilities.

But we also need to institutionalize innovation, we need a 'home': a group of people driving it, gathering people and money to get new business models out the door.

5 Leadership Shifts

To transform into an agile organization, leaders need to make 5 shifts. Below image shows the 5 shifts, accompanied by behaviors.

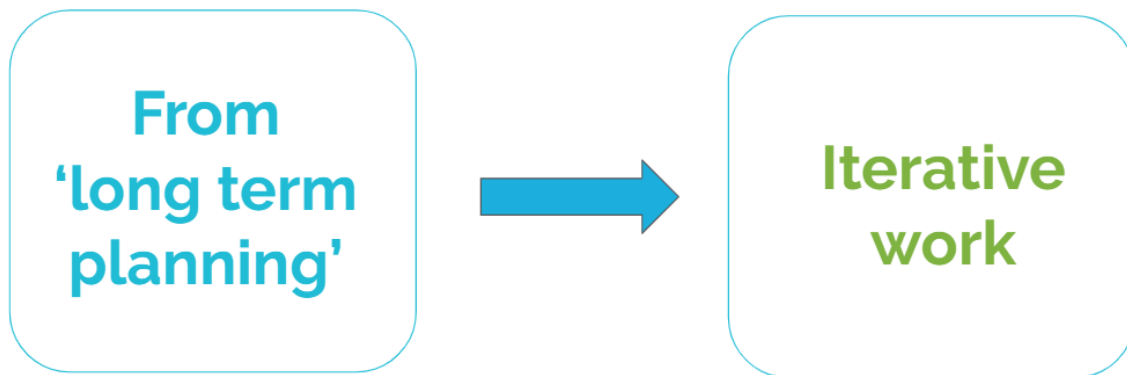
Agile leadership - shifts & behaviors

To grow leadership that supports the agile way of working, we've defined 5 shifts along with the desired behaviors. Leaders at all levels can use this to become stronger agile leaders.

01	From long term planning to iterations	<ul style="list-style-type: none"> Driving experimentation and 80-20 mindset Understanding things change and clarity emerges step by step The budgeting process supports iterative funding on the product or team level
02	From command and control to self organization	<ul style="list-style-type: none"> Trust (let people know what matters, then let them do it) Providing feedback (in all directions) push down decision making and make clear who can decide on what
03	From inside out to customer drives innovation	<ul style="list-style-type: none"> GOOB > stimulate people to improve products based on customer feedback Allowing 'good enough' experiments to drive learning coach people to see their contribution from the customer's point of view
04	From output to outcomes	<ul style="list-style-type: none"> The leaders always focus on outcomes, consistently communicate this with the team moving away from individual kpi's towards shared goals
05	From silos to cross functional teams	<ul style="list-style-type: none"> co-creation, building on participation and empowerment Gather teams around objectives, not roles Work towards having cross functional teams with full time members Hiring people for cultural and character fit, not for role fit



From Long Term Planning to Iterative Work

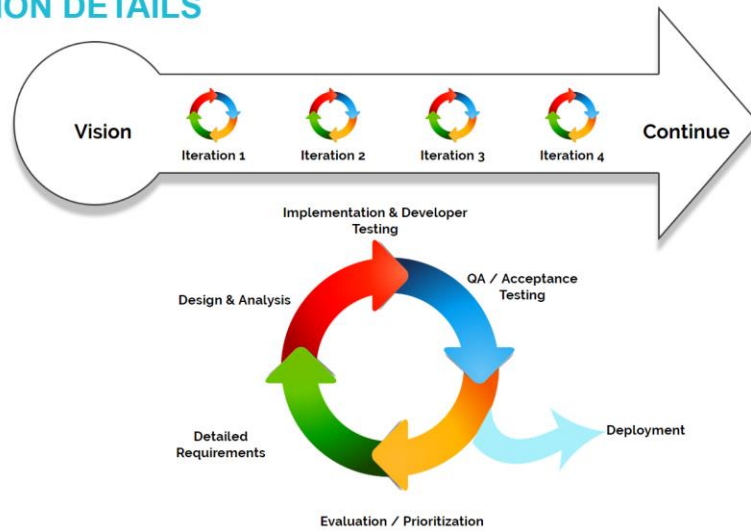


What's needed from leaders here is to let go of the idea that everything can be planned. To leave the traditional project management planning and gantt chart. Instead, we set clear long and medium targets on a high level.

We invest time to describe the details of the execution for the upcoming quarter or sprint. This also means we must look at financing projects differently. Financing should be more 'startup-style': teams must show specific outcomes.

To get there, they get a small piece of the total budget. If they hit their outcomes, they get the next part of the funding. Startups go through different rounds of funding (angel round, series A, B, etc).

ITERATION DETAILS

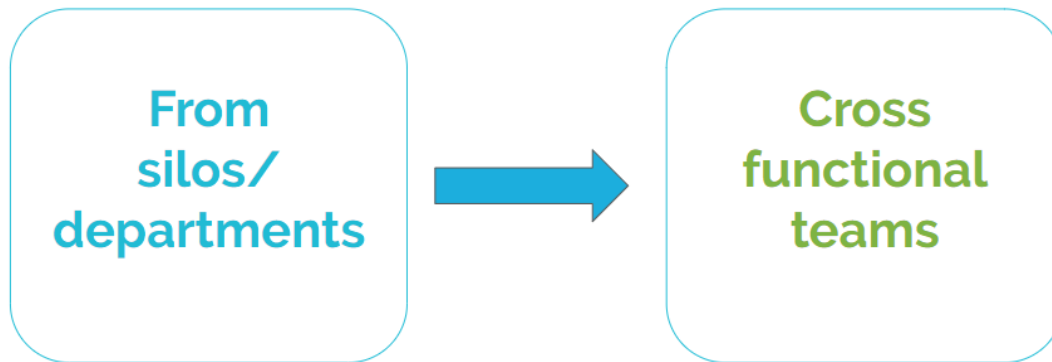


On a deeper level, we need to let go of traditional control. As the future is more unpredictable, it is hard to plan things upfront and then push teams to deliver on time and on budget.

Instead, we must demand specific outcomes in shorter cycles. We control the outcomes, not the inputs or outputs. And we trust teams to deliver them instead of micromanaging them. T

his is where a big mind shift is required from you as a leader. Let this idea sink in and reflect on how things are working in your 'world' when you compare control, long term planning with letting things go and working incrementally.

From Silos/departments to Cross Functional Teams



Most organizations are organized around functions. We have groups of people working in marketing, finance, IT. This creates challenges for agility: handovers, waiting for others, miscommunication, lack of collaboration.

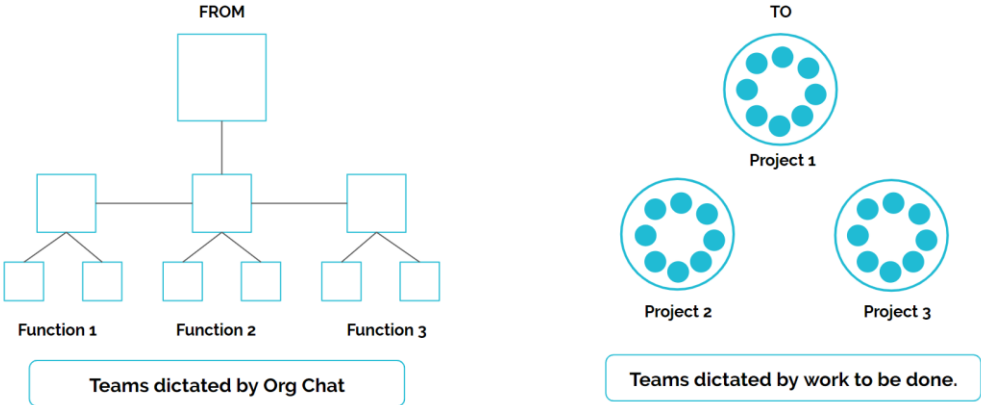
A cross-functional team instead mixes these different functions into 1 team. This team has a mission or set of goals to achieve. They use their interdisciplinary knowledge and experience to reach that mission.

Together, they execute everything to achieve goals, spanning activities across development, sales, marketing and finance. Ideally, we have 'stable teams'.

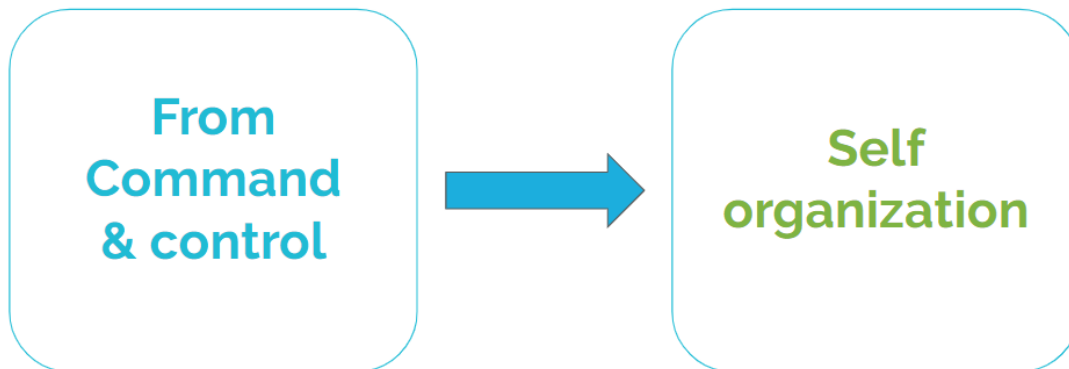
Instead of projects-based allocation, we have people working long term on the same product. This is the same line of thinking as a soccer team: the soccer team keeps the team stable; the same 11 players go into the field over as long a period as possible.

They train together, they play together and know each other's strengths and weaknesses. Only if there's no other way, will they put a reserve player in the field.

WE DESIGN OUR TEAMS FOR AGILITY, ALIGNMENT, COLLABORATION, AND SPEED.

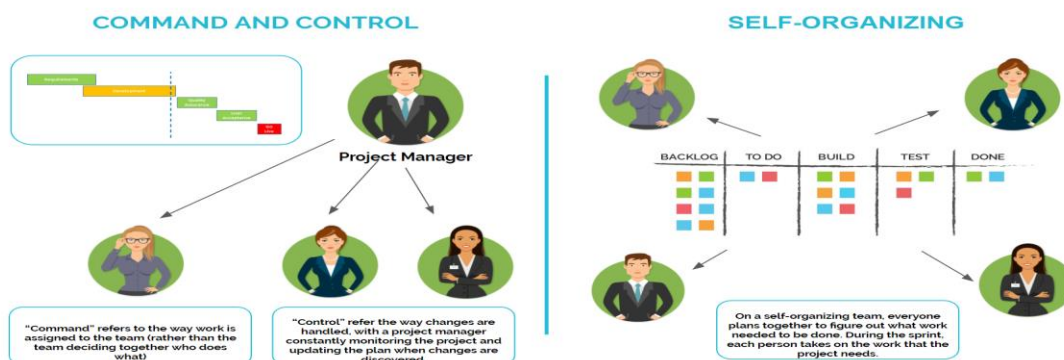


From Command and Control to Self-Organization



One of the core ideas in Agile is to move from 'command and control' to 'self-organization'. This means we must see our role as leaders differently.

Instead of telling people what to do (which trickles down from the top), we give people autonomy and accountability. We trust people to figure out 'how' to deliver 'what' we expect. And we're there to coach, serve, support them.



John Maxwell calls this a shift from 'soloist to conductor'. [This video](#) is a short explanation by John.

Most leaders are operating from a position of 'power'. They have experience and knowledge and know what needs to get done. They also know how to do things and likely can do everything themselves.

To get people towards self-organization, we need to abstain from giving people orders based on our own experience. As long as we keep doing that, people remain dependent on us. They'll keep getting back to us for making decisions.

They will not move things unless we ask them to. Where we want to get is people feeling 100% accountable for outcomes. And to get there, we need to change our own position and behavior.

One of the people worth studying is Frederik Laloux ([Reinventing Organizations](#)). Some of the ideas that he has described:
Roles; no hierarchy, no job descriptions

All power is with the team. There is no formal hierarchy, we abandon the traditional system. There are no bosses to turn to. Teams decide what they focus on, how they achieve outcomes and how they solve problems.

Roles are fluid. We have no job descriptions. Every co-worker is free to create, modify or remove roles. If you think something adds value to your team's purpose, you own it and move it forward.

Once a week, we organize an overall team retrospective. During this event, the teams discuss the current roles and propose additions, improvements, modifications.

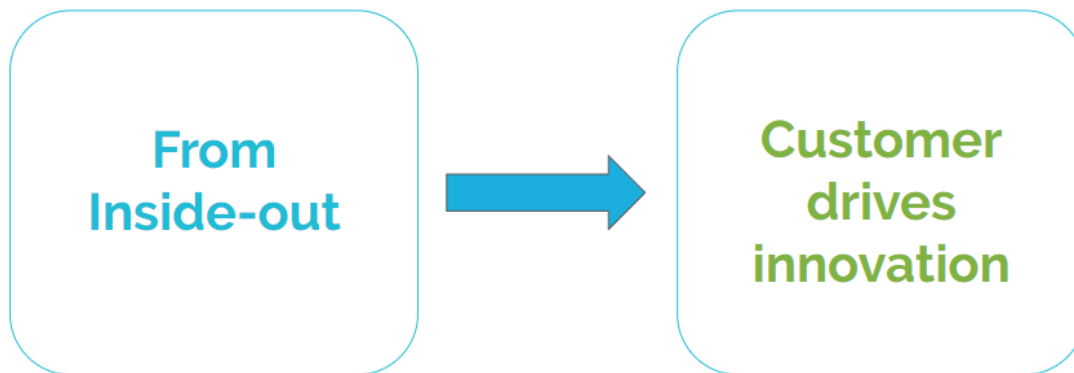
Decision Making

Decisions are made based on the '[advice process](#)'. If you recognize a decision has to be made, you own that decision. We assume you can make most of the decisions by yourself. If you need the help of other co-workers, ask them for advice, make your decision and move on.

For a large company like Telkom, you may be thinking that these concepts are not feasible. And you might have a point in that it's very different and will take time to implement.

However, Telkom needs you as a leader to make baby-steps on the road to agility. Even if you get inspired by one of these ideas and implement it in one team, you've grown as an agile leader.

From Inside Out to Customer Drives Innovation



Traditionally, digital products are developed top-down. A BOD member reads something and drives the plan.

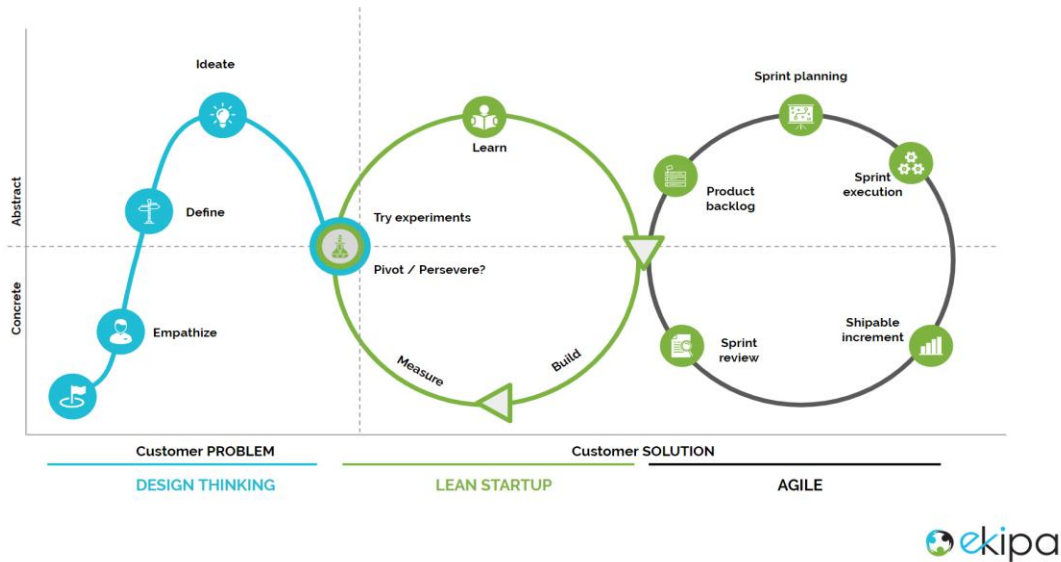
The ministry asks for a tool and we execute according his ideas. Or we hire an external consultant and let them lay out the plan.

The people building the product follow the roadmap without interaction with the customers eventually using the product.

Often the result is we build a product that nobody uses; or we built a killer app, but missed the most important features that users actually need and a startup runs away with the success (what's app?).

The lean startup movement tries to turn this upside down. A lean startup starts with the user. They don't build anything before they've validated user needs.

Based on MVP's that keep evolving, they validate functionalities before building everything. To put lean startup into perspective, have a look at below overview of the different concepts in the agile 'lifecycle':

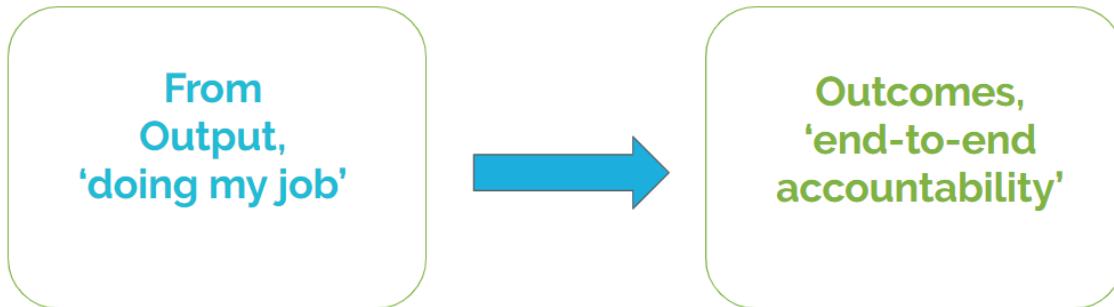


Design thinking helps us structure our ideas and idea selection in the ideation stages. Lean startup gives us the tools to validate our value proposition, product and business model in the incubation phase.

Agile helps us set up structured processes and teams when we start scaling our product in the market validation phase.

What all these ideas have in common is: the customer drives everything. We continually try to base our decisions on validated learning from the customer's point of view.

From Output 'Doing My Job' to End to End Accountability

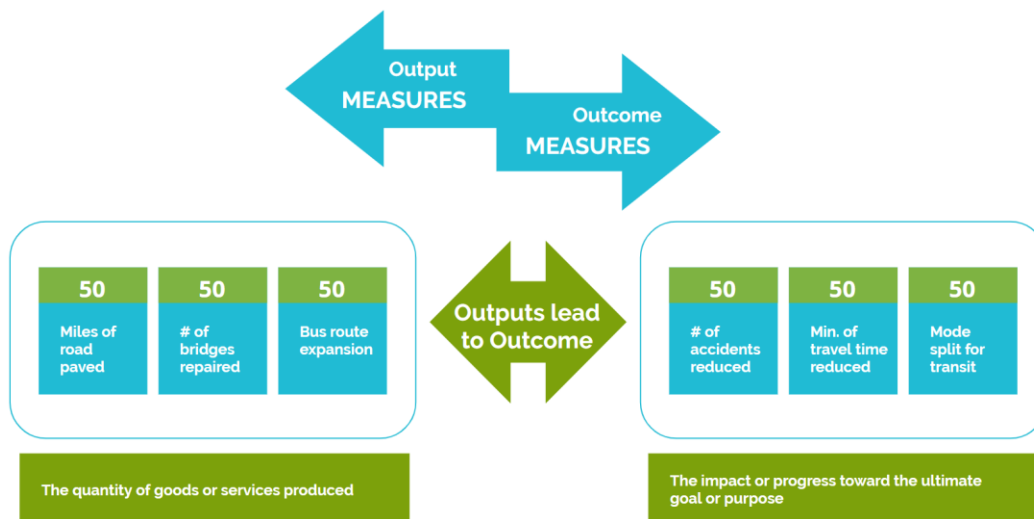


The traditional leader uses KPI's to focus people's attention. This leads to individualism: people push their own agenda as long as they hit their KPI (and with that, their bonus).

In Agile, we want people to achieve outcomes as a team. Imagine a soccer team: 11 people orchestrating, collaborating to win the match.

They have a team goal, they train and work together to achieve that and the coach facilitates them, helps them win.

A few simple examples to show the difference between output and outcomes in a road construction company.



The left side shows clear outputs. People will ‘tick off their box’ once you as a leader have asked them to pave 50 miles of road. They’ll be happy and proud and you can also tell your boss ‘we did our job’. Now on the right side, you have outcome-based metrics.

Your team has adopted a metric showing that they have reduced accidents with 50 on the stretch or road you’re responsible for. Your team has ideated and implemented ideas that reduced the number of accidents. Part of that was good road, but they also came up with new lights that improved visibility.

They also put flexible rails along the road and they added some icons to google maps to indicate dangerous points. All of that happened because you asked them to ‘reduce accidents’, not just to ‘put the road in place’.

The ideal structure is the team that works on a specific product or service offering. The team consists of people from different functions. Those people sit together instead of 'in their department'.

They all have a single goal and everyone collaborates to achieve it. Their performance system is in sync with this team-work. Performance is evaluated on the team level (or both on the team and individual level).

This makes team and the people in the team feel full accountability for the outcomes. Instead of creating a small piece of the puzzle (output), they sync their efforts to create the whole puzzle. This also means for you as a leader that you must become an entrepreneur.

You act as the owner, as if your life depends on the success of your product (and it does). You feel like a startup founder, you move in unknown territories and do anything required to make the product add value to Telkom.

Steve Blank, a famous silicon valley teacher, has described an [inspiring article about innovation inside large enterprises](#). As a leader, your role is to make sure that you don't just produce coffee cups and theater, but true, long term value.

Agile Maturity Assessment (AMA)

A question that comes up in any transformation is: how do we measure the impact? The challenge in answering this question is the definition of impact.

It's not easy to have a direct link between the transformation and the business outcomes (for example revenues, client score, profit). What's easier is to measure the agile maturity of the organization. This shows agile as an enabler of business outcomes.

At Ekipa, we have developed an agile maturity assessment. The assessment looks at:

People: do we have the right people in the right roles?

Process: is our way of working agile?

Structure: what does our hierarchy and structure look like?

Tools: do we make proper use of tools to stimulate transparency?

Leadership: do our leaders empower and show progress on the 5 shifts?

Our assessment consists of 30-40 questions across these 5 categories. The outcome shows the maturity of a team or the whole organization:

EKIPA AGILE MATURITY ASSESSMENT

Agile Maturity Assessment (AMA) is a way to measure team's agile maturity. It is presented in the form of a survey, and distributed to the team at different times to measure and analyse the team's progression towards an ideal agile state. It is a self assessment, not reflecting Ekipa's opinion.



5 Levels Of Agile

- Not Agile - **No adoption** of agile values and principles in the way of working
- Fake Agile - **Started to adopt** the agile values and principles in the way of working.
- Basic Agile - **Limited adoption** of agile values and principles in the way of working.
- Good Agile - **Adopted most** of the values and principles in the way of working.
- Excellent agile - **Adopted all** values and principles in the way of working by the whole organization.

Agile Fundamentals

In this chapter, we look at the fundamentals of Agile; the building blocks of an Agile organization and all the big questions that appear when thinking about Agile.

We start from the agile manifesto and look at the traditional waterfall model. We also look at budgeting. While there are many more topics than we can cover in this short book, we hope this gives a good overview and starting point for your journey.

Why Agile?

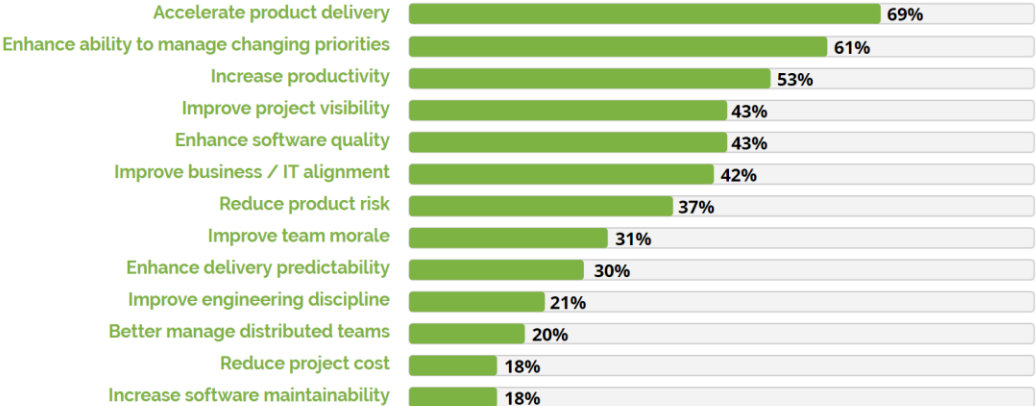


The main promise of Agile is for companies to adapt to the speed of changes in the marketplace. All across the world, digital products disrupt existing industries and business models.

Startups search for disrupting product ideas and move much faster than traditional, larger companies. By becoming more agile, companies can catch up with this speed of change.

In a research 'state of agile' conducted by [version one](#) across more than 1000 respondents, the following list of reasons was generated:

Reasons for Adopting Agile



Improving project visibility (43%) moved up three places to become the fourth most popular reasons started for adopting agile this year and accelerating product delivery increased from 62% last year to 69 this year.



1. Accelerate product delivery

While the promise of Agile is to speed up product delivery, it is not always the first outcome of an agile transformation. It takes time to create the right habits and culture in teams and the wider organization to reap the 'speed' benefit of Agile.

For management, speeding up is however the main reason (69% of respondents stated it as the main reason for adopting agile) to start an agile transformation.

As stated above, companies need to speed up product launches, because if they don't, their competitor or a new startup may do it faster than them.

2. Enhance ability to manage changing priorities

One of the basic ingredients of Agile is iterative work. Teams work in iterations or cycles of 1-4 weeks. That's the agile timebox.

The product owner can decide what user stories are added to that iteration. As the product owner is typically someone from the business side, he understands what features have the highest value for customers.

This enables the product owner to change the priorities of what will be built by the team on a sprint by sprint basis. The product owner can watch competitors, learn from users, speak to customers and all this input leads him to prioritize the product backlog.

Traditional approaches to product development are based on long term planning. Requirements are fixed and made upfront. The requirements are then built sequentially by designers, developers and testers over a longer period of time.

Within this longer time horizon, change is not welcome as it disrupts the plan. This puts such product teams in a weaker position when things change quickly in the market.

3. Increase productivity

Agile teams work in short timeboxes (sprints or iterations). Within their timebox, they plan their work, align on a daily basis, demonstrate the pieces of the product they built and also look at their own performance, collaboration and productivity.

This means there are very short feedback loops that enable the team to learn. As they align daily, they remove blockers more quickly (with the help of a scrum master). All of this leads to a higher productivity in agile teams.

Traditional, waterfall based teams, have a longer time horizon. If a project spans 6 months, in the early stages, things will move slower. As we near milestones and deadlines, things will speed up. But overall, the productivity of teams will be lower than in Agile teams.

In the version one research, there are more reasons for adopting Agile, which their report explains in more detail.

Most companies adopt agile for IT product development. The above research is mainly based on reasons for adopting agile for software development.

In the past years however, companies have also started adopting Agile outside IT. There is Agile for marketing, Agile for HR, Agile for Sales, Agile for finance.

In all of these domains, Agile helps the overall organization to better be able to respond to the market, to become more flexible in their execution.

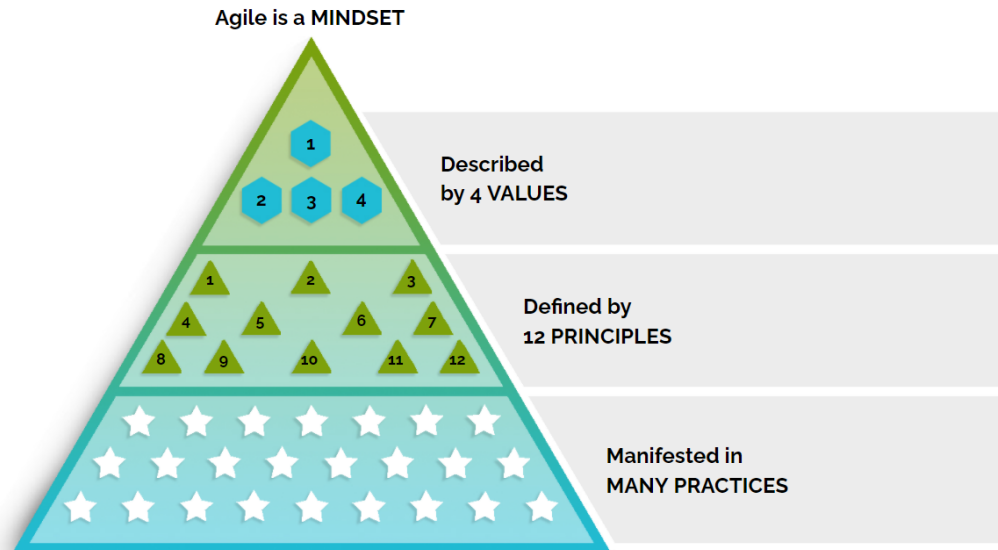
Some companies have even started enterprise agile transformations, in which they change their traditional hierarchical model. The whole organization changes to a cross functional (cross departmental), Agile team-based mode of operating.

The Agile Manifesto

The Agile manifesto was created by 17 seasoned IT practitioners in 2001. They gathered all their experiences into a high level for software development'.

This manifesto can be used by people in their own context. There are values, which are high level enough to be translated into anyone's specific context. Underlying the values are 12 principles that give more practical direction on how to live the values.

AGILE MANIFESTO-A BRIEF HISTORY



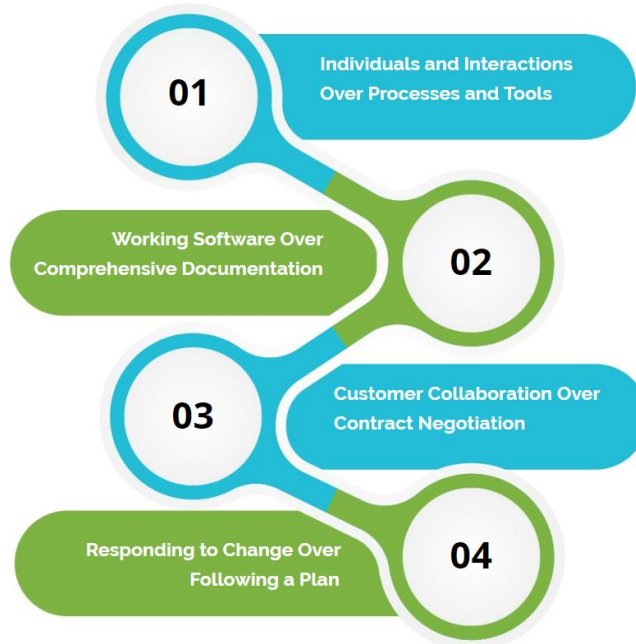
In February 2001 a group of 17 developers met in Snowbird, Utah and created
'A Manifesto for Agile Software Development'

- ▶ A group of people who held a set of compatible values
- ▶ based on trust and respect for each other and promoting organizational models based on people, collaboration, and building the types of organizational communities in which we would want to work'



AGILE VALUES

We are uncovering better way 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

A well-functioning team has the right people, collaborating and communicating effectively. Instead of using the ‘McDonalds’ model that uses SOP to force people to do things in a specific way, we empower people to do what’s right. Creating software is ‘brain work’; thinking doesn’t happen in linear ways dictated by process.

Working software over comprehensive documentation

The outcomes a software team are after? Working software. Documentation has a function, but we shouldn’t see them as the outcomes in itself.

The goal is to minimize the time spent on documentation, while complying with the (government) rules. This means we might

challenge some of the rules if they conflict with our goal of creating great products.

Customer collaboration over contract negotiation

If we collaborate with other people, our focus should be on collaboration. We want to form strong relationships and trust. We aim to co-create and form 'one team'.

Time spent on contract negotiations (for example on 'how to deal with change requests') goes against this aim. Although we recognize that we do need contracts, we could suffice with a 5 page contract instead of 50 pages.

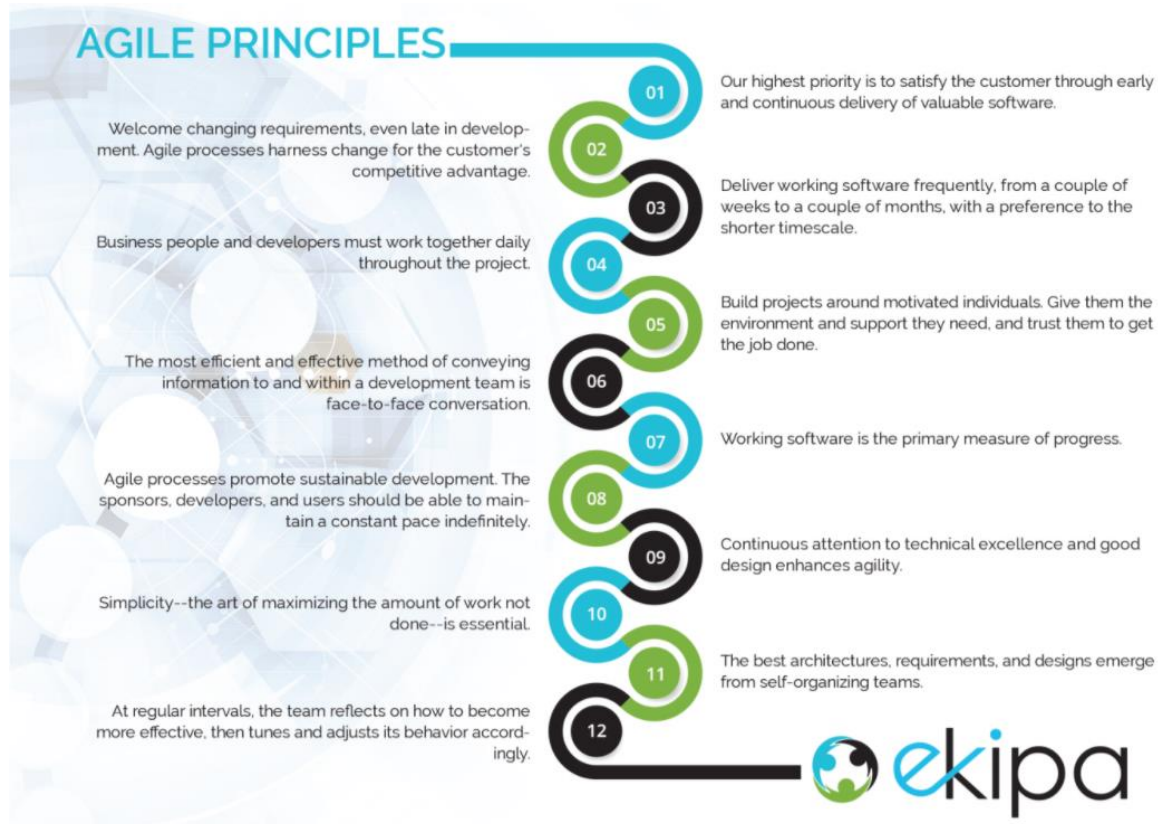
Responding to change over following a plan

The market, needs and ideas change all the time. As we iterate towards a great product, we stay open to those changes. Our process accommodates change.

Instead of deferring changes to 'after the project', we work on the highest value items. We don't blindly follow a plan devised upfront, but always question our assumptions as we get wiser based on reality

The 12 Agile principles

The 12 principles articulated in the Agile Manifesto make the values more practical. They specify each of the values, which serves as guidance to 'how to be and do agile':



Iteration and Sprint

Iteration is the common agile term for one cycle. It is a generic term used in the Iterative and Incremental Development (IID) process. Scrum calls the iteration a 'sprint': a time box of 1 up to 4 weeks in which an increment of work is executed.

As per the Scrum Guide:

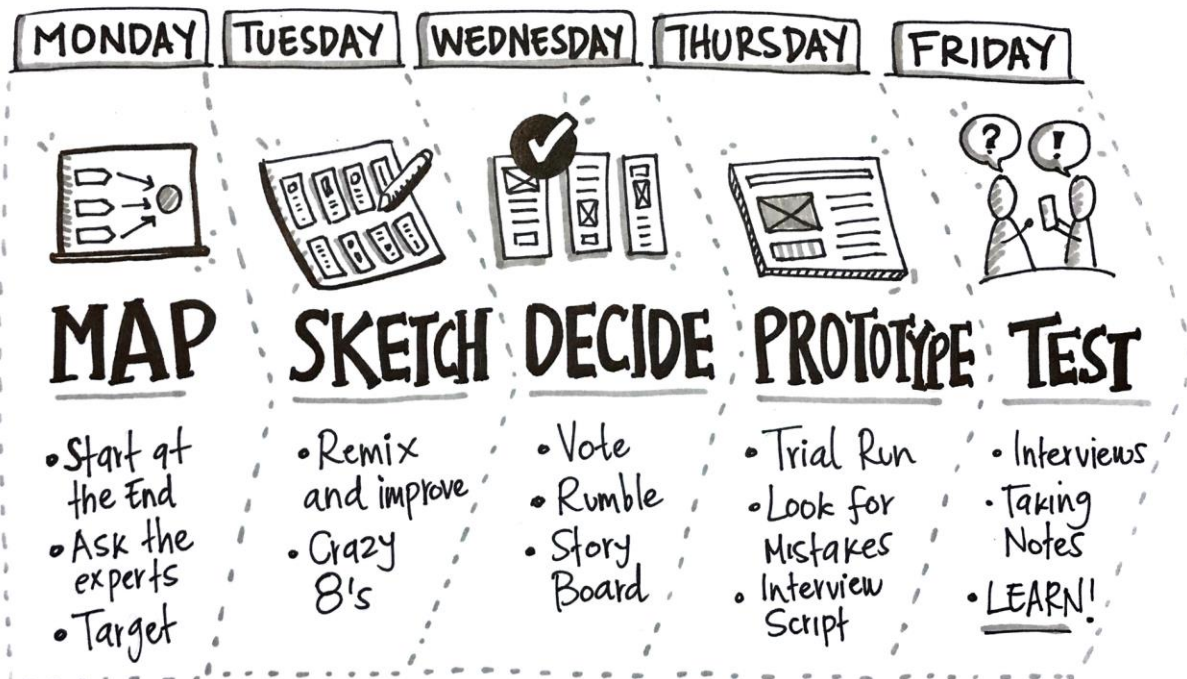
"The heart of Scrum is a Sprint, a time-box of one month or less during which a "Done", usable, and potentially releasable product Increment is created.

Sprints have consistent durations throughout a development effort. A new Sprint starts immediately after the conclusion of the previous Sprint.

Sprints contain and consist of the Sprint Planning, Daily Scrums, the development work, the Sprint Review, and the Sprint Retrospective."

To make the confusion even better, we also have the concept of 'design sprint'.

* DESIGN SPRINT WEEK *



[Source: Nelson Almanzar](#)

The most important thing to remember is that a time box is a short period of time in which a team has a strong focus to get an important 'thing' out the door.

If it's a product team, then the 'thing' could be a feature. If it's a team focused on marketing, the 'thing' could be 'reach 1000 people with our campaign' or 'launch our campaign website'.

Each team member takes up work that contributes to the sprint goal. We all work hard together to get this goal completed.

EXPLORE AND GET YOUR QUALITY COURSE NOW!

AGILE FUNDAMENTAL

BEGINNER

SPECIAL OFFER

**60%
OFF**

~~1,25 JT~~

500 K

**FOR 25 FIRST!
PURCHASES**



What will you learn:

- Understanding of what it means to be 'Agile'
- Understanding iterative vs continuous workflow
- Practical experience in applying Agile Methods via stimulations & workshopping client scenarios
- Benefits of implementing Agile frameworks & practices
- Knowledge of Kanban, Scrum, Lean, Principles
- Action plan to implement Agile frameworks & practices

What will you get:

- 24 Amazing Lectures
- Access on tablet and phone
- Certificate of completion
- 1:55:06 On-demand video

Student Feedback:



Rating scale: 5.0
"Great to start Agile journey"

CLICK HERE TO GET IT NOW

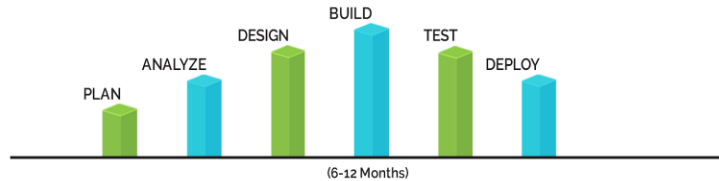
Waterfall Vs Agile

WATERFALL VS AGILE

LET'S COMPARE AGILE DELIVERY TO MORE TRADITIONAL METHODS OF DELIVERING 'PRODUCTS':

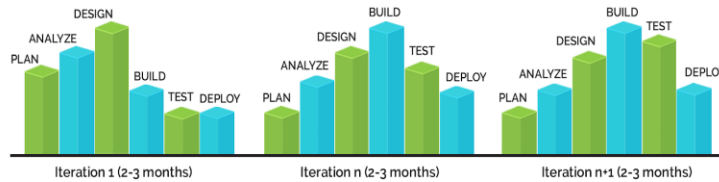
WATERFALL DELIVERY

Waterfall method involves implementing something in sequential approach



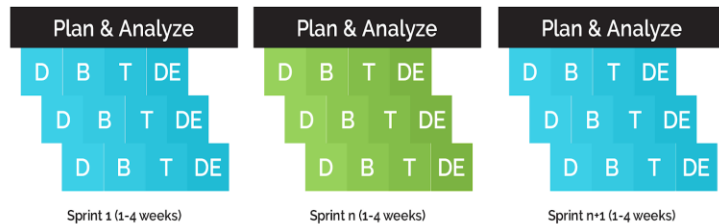
ITERATIVE DELIVERY (MINI WATERFALL)

Iterative focuses on short iterations addressing all aspects of the lifecycle in each iteration



AGILE DELIVERY (PARALLEL)

Agile development focuses on continuous planning, design, build, test and deployment



Waterfall has been 'invented' decades ago. Based on traditional project management from other industries, the software industry copied the long term planning approach.

As in construction, we first build a foundation (architecture), we lay bricks (coding) and then we test and integrate everything.

There are 2 big differences between waterfall and agile:

A. From Long Term Planning to Iterations

In the waterfall model, we gather all the requirements for the product we are building upfront. A business analyst speaks to stakeholders and gathers all their ideas.

He then translates that into software requirements documents. These documents can be very large for big projects as their goal is to capture 'everything' that needs to be built in the product.

Based on this extensive requirements document (which can take months to complete), the engineers and testers estimate the workload (by reading all the requirements).

They come up with a plan to build the product from beginning to end. This includes everything from architecture, designs, coding, to testing and deployment.

Once the estimates are confirmed, a planning is made (which can sometimes span months or even years) with deadlines and milestones. Once the planning is confirmed and the budget approved, the team starts working on the product.

First, the architects will design the architecture and make choices that impact the whole lifespan of the project (upfront). Then designers make designs that are used by the developers to create the software code.

All this work is done stage-wise. The designers work as a team to finish all the designs and when approved, hand them over to the programmers.

Once the programmers finish their coding, the testers test all the code. Once this testing process is done, the product can be integrated and deployed.

Any request for change that comes in during the project is parked as a change request and ideally done after the initial project is completed. If it's done in between, the whole planning needs to be revised, something which is seen as 'disrupting'.

In agile, we work differently. Instead of planning everything up front, we work in iterations. We describe the requirements for the product on a high level.

The business people gather in a room with the development team and discuss all the ideas they have. Requirements are captured in 'user stories' (short user-centric descriptions of the user needs).

These user stories are written down on sticky notes and prioritized on a backlog. Based on this initial discussion, the team has a high-level understanding of what needs to be built. They can roughly estimate the time required to complete the whole project.

Using this estimate, a budget can be assigned along with a timebox for the project. Within that timebox, the goal for the

team is to build as much 'value' as possible. The team then starts the first iteration (typically 2 weeks for an agile software team).

To start, the team discusses in detail with the product owner what they need to do to complete the user stories. At this stage the requirements become 'detailed' (so we do make detailed requirements, but only when we are close to the implementation of those requirements).

The team executes the iteration and then delivers the user stories to the product owner and stakeholders.

Based on what they can see and inspect, the product owner decides on the priorities of his product backlog. This is then the input for the team to decide what they can build in the next iteration.

Changing requirements is welcomed, because it means the team is recognizing that the user needs something which was not clear to the team when the project started.

B. From sequential work to cross functional teams.

To make the iterative approach work, teams need to be cross functional. Where waterfall has teams of the same function working together (developers, testers, designers) to make the stages work out; agile has all these functions in the team.

Each team has 3-9 developers, a mix of designers, business analysts, coders and testers. This team should be capable to deliver the next increment of the product and we keep the team as stable as possible.

Within the team, we have a scrum master to facilitate the work within the team. He is not a project manager or team lead who assigns work; instead, he is a coach who helps the team communicate and collaborate as productively as possible.

What Is Better: Waterfall or Agile?

We often hear people ask about the 'best choice' between waterfall and agile.

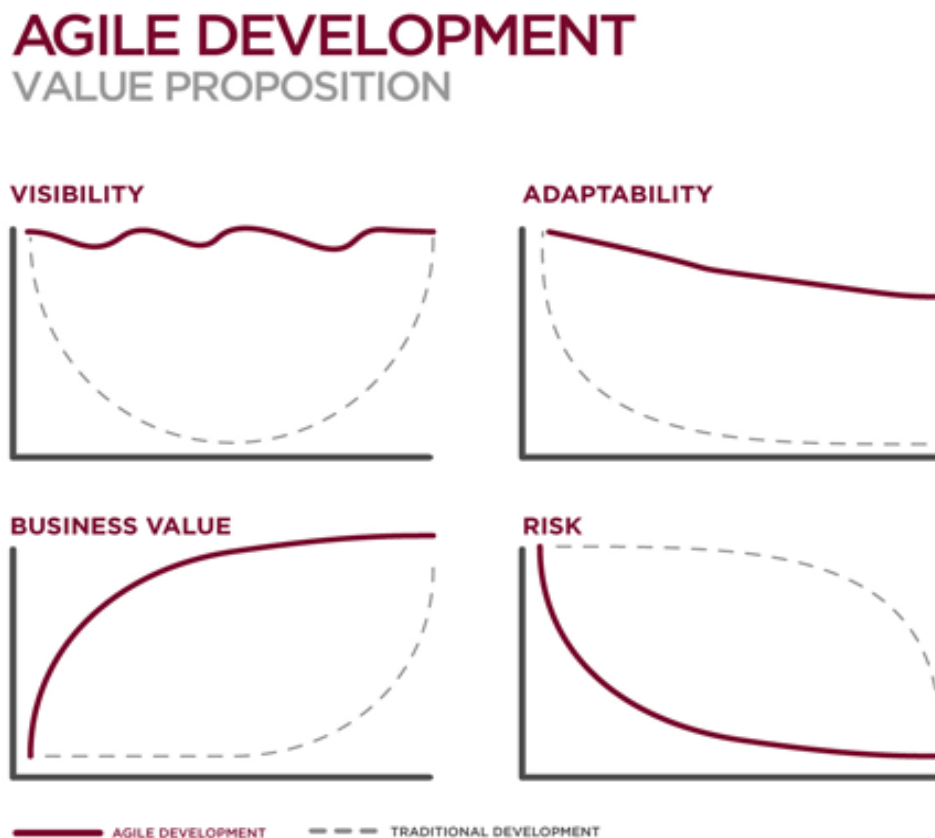
We believe that there are products that may be better to run in waterfall: products that are predictable to build (for example copying or migrating an existing tool).

There may also be situations where it's better to not do everything in agile, because an organization has just started with agile.

It takes time to get buy-in from everyone if people have been working in the waterfall model for a long time.

It also takes time to create the right culture, tooling and organizational support to move work to agile (Which is why agile transformations typically take years in a large enterprise).

Having said the above, we believe agile brings a lot of benefits to most projects and organizations. The below image, courtesy of [Henrik Kniberg](#), shows the 4 'value propositions' of agile.



Visibility

In waterfall, the visibility of a project is high in the beginning. This is because people have discussed and analyzed requirements extensively.

Everything is documented in requirements documents. Once the project starts, the visibility goes down, because people use gantt

charts (instead of 'real progress') to show completion against plan. Real, usable software, usually is delivered very late in the project.

In agile, the visibility is high in the beginning and stays high over the course of the project. Requirements are captured on boards using sticky notes (or online tools like Jira or Trello).

Each iteration, the business people interact with the team to see what needs to be delivered. They then get a piece of the product delivered in the sprint which they can use, click, test and give feedback on.

This feedback leads to new insights and priorities. It is always clear to everyone what has been delivered and what the bottlenecks are. There is transparency and constant communication and alignment.

Adaptability

In waterfall, everything is adaptable in the early stages of a project. As the requirements are developed, we can still choose what to build or change.

But once the requirements are signed off and the team is doing the work, things can not be changed that easily. If they do, it leads to disruptions of the plan.

In agile, we welcome change. In each iteration, new insights, new requirements, can be added, removed or prioritized in the product

backlog. Over time, some technical things get nailed, which is why the graph shows a small decline.

Architectures get fixed, code is made and deployed, so it becomes harder to change things over time, but still things are much more adaptable than in a waterfall project.

Business Value

In waterfall, value is delivered in a big bang. The different groups of people have been working sequentially on the software.

Once the testers have finalized their work and bugs have been fixed, the product is delivered to the business side. That brings value at the end of a project.

Agile delivers value from the early stages of a project. After each iteration, a (small) piece of the product (a potentially shippable product increment) is delivered.

This piece of software can be inspected and feedback can be provided to the team on what (not) to build next. Some parts of the product can already be used by customers. The value is delivered incrementally from the beginning of the project.

Risk

Waterfall is high risk. As we don't see real software being delivered, we need to wait till the end of the cycle to see what's delivered. All types of risk may come up: technical risks, business risks,

regulatory risk. These risks stay alive until the end of the project when all is 'done'.

In agile, risk can be managed continuously. Risk people can even be part of the team and monitor what the team sees. They can see on a daily basis what technical risks come up and how the team deals with it.

They can see how users respond to the software or whether people use the software at all, once the increments are 'shipped' to users.

Agile: Risk & Budget

One perception many leaders have is that Agile brings more risk. This perception is usually based on the sense that there is less 'control'.

In waterfall-style processes, there is large upfront planning with deadlines at which parts of a project are delivered. The manager can use this planning to push and control progress.

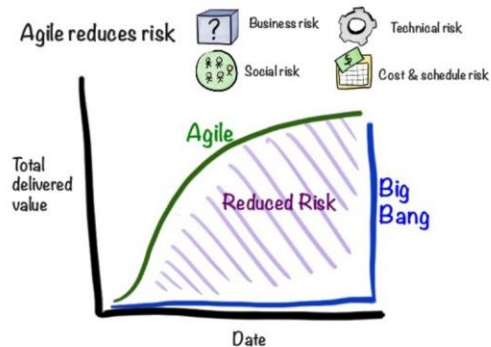
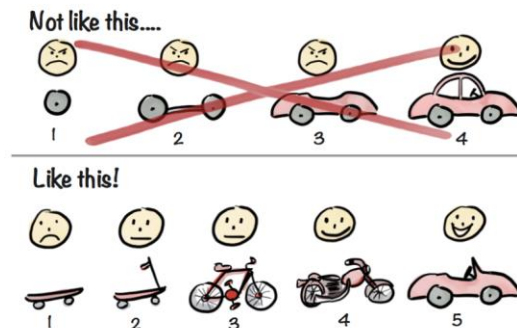
In Agile, there is less up-front planning and we change as we get into the project. The bottom line is: the iterative nature of an agile project gives regular opportunities to bring any risk to the surface.

Based on these insights, we can take steps to reduce the risk. This leads to less risk, not more. Budgets can also be spent incrementally.

As each iteration delivers outcomes, we can decide at regular intervals to invest more or stop investing in certain projects or products. It's easier to kill projects that aren't delivering the expected outcomes (and hence saving money).

It's easier to see what products are producing the right outcomes and invest more in those. If budgeting becomes agile this way, we can invest money where it has most impact.

HOW ABOUT RISK?



Henrik Kniberg



Budgeting can also follow quarterly goals, for example using OKR. Each quarter, a team create objectives. They define the key results they will deliver during the quarter to reach those objectives.

Based on those goals, they indicate the budget and people they need to accomplish those goals. Budgets can be provided annually to a team or a tribe (a collection of teams).

The teams then decide how to spend that quarter on quarter. Budgets could also be created on a per quarter basis. Or even more extreme: no more budgets.

If there are no budgets, teams may 'pitch' on a quarterly basis to an investment committee. They show outcomes from the previous quarter along with the OKR for the upcoming quarter.

The committee decides whether and how much to invest. This follows the model used by venture capitalists.

A more extreme case is used in Netflix: Act in Netflix best interest. That's the only guidance people at Netflix get to make investment decisions.

If they want advice from people, they can ask. But it's up to them to make the decision, no matter whether the decision is worth a million or 10 dollars.

Agile and ROI

Many companies rely on the concept of ROI (Return on Investment) to justify expenditure on projects or products based on project future financial returns.

Usually expressed as a percentage gain or loss on an investment over time, it is useful to compare profitability between different options.

Can We Use ROI in Agile Contexts?

Simple ROI calculations become more challenging in an Agile context however.

Agile should focus more on products rather than projects - so team size/expenditure is flexible over time, goals shift and change over time, so returns are not as stable as you would expect with a more traditional project management approach.

Agile gives better returns. Leaving aside the question of measurement for now, we can say that, in theory, Agile should offer a better ROI than slower project management because:

- Incremental agile product development with a strong focus on customer means products are a better fit.
- Agile allows us to focus on the highest value features of the product first - we don't build things customers don't need.
- Agile gets product to market faster giving us an earlier return on investment.
- Agile allows us to modify our product faster in response to customer/market changes again increasing returns.
- Team is more efficient working in an Agile way - better collaboration and communication.
- An Agile focus on higher quality software will have fewer defects, less downtime, better maintainability/

OK, So How Do We Measure Agile ROI Then?

Agile really challenges the traditional conception of ROI. There are some things we could do to try to measure traditional ROI in an Agile context:

- Assign value (in \$ or other) to features delivered by an Agile team
- Use analytics to measure conversion in sales (or other) for features delivered
- Track team costs over time and compare against value of features delivered

These approaches are really trying to force a traditional accounting method onto an Agile way of working, which is not necessarily the right way to do it. In theory Agile should improve ROI but being Agile makes ROI harder to measure.

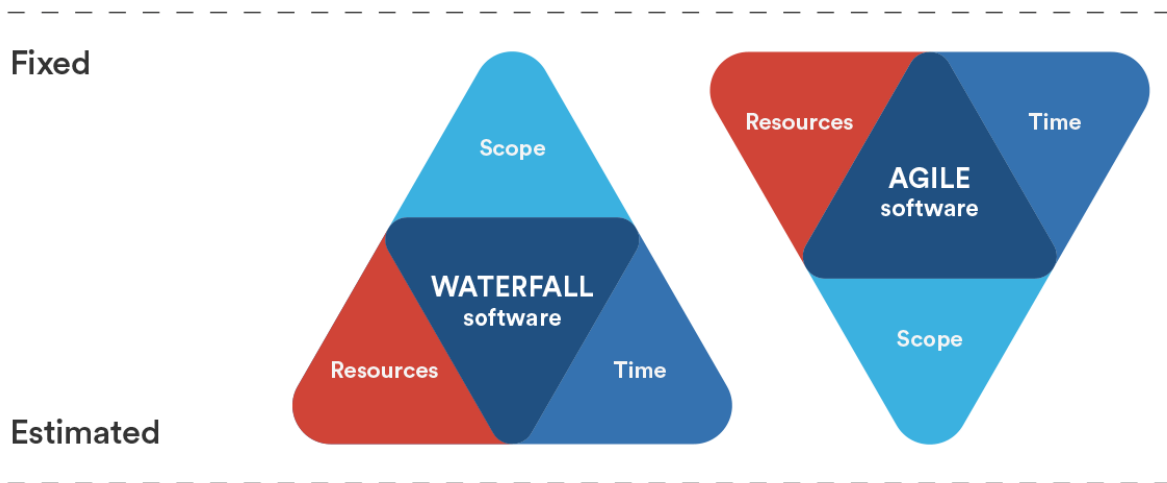
Innovation Accounting

Eric Ries (The Lean Startup) proposes a new way of thinking about this - Innovation Accounting which has three basic steps:

1. We work in an Agile way towards an MVP (Minimum Viable Product) to test business hypotheses before we spend a lot of money on building the full product. We iterate and experiment.
2. As teams move towards the delivery of the MVP, they can decide whether or not to enter the market.
3. At a critical point, we must decide whether to continue, end the product or pivot - ie change the product/market segment in a significant way because evidence suggests a different way might work

This is the startup way - generally teams have a bucket of money they “burn down” which they can track. Check out [Eric Ries talking about Innovation Accounting.](#)

Scope in Agile Versus Scope in Waterfall



The above image shows the basic difference in how waterfall deals with the golden triangle of project management versus how agile does.

In waterfall, scope is fixed; requirements are captured in software requirements documents and signed off and 'frozen'. The resources and time are estimated and variable.

If we reach the middle of the project and we see things are delayed, we add more people. If we still haven't finished the complete scope by the end of the timeline or deadline, we extend the deadline.

In agile, contrary to waterfall, the scope is estimated and can change. Instead, the resources are fixed: we have a stable team with people who know each other well and have worked together for a longer period of time. We try not to add or remove people,

because that leads to more complexity as the team needs to communicate and adjust to work with the new team members.

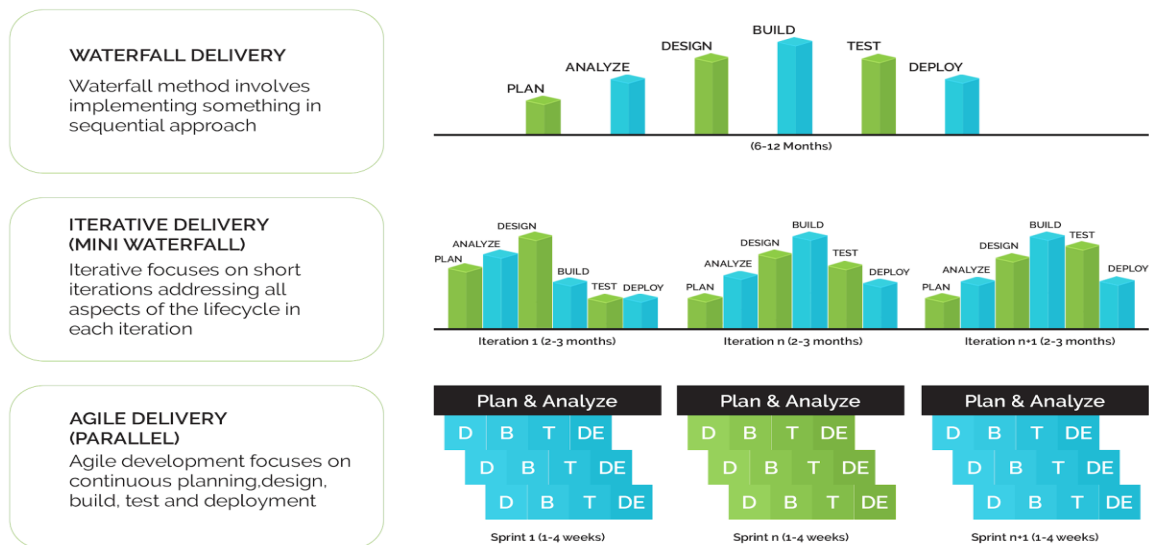
The time is also fixed in the time boxes. Within each time box, we try to deliver the maximum value (the maximum scope).

But since timebox and team size are fixed, the only thing that we change is the scope. If one user story can't be finished in the current sprint, we move it to the next sprint.

How to Combine Waterfall and Agile?

WATERFALL VS AGILE

LET'S COMPARE AGILE DELIVERY TO MORE TRADITIONAL METHODS OF DELIVERING 'PRODUCTS':



In answering this question, we want to start with a word of caution: combining waterfall and agile is an agile anti-pattern. The agile way of developing software has been designed to 'replace' traditional software lifecycles.

Especially the timeboxed approach and the focus on 'delivering value' instead of 'following a plan' do not match with the way waterfall projects work.

Having said that, we believe that any starting point on the path to agile is good. If an organization doesn't allow immediate abandonment of the waterfall process, running sprints inside the waterfall project can be the middle way to get started.

In waterfall, we typically agree on a planning and deadline for implementing 'all the requirements'. Once we have a deadline and we know what needs to be developed, we can start running iterations within that planning.

This means that we 'cut' all requirements into smaller parts. Each iteration of 2-3 weeks, we deliver parts of the requirements. The big challenge in combining a waterfall process with Agile is change. In the above description, we can see that all requirements are fixed.

If we deliver parts of the requirements each sprint, the big question is 'how do we deal with new insights and changing the requirements?'

There are some possible paths to address this questions:

1. Fix everything

If we fix everything upfront, we have mini-waterfall 'phases' instead of real sprints. In each iteration, we deliver a fixed set of

requirements, which has all been planned upfront. We try not to change the requirements from iteration to iteration, but just follow the plan. Any additional requirement is parked till the end of the project.

2. Changing scope

An alternative is to be flexible with the scope. While we have a fixed scope, we allow for change in between the iterations. When a product owner has a new requirement, he can add this to the priorities in his backlog.

Now instead of parking it till the end of the project, we can implement the new requirement. To do so, we replace that new requirement with an old requirement of the same size.

This means the workload is equal and we can still make our deadline. The tension we get with this is that by the end of the deadline, people who are still in the waterfall mindset, will recognize that not all requirements have been implemented.

We then need to plan for additional sprints and with that, move the deadline for release of the project. One of the biggest benefits of the agile way of developing software products is the focus on 'delivering the highest value within the timeboxes'.

This means in each sprint and it also means we aim for maximum value within the timebox for the whole project. If people are in the agile mindset, they will recognize that by the

end of the project timebox, what they will get is high value, provided that the project has been managed well.

This may mean that some of the initial requirements are not done yet, but that's ok, because they have been substituted by the product owner by 'better the requirements'.

People in the waterfall mindset expect everything to be completed by a certain date. If we run agile inside a waterfall project, with stakeholders 'thinking waterfall', this will inevitably lead to tension.

We therefore always recommend educating everybody involved on the way agile works and explaining how the combination of waterfall and agile deviates from 'pure agile'.

This will over time lead to support and understanding of the Agile way, which may allow for future projects to be run fully agile.

Being Agile Vs Doing Agile

Being agile is about the way you think and behave. Doing agile is about the way you execute. Understanding this difference is fundamental to changing oneself and a larger group of people. Find below the definition of Agile according to Google.

ag·ile

/ˈajəl/ 

adjective

adjective: **agile**

1. able to move quickly and easily.
"Ruth was as agile as a monkey"
synonyms: nimble, lithe, supple, limber, acrobatic, fleet-footed, light-footed, light on one's feet;
[More](#)
 - alert, sharp, acute, shrewd, astute, perceptive, quick-witted*antonyms*: clumsy, stiff, slow, dull
 - able to think and understand quickly.
"his vague manner concealed an agile mind"
2. relating to or denoting a method of project management, used especially for software development, that is characterized by the division of tasks into short phases of work and frequent reassessment and adaptation of plans.

Translations of agile

adjective

■ tangkas	agile, swift, dexterous, nimble, competent, deft
■ lincah	agile, lively, mercurial, vivacious, frisky, nippy
■ gesit	agile, nimble, mobile, adroit, spry, active
■ cerdas	intelligent, smart, clever, discerning, bright, agile
■ cekatan	deft, nimble, workmanlike, agile, handy, skillful
■ galir	loose, fluent, glib, articulate, agile, nimble

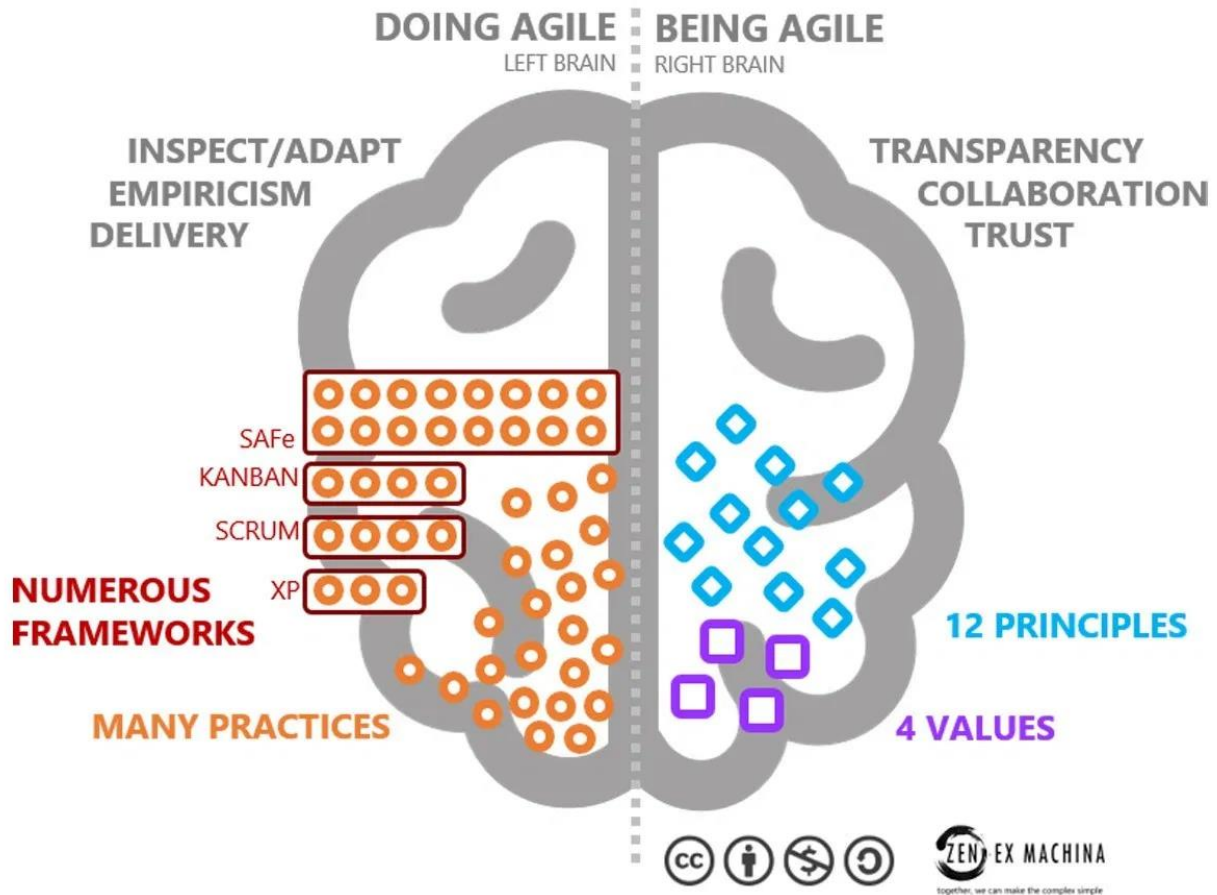


Being Agile

As Google says, being agile is:

- The ability to move quickly and easily.
- About the way we relate to other people.
- Decreasing hierarchy and power distance.
- We work on being transparent to each other.
- We are open and honest and share what's on our minds.
- We don't play political games.
- Trusting each other. Empowerment. A product owner indicates 'what' he wants to build and in what order of importance. He trusts the team to deliver, leaving them free to decide 'how' they will do it.

THE AGILE MINDSET



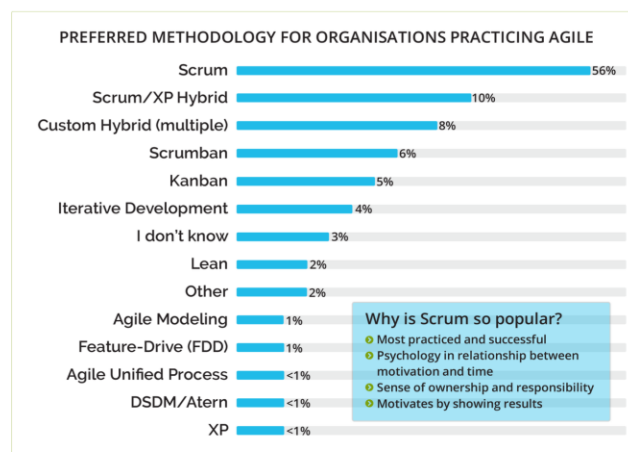
Doing Agile

This is the 'easy' part of Agile. We teach a team how the scrum framework works. We train someone to become a scrum master and that person helps the team run sprints, create Kanban boards and run retrospective.

There are many methods that support Agile working. Scrum is the most used Agile method. It consists of empowering a small and cross-functional team to solve a problem or achieve a goal. Scrum is most often used to manage complex software and product development, using iterative and incremental practices.

Scrum significantly increases productivity and reduces time to benefits relative to classic "waterfall" processes. Scrum processes enable organizations to adjust smoothly to rapidly-changing requirements, and produce a product that meets evolving business goals. But there are also other methods as shown below:

MANY WAYS TO PRACTICE AGILE!

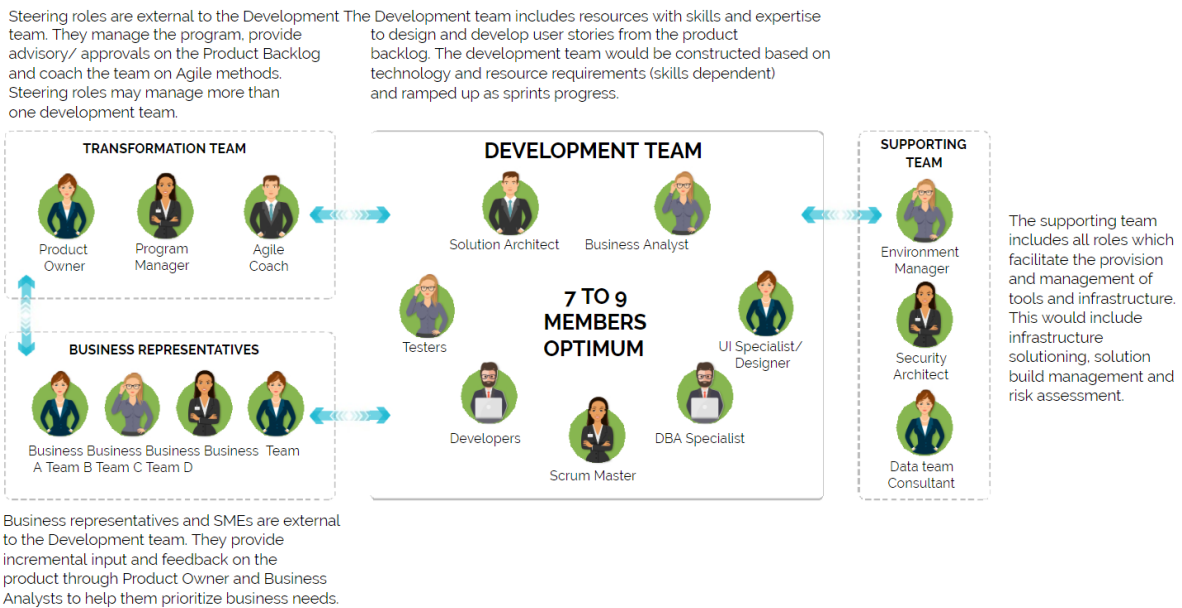


Roles in An Agile Organization

Roles and hierarchy are important topics for any agile organization. To understand how to reshape roles to make a more fluid and agile structure, we must look at different levels.

On the (software) agile team level, the below images give an overview of the different roles in a typical agile team:

TYPICAL AGILE TEAM STRUCTURES



1. Business Representatives

This group is often called 'stakeholders' in agile terminology. They are the people who drive a business or drive a department or other group inside the company. They know where the

business is going, they understand the market, clients and other dynamics.

They closely collaborate with the product owner to translate business ideas into technology. Business representatives and SMEs are external to the Development team.

They provide incremental input and feedback on the product through Product Owner and Business Analysts to help them prioritize business needs.

2. Transformation Team

Steering roles are external to the Development team. They manage the program, provide advisory/ approvals on the Product Backlog and coach the team on Agile methods.

Steering roles may manage more than one development team. We often use the name 'Agile center of excellence' for this team.

In most organizations, the transformation team has 1 strong driver in it from inside the organization. This is the 'Agile entrepreneur'. Entrepreneurs today are everywhere.

In the past, an entrepreneur was a person risking his 'job' and 'money' to start a new company.

Today, we have intrapreneurs, startups, venture builders, innovators. Many don't run the risk a 'traditional entrepreneur' has, but they create 'new stuff' in the safety of a paid job.

The agile entrepreneur is a special type of entrepreneur. It's someone who drives change and transformation, mostly in large enterprises.

This transformation eventually leads to new products and innovation. It also leads to a different culture, mindset and according behaviors.

The agile entrepreneur brings the 'startup vibe', removes hierarchy, empowers teams. Those teams are cross functional and iteratively create impactful outcomes for the company and themselves.

Without the agile entrepreneur, everyone sleepwalks his way to pension. People do 'business as usual' and tick the boxes.

The organization produces value as it has been doing for decades. Until something changes (Covid). Or a startup shakes things up.

Another important transformation team member is the agile coach: the role that ensures people work and think 'the agile way'. He has a very deep knowledge about the different tools and practices in Agile.

He can work at different levels within the organization as he is the spider in the web.

The agile coach helps the product owner to become a better product owner (for example writing good user stories, gathering requirements from business people).

The agile coach also helps the scrum master become 'independent'. Initially, an agile coach might take some of the responsibilities of a scrum master, but his goal is always to teach and empower the scrum master, until the agile coach is not needed anymore.

3. Development Team

The Agile development team consists of the people building the product. They are a cross functional agile team, meaning we have all the roles required to complete the work within a sprint, represented.

Overall, we always try to keep the team 'stable', meaning we don't rotate people, but we keep the same people in the team over a longer period.

Typically, an agile development team consists of programmers, designers, testers, architects and business analysts. Additionally, the team has a scrum master.

4. Supporting Team

Besides the technical specialists within the team, there are supporting roles. These are usually placed inside the 'supporting team'.

In some organizations or in some specific sprints, we could place the supporting role inside the development team if the role is required more intensely.

Typically, the supporting team consists of database administrators, devops people and security roles.

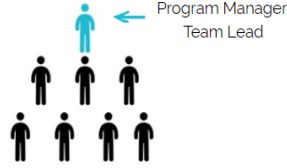
The supporting team provides the development teams with the right tools and infrastructure at the right time.

This means they also need to be aligned with the product roadmaps of the different development teams they support.

As most organizations move from a traditional 'waterfall based' model to an agile software development process, we have summarized the key roles in waterfall and agile in below image:

COMPARING ROLES IN SCRUM AND TRADITIONAL TEAMS

Traditional Teams



Agile Teams

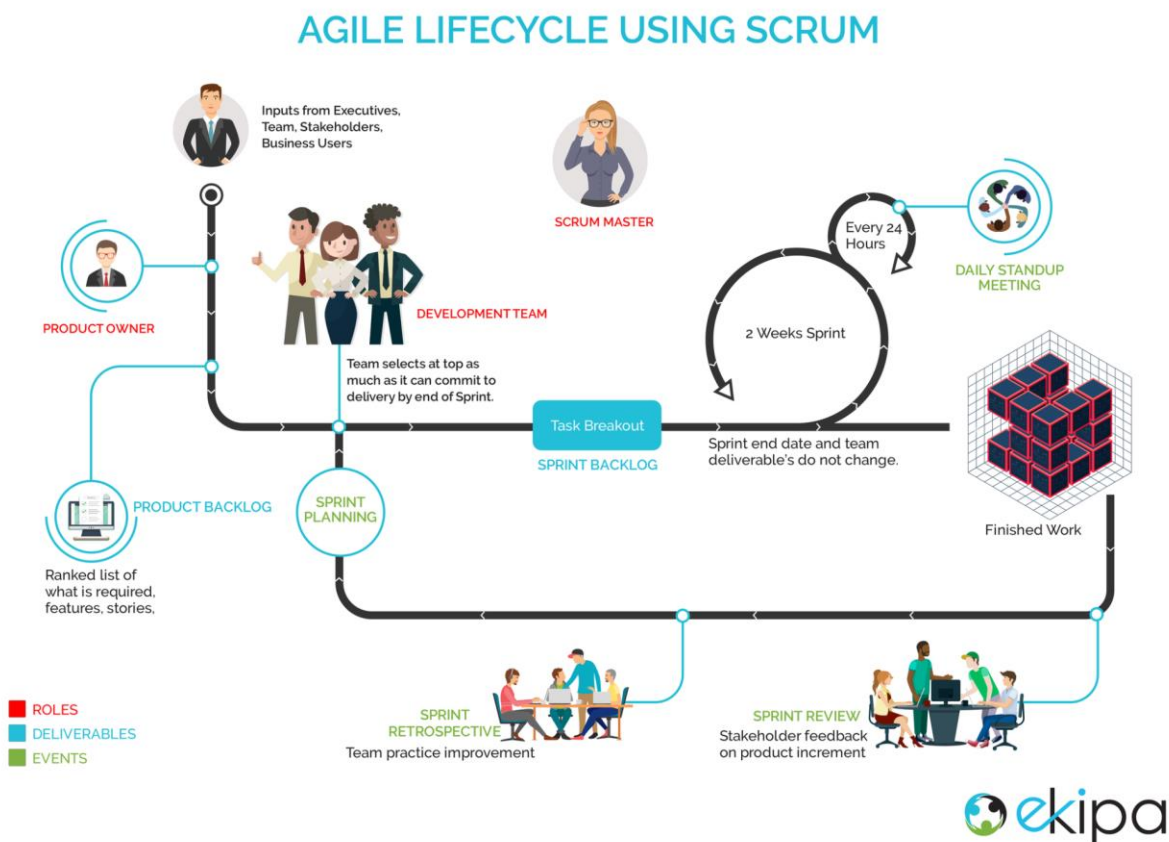


ROLE / FORUM	HOW DO WE KNOW THIS TODAY?	WHAT DOES IT MEAN IN AGILE?
Steering Committee	Senior Management Teams responsible for Steering individual program success	Steering Committee still required to steer individual program success – frequency and cadence changes
Program Sponsor	Decision maker for program priorities	Provide the overall strategic direction and controls the funding/budget for the program
Program Manager	Program and governance focus, responsible for program delivery	May be required although delivery accountability moves to Scrum Master (Development) and Product Owner (Product)
Scrum Master	N/A	Focus on team improvement, resolving blockers
Product Owner	N/A	Owns delivery of customer vision and value
Agile Coach	N/A	Key role to drive continuous adoption and improvement for agile teams throughout multi-year transformation
Team Members (Project vs Development team)	Part time, mostly specialists representing own area	Full time, cross functional, collaborating teams
Supporting Teams	Called in as and when necessary at key checkpoints and milestones e.g. sign offs	Actively involved from Sprint 0 and built into future sprints as needed depending on backlog priority



The Scrum Framework

Scrum is the most popular framework in Agile today. The power of scrum is its simplicity. It finds the right balance between 'up to you' and 'guidance'. Especially for starting agile teams, scrum gives people a clear answer on 'what to do'.



The framework consists of the following elements:

1. Roles

- Scrum master
- Product owner
- Development team

2. Events

- Sprint planning
- Daily scrum
- Sprint review
- Sprint retrospective

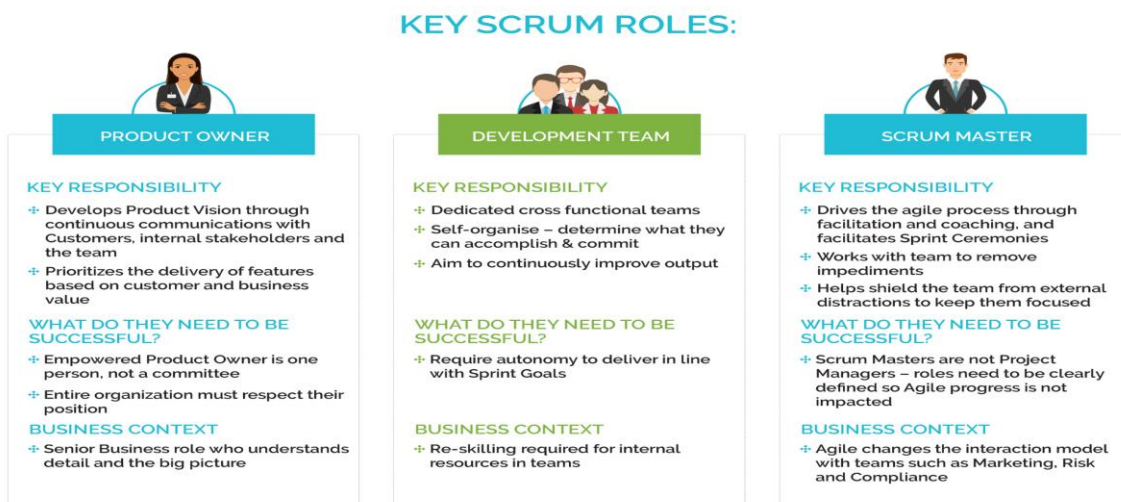
3. Artefacts

- Product backlog
- Sprint backlog
- Product increment
- Scrum roles

The Scrum Roles

In this overview, we have compared the key roles in a scrum team:

- Product owner
- Scrum Master
- Development team



The Scrum Master

The Scrum Master is responsible for upholding agile values and principles for the delivery effort. The Scrum Master helps the team execute its agreed-upon delivery process and typically facilitates common team ceremonies, like daily stand-up meetings, planning meetings, demos, retrospectives, and so on.

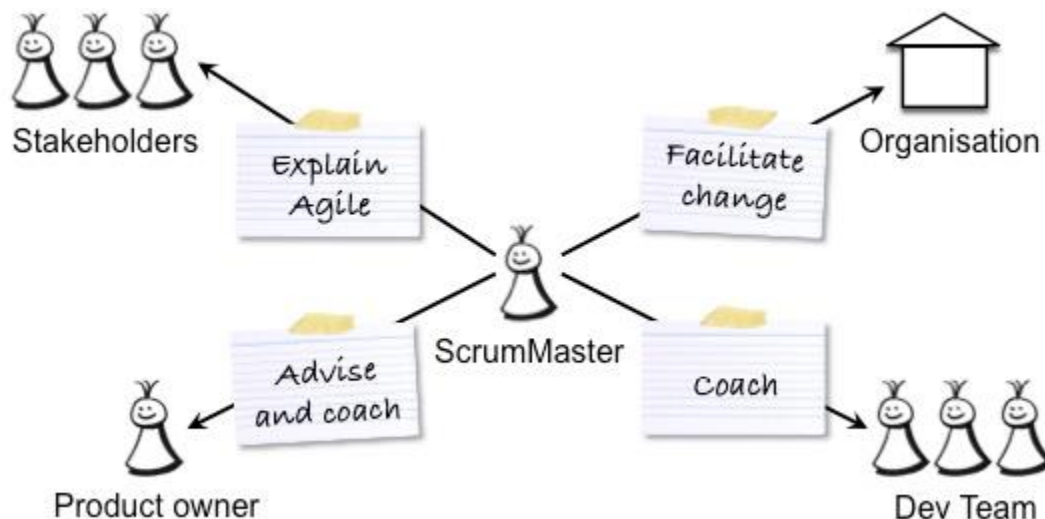
The Scrum Master biases the team toward action and delivery and stretches the team to continuously improve, hold each other accountable, and take ownership of the way the process works.

AS A SCRUM MASTER YOU



Scrum masters are the spiders in the agile web. They coach the development team and remove their impediments. They spread scrum throughout the company.

They teach customers how to work with their company using scrum. They advise the product owner how to collaborate best with all people involved. In my trainings, I always show the below image from Roman Pichler, which shows these responsible in a simple and clear way:



What Makes A Good Scrum Master?

People who want to become true scrum masters in their organization, need to understand the importance of their role. They need to realize that not only should they thoroughly understand the framework and gain experience using it.

The main thing you're looking for is 'empathy'. As the scrum master needs to interact with many different people at different levels, he

(or maybe even better: she) should be good at understanding what drives each person.

Empathy means the ability to put yourself in the shoes of another person. Through empathy, a scrum master creates positive relationships and is able to make scrum work at different levels.

The second thing is training and coaching skills. The scrum master should be able to do training at different levels to make people understand the scrum framework and the roles. He should also be a good coach.

This is important because a team lead hands out tasks. A team lead solves problems for team members. But a scrum master coaches people to solve their own problems.

A scrum master coaches the team to self-select the work they can commit to and then helps them do it. The scrum master of course also needs to be a strong team player.

She needs to relate to many people as the 'spider in the web' (product owner, stakeholders, developers, designers, maybe even owners). She needs to be a strong problem solver.

The Scrum master gets the team 'unstuck' whenever they get stuck on anything, so they can finish their commitment. The last thing is understanding of technology.

It helps a scrum master if he understands the world of a developer. He will gain respect and can more easily help solve impediments. At the same time it can become an impediment.

The scrum master may do coding himself or prescribe solutions to developers. So we would rather have an empathic person who gets everyone into the scrum 'flow' through personal interaction, training and coaching, than someone who can code.

The Product Owner

The Product Owner is the person who most considers the mission or business value of the solution being developed. They are responsible for maximizing the return on investment for the development effort, and they speak for the interests of the users.

Minimum Set of Visualizations

One of the statements we have heard often in Telkom is: it's not clear what product we are building, for whom and what the product roadmap is. The answer to this sits with the product owner.

As a product owner, we need to ensure that (new) team members, business people and leaders can quickly understand **WHAT we are building, for WHOM and HOW.**

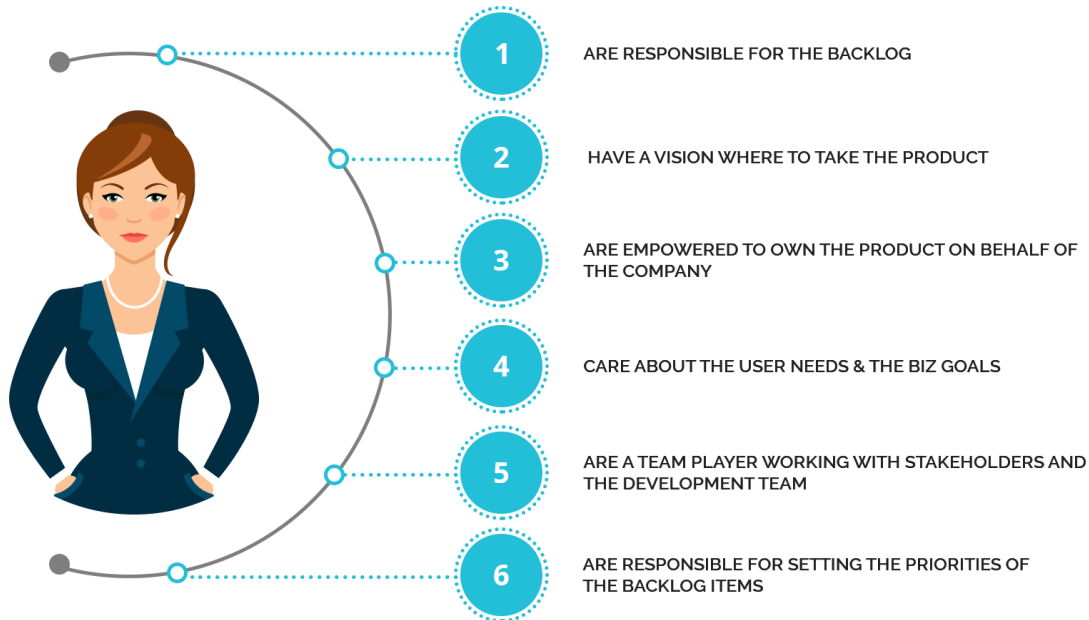
Accountabilities of A Product Owner

The product owner is typically a key stakeholder. Part of the product owner responsibilities is to have a vision of what he or she wishes to build, and convey that vision to the scrum team.

This is key to successfully starting any agile software development project. The agile product owner does this in part through the product backlog, which is a prioritized features list for the product. The product owner is commonly a lead user of the system or someone from marketing, product management or anyone with a solid understanding of users, the marketplace, the competition and of future trends for the domain or type of system being developed. On high-level the main functions of this role are as follows

AS A PRODUCT OWNER YOU:

RESPONSIBLE FOR THE PRODUCT BACKLOG AND MAXIMIZING THE PRODUCT ROLE.



- Define the features of the product
- Decide on release date and content
- Be responsible for the profitability of the product (ROI)
- Prioritize features according to market value
- Adjust features and priority every iteration, as needed
- Accept or reject work results

On a broader level the Product owner or POs day by day jobs are explained in the below image.

AS A SCRUM PRODUCT OWNER YOU...

...hold the **vision** for the product on behalf of the business, the customer, and the user.

...represent the **interests of the business** to the team.

...represent the product and the team to the business.

...communicate with **stakeholders** regularly.

...do not hold the role of scrum master.

...do not do implementation work.

...write **user stories**.

...help others write user stories.

...understand the **business value** of each user story.

...assign numeric business value to each user story.

...**prioritize** the user stories into a strictly ordered backlog.

...identify **acceptance criteria** for each story.

...collaborate with the rest of the team to create the team's **definition of done**.

...**accept** or **reject** completed work, determining whether it has met the acceptance criteria.

...arbitrate **conflicting requirements** from stakeholders.

...do not tell the team how to do the work.

...provide the **information** that the team needs to estimate each story.

...make yourself **available** to answer the team's questions about requirements and business value.

...do not estimate stories—that's the team's purview.

...make the call in the rare instances when a sprint needs to be terminated early.

...clarify requirements for the team.

...Lead the first part of the **sprint planning** meeting.

...Lead the **story time** meetings.

...gather **feedback** from stakeholders at the sprint review.



Development Team

Development team is the other members of the team, such as testers, developers, and designers. They are the people who, collectively, work together to deliver value on the project. They carry a diverse set of skills and expertise, but they are happy to help out in areas that are not their specialty.

Team members, together, take collective responsibility for the total solution, rather than having a “just my job” outlook. Team members do whatever they can to help the team meet its sprint goal and build a successful product.

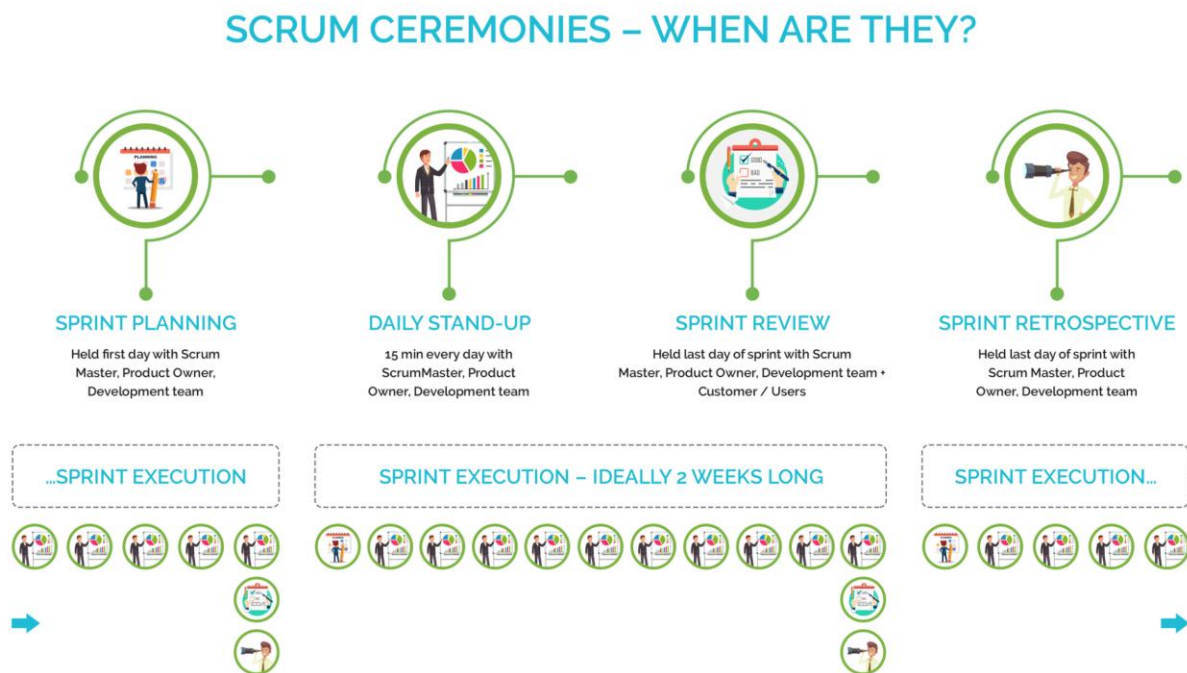
A SCRUM DEVELOPMENT TEAM...

Responsible for delivering a potentially shippable increment of working software.



Scrum Events

The event, sometimes called 'ceremonies' are the alignment meetings happening throughout a sprint. This image gives an overview before diving into them individually.



There are also some general rules for the overall sprint. Once a sprint starts:

- No changes come in that would affect the end goal and that would upset the team rhythm.
- Sprint goal does not change. So does quality standard.

- Sprint scope can be re-negotiated between the Product Owner and the rest of the team.
- Only the product owner has the right to cancel a sprint.
- Every Sprint is Four Weeks or Less in Duration
- There are no Breaks Between Sprints
- Every Sprint is the Same Length (ideally)
- The Intention of Every Sprint is “Potentially Shippable” product increment

Sprint Planning

SCRUM CEREMONIES: SPRINT PLANNING



The Purpose of Sprint Planning

The main purpose of this meeting is to bring the scrum team together and come up with a common goal to achieve in that sprint.

This will help us to refocus execution, minimize surprises, and guarantee overall higher quality code which is useful and valuable. This is the meeting whereby the Development Team agrees with the 'Product Owner' what the Sprint Goal is and which Product Backlog Items (PBI), that will contribute to that goal, will be attempted during the particular Sprint.

Who Attends the Sprint Planning?

1. Product Owner, both full and part-time
The 'Product Owner' needs to be there to set the sprint goal, agree the PBI to be attempted in business order and agree on the sprint.
2. Development Team (Required)
The whole Development Team needs to be there because all team members have to agree on the sprint plan. If any team member is absent, he/she may feel that the work has been 'foisted' upon them and will have reduced motivation to do it.
3. Scrum Master (Required)
The 'Scrum Master' needs to be there to listen to the conversations so that they can keep a 'fly on the wall' view of the team during the Sprint and remind any team member that may be straying from the plan.

Setting Up the Sprint Goal

Coming up with a Sprint goal is an essential part of every Sprint planning. It should be **specific** and **measurable** and the result of a discussion between the Product Owner and the Development Team.

In order to help you decide on the Sprint goal, ask yourself the following questions:

- **What do we want to achieve?**

This includes not only the isolated Sprint but also its contribution to the product as a whole.

- **How can we achieve that goal?**

The team must go through each item and make sure everyone understands its requirements and can foresee possible impediments. This contributes both to the estimation and the Sprint goal clarity.

- **How can we check whether we have achieved that goal?**

In other words, the definition of "Done", which is unique to your team and it's your job to make sure you're all on the same page.

Another important aspect to take into consideration is **whether the goal you want to set is challenging enough**. You don't want to create a goal that is too easy to achieve only to have a successful Sprint.

After all, Scrum is all about the continuous improvement from iteration to iteration, and setting the bar too low won't contribute to the product nor the team. Make it challenging enough to bring out the best out of your team.

Once your Sprint goal is set, it will provide guidance to the Development team and help them set priorities to achieve that goal.

Process & Practices

- All team member product owner and scrum master need to be present
- Understand the chick and pig rule in the scrum team.
- Remember the sprint planning meeting is run by scrum master and he has the highest authority.
- PO owner role in sprint planning meetings is only as a contributor, (s)he makes an active contributor in answering the question if the team has and explaining the user stories to the team.
- Scrum master is the timekeeper, leader, facilitator, and coordinator of the agile sprint planning meeting

The Scrum Master opens the planning meeting by presenting any relevant action items from the team's retrospective which was done previously.

Next, the product owner gives product or market updates so everyone is on the same page and has the broader context fresh in their minds.

After the debriefs, it's the product owner's responsibility to start the actual planning conversation. To get started, the product owner uses the team's average velocity (the amount of work typically completed in a sprint) to compile a suggested set of stories for the sprint, called the "sprint forecast", that maximizes value to the customer. The product owner should also consider these three factors:

- Public holidays, personal vacation, and team-wide events.

- Priority of stories in the backlog (ideally, they suggest the top-ranked items).
- How (and whether) that body of work will get the product closer to its final goal.

The entire team, including the scrum master, should listen carefully to the product owner's vision and explanations during the sprint planning meeting later, while planning the sprint's progression, team members should listen to one another and respect everyone's input.

- What was the Sprint Goal and why he/she chose it?
- Explain why it is important for the product and for the business as a whole?

Daily Scrum

HOW TO DO A DAILY STAND UP

What is a Daily Stand-Up ?

15 minute daily meeting for each of the Development team members to answer three questions...



What did I do Yesterday?



What will I do Today?



Do I have any Impediments?

Dos and DON'Ts

YOU SHOULD ALWAYS:

- ✓ Time box the meeting to 15 mins.
- ✓ Ensure the Scrum Board is updated before the meeting.
- ✓ Ensure the Product Owner acts as an observer.
- ✓ Conduct the Daily Stand-up in front of the Scrum Board.
- ✓ Ask team members to stick to the facts and not to go into in-depth discussions.
- ✓ Note any topic that needs more discussion offline, and any impediments to be cleared.
- ✓ Invite support groups (i.e. Infra teams, architecture, security) based on the stage of the sprint and release.

TRY NOT TO:

- ✗ Wait around for your team — always start your meeting at the set time.
- ✗ Introduce new topics which may divert attention away from the 3 questions.
- ✗ Start solving problems during the Daily Stand-up, but agree to have an offline discussion instead.
- ✗ View the Daily Stand-up as a "status report meeting".
- ✗ Be regularly absent during the Daily Stand-up.
- ✗ Abandon team communication throughout the Sprint in favour of the Stand-up.
- ✗ Conduct Stand-up meetings without the Scrum Board.



What happens in the daily scrum?

The main objective of this meeting is to create alignment and visibility among the team members about how much work is finished and how much is left to finish during the sprint.

This meeting is also to align team members where they are in terms of achieving the sprint goal(s). The team also assesses any risks to their Sprint commitment.

Who attends the daily scrum meeting?

1. Development Team (Required)
2. Other team members in case there is any dependency (Optional)
3. Technical Experts (Optional)
4. Scrum Master (Optional)
5. Product Owner (Optional)

How should we conduct daily scrum meetings?

All the development team members gather at a fixed time, fixed place everyday. This meeting is strictly time-boxed to 15 minutes. During the daily scrum, each team member answers the following three questions:

1. What did you do yesterday?
2. What will you do today?
3. Are there any impediments in your way?

All the team members talk about the above questions one by one and they move their user story/task cards to different columns (Development/Testing/Deployment/Done) on the scrum board.

Any impediment that is raised during the daily scrum meeting should be solved by the Scrum Master. In case he is not able to solve the impediments himself then he/she makes sure that the impediment is resolved as soon as possible. Scrum Master should also make sure that all the team must be present during this meeting.

Things to Avoid During Daily Scrum

- No mobile phones/tablets/laptops should be allowed.
- This meeting should not exceed time limit of 15 minutes.
- We should not discuss solutions during the daily scrum meeting.
- There is no leader, so anyone can start.
- People who are committed (i.e. The Development team) to finish work should only speak.
- Product Owner should not speak or instruct the team.
- Other people can also join the meeting like sales/Marketing but they are not allowed to speak.

Sprint Review

Purpose of sprint review

The purpose of this meeting is not to show that your application works, but that it is useful and valuable. You as a team need to provide context & scenarios so that people can relate to the actual features.

One important thing that should not forget is that there should be some element of show and entertainment, because if people give up time every 2 weeks or so to come to your demo, make it a good experience for them. Most importantly never show anything that is not 100% done, ie. Something which doesn't satisfy the Definition of Done.

SPRINT REVIEW

What is a Sprint Review?

User Stories meeting the 'Definition of Done' are demonstrated to the Product Owner & End Users to...



Demonstrate work Delivered in Sprint



Seek feedback & Update Backlog



Accept or Reject Delivery

Dos and DON'Ts

YOU SHOULD ALWAYS:

- ✔ Demonstrate work from the user's point of view.
- ✔ Creatively demonstrate Stories in a way that shows end-to-end value and usefulness to the Sprint review team.
- ✔ Face the demonstration - remember the audience is seeing the work for the first time.
- ✔ Help the Sprint review team understand what is demonstrated by providing context and scenarios.
- ✔ Ask and take questions, interact and welcome all feedback.
- ✔ Rotate the presenter (e.g. one for each Product Backlog Item).
- ✔ Send the list of the features that will be demonstrated before the meeting.

TRY NOT TO:

- ✘ Give a PowerPoint presentation — try to use the actual product.
- ✘ Spend too much time on preparation.
- ✘ Demonstrate unfinished work - Never show anything that is not 100% done.
- ✘ Allow observers to disrupt the review - take comments offline.
- ✘ Say yes to requests - add the requests as Product Backlog Items to the Product Backlog instead.



Process & Practices

1) Opening

The Product Owner opens the review with a warm welcome to all. And instead of jumping into the review, I recommend the Product owner present a very brief overview of:

1. *What was the Sprint Goal and why he/she chose it?*
2. *Explain why it is important for the product and for the business as a whole?*

After this the Product owner can invite the Scrum Master for facilitating the demo.

2) Agenda

The Scrum Master can start giving a brief on:

- *Telling the story of the sprint: How did it go? Did you have level 3 support incidents? (We handle those in a support*

work time box) Was anyone sick? New team members? Anything else important.

- *Give a status of the sprint and an overview of which stories were finished and which ones weren't.*
- *What will be demonstrated?*
- *What will not be demonstrated?*

To avoid a Scrum Master just reading a list of items that participants won't be able to follow, display something on the monitor or projector being used.

I always recommend sending a proper agenda along with the invite for Sprint review which gives the attendees a clear idea about what will and what will not be demonstrated for the review. Each person can then intelligently decide whether to attend or not based on what will be shown. Because we must respect time which is equally important for all.

3) Demo New Functionality

During this portion of the review, proceed down the list of items you've previously shown meeting participants. Keep in mind that the purpose of the sprint review is to solicit feedback.

There is no hard rule about who gives the demo. In some cases, a product owner will operate the keyboard. I'd recommend doing that in a review with particularly challenging stakeholders.

Other times, though, team members will demonstrate the specific product backlog items they worked on. So experiment to find the one that works best for your team.

Yet another important thing to be noted is that before the demo, make sure you are ready with everything you need for the demo, for Eg. Environment, Test Data etc.

4) Review-Rating and Discuss Key Occurrences

After all completed product backlog items have been demonstrated, the Scrum Master can ask the Business to give a rating for the sprint out of 10.

You can also make it more creative by providing them with Smiley cards with different emotions. After this, discuss key events or problems that occurred during the sprint.

This discussion could be facilitated by Scrum Master. Make sure we have one person always there to note down the feedback. I would always recommend to take up that monkey. He/she can then send these Review Comments to all via mail as a matter of information.

5) Present Upcoming Product Backlog Item

The final item on a sprint review agenda should be a discussion of the next work on the product backlog. Because even though the purpose of the sprint review is to acquire feedback on the work of the current sprint, this will often influence what will be worked on in subsequent sprints.

6) Closing Ceremony

Now the Scrum Master can simply wrap up by thanking everyone for participating. Appreciation is one key thing which we normally forget to do but will have a great impact.

Consider appreciating/thanking the team as a whole which will motivate them to move forward. Also you can remind everyone when and where the next review will be held.

What's Next ?

Maybe you can take a break for 5-10 minutes and come back to have your Sprint Retrospective. We recommend to have it only within the Scrum Team (Dev Team, PO, SM).

Sprint Retrospective

The Sprint Retrospective is an opportunity for the Scrum Team to inspect itself and create a plan for improvements to be enacted during the next Sprint. The Sprint Retrospective occurs after the Sprint Review and prior to the next Sprint Planning.

This is at most a three-hour meeting for one-month Sprints. For shorter Sprints, the event is usually shorter. The Scrum Master ensures that the event takes place and that attendants understand its purpose. The Scrum Master ensures that the meeting is positive and productive.

The Scrum Master teaches all to keep it within the time-box. The Scrum Master participates as a peer team member in the meeting from the accountability over the Scrum process.

HOW TO RUN A RETROSPECTIVE

What is a sprint Retrospective?

Encourages continuous improvement.
The Development team meet at the end of each Sprint to discuss..



What worked well?
Continue doing



What didn't work well?
Stop doing



What should be improved?
Start doing

Dos and DON'Ts

YOU SHOULD ALWAYS:

- Have an independent and unbiased facilitator
- Make the meeting interactive
- Conclude the Sprint Retrospective only when each Development team member has had an opportunity to voice their opinion, and improvements have been recorded
- Capture feedback and action items, including owners and due dates
- Ensure you are open and honest with feedback, in a constructive way

TRY NOT TO:

- Be ill-prepared-poor execution will result in an ineffective Retrospective
- Be too formal-an effective retrospective result in decisions and action items. It should be interactive and involved all members.
- Expect the Scrum Master to provide answers - the Scrum Master should steer rather than lead conversations towards continuous improvements
- Focus on improvements alone also celebrate achievements and what was done well



The purpose of the sprint retrospective is to:

- Inspect how the last Sprint went with regards to people, relationships, process, and tools;
- Identify and order the major items that went well and potential improvements; and,
- Create a plan for implementing improvements to the way the Scrum Team does its work.

The Scrum Master encourages the Scrum Team to improve, within the Scrum process framework, its development process and practices to make it more effective and enjoyable for the next Sprint.

During each Sprint Retrospective, the Scrum Team plans ways to increase product quality by improving work processes or adapting the definition of "Done", if appropriate and not in conflict with product or organizational standards.

By the end of the Sprint Retrospective, the Scrum Team should have identified improvements that it will implement in the next Sprint.

Implementing these improvements in the next Sprint is the adaptation to the inspection of the Scrum Team itself. Although improvements may be implemented at any time, the Sprint Retrospective provides a formal opportunity to focus on inspection and adaptation.

Scrum Artifacts

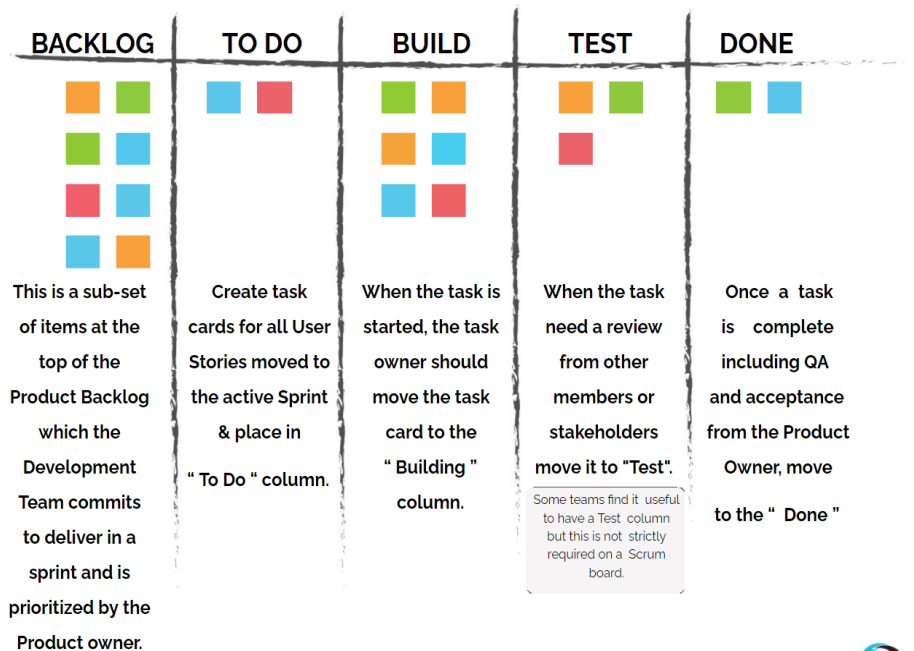
Artifacts defined by Scrum are specifically designed to maximize transparency of key information so that everybody has the same understanding of the artifact.

There are 3 main Artifacts in Scrum:

1. Product Backlog
2. Sprint Backlog
3. Increment

How to create a scrum board

A Scrum Board is a visual aid that allows teams to manage Sprint Tasks



Product Backlog

The Product Backlog is an ordered list of everything that is known to be needed in the product. It is the single source of requirements for any changes to be made to the product.

The Product Owner is responsible for the Product Backlog, including its content, availability, and ordering. A Product Backlog is never complete.

The earliest development of it lays out the initially known and best-understood requirements. The Product Backlog evolves as the product and the environment in which it will be used evolves.

The Product Backlog is dynamic; it constantly changes to identify what the product needs to be appropriate, competitive, and useful. If a product exists, its Product Backlog also exists.

The Product Backlog lists all features, functions, requirements, enhancements, and fixes that constitute the changes to be made to the product in future releases.

Product Backlog items have the attributes of a description, order, estimate, and value. Product Backlog items often include test descriptions that will prove its completeness when "Done".

SCRUM ARTEFACTS: PRODUCT BACKLOG

- ▶ The Product Backlog is a prioritized list of all the requirements for a particular product
- ▶ This 'product' could be anything from a CRM system to an HR initiative – anything that can be built iteratively
- ▶ The Product Backlog is owned and prioritized by the Product Owner
- ▶ Items on the Product Backlog can be re-prioritized, added and removed at any time
- ▶ Items at the top of the Product Backlog are detailed and ready to work on, whereas items lower down the Product Backlog can be higher level
- ▶ Items lower down the backlog will get refined as we get closer to them, this Product Backlog refinement happens in parallel to sprint delivery

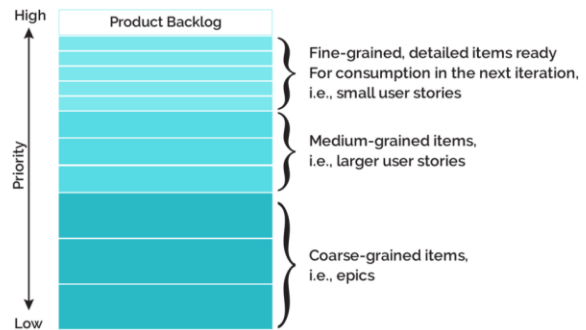


FIGURE 3.1 Product backlog prioritization determines the level of detail



As a product is used and gains value, and the marketplace provides feedback, the Product Backlog becomes a larger and more exhaustive list.

Requirements never stop changing, so a Product Backlog is a living artifact. Changes in business requirements, market conditions, or technology may cause changes in the Product Backlog.

Multiple Scrum Teams often work together on the same product. One Product Backlog is used to describe the upcoming work on the product. A Product Backlog attribute that groups items may then be employed.

Product Backlog refinement is the act of adding detail, estimates, and order to items in the Product Backlog. This is an ongoing process in which the Product Owner and the Development Team collaborate on the details of Product Backlog items.

During Product Backlog refinement, items are reviewed and revised. The Scrum Team decides how and when refinement is done.

Refinement usually consumes no more than 10% of the capacity of the Development Team. However, Product Backlog items can be updated at any time by the Product Owner or at the Product Owner's discretion.

Higher ordered Product Backlog items are usually clearer and more detailed than lower ordered ones. More precise estimates are made based on the greater clarity and increased detail; the lower the order, the less detail.

Product Backlog items that will occupy the Development Team for the upcoming Sprint are refined so that any one item can reasonably be "Done" within the Sprint time-box.

Product Backlog items that can be "Done" by the Development Team within one Sprint are deemed "Ready" for selection in a Sprint Planning. Product Backlog items usually acquire this degree of transparency through the above described refining activities.

The Development Team is responsible for all estimates. The Product Owner may influence the Development Team by helping it understand and select trade-offs, but the people who will perform the work make the final estimate.

Monitoring Progress Toward Goals

At any point in time, the total work remaining to reach a goal can be summed. The Product Owner tracks this total work remaining at least every Sprint Review.

The Product Owner compares this amount with work remaining at previous Sprint Reviews to assess progress toward completing projected work by the desired time for the goal. This information is made transparent to all stakeholders.

Various projective practices upon trending have been used to forecast progress, like burn-downs, burn-ups, or cumulative flows. These have proven useful. However, these do not replace the importance of empiricism. In complex environments, what will happen is unknown. Only what has already happened may be used for forward-looking decision-making.

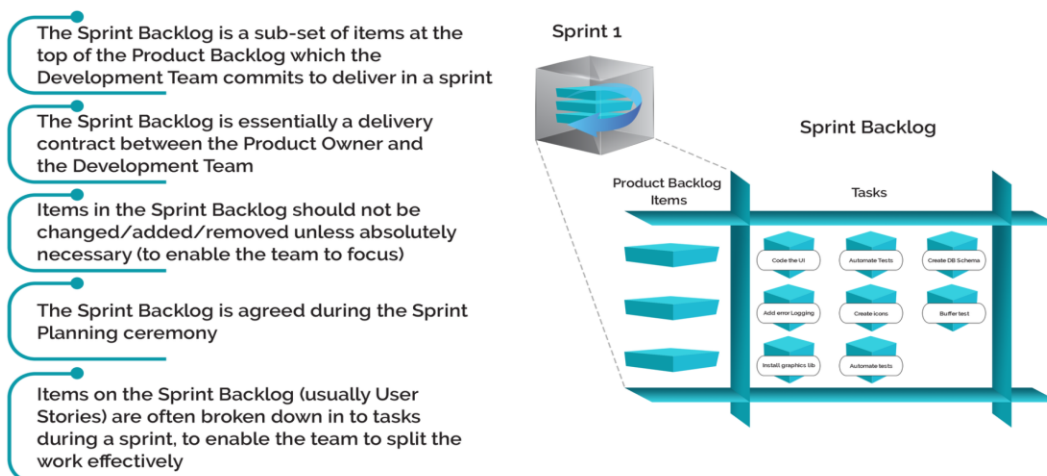
Sprint Backlog

The Sprint Backlog is the set of Product Backlog items selected for the Sprint, plus a plan for delivering the product Increment and realizing the Sprint Goal.

The Sprint Backlog is a forecast by the Development Team about what functionality will be in the next Increment and the work needed to deliver that functionality into a "Done" Increment.

The Sprint Backlog makes visible all the work that the Development Team identifies as necessary to meet the Sprint Goal. To ensure continuous improvement, it includes at least one high priority process improvement identified in the previous Retrospective meeting.

SCRUM ARTIFACTS: SPRINT BACKLOG



The Sprint Backlog is a plan with enough detail that changes in progress can be understood in the Daily Scrum. The Development Team modifies the Sprint Backlog throughout the Sprint, and the Sprint Backlog emerges during the Sprint.

This emergence occurs as the Development Team works through the plan and learns more about the work needed to achieve the Sprint Goal. As new work is required, the Development Team adds it to the Sprint Backlog.

As work is performed or completed, the estimated remaining work is updated. When elements of the plan are deemed unnecessary, they are removed.

Only the Development Team can change its Sprint Backlog during a Sprint. The Sprint Backlog is a highly visible, real-time picture of the work that the Development Team plans to accomplish during the Sprint, and it belongs solely to the Development Team.

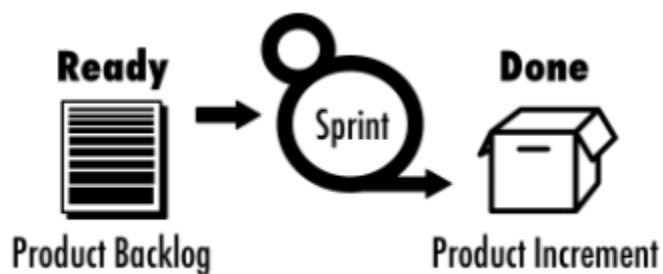
Monitoring Sprint Progress

At any point in time in a Sprint, the total work remaining in the Sprint Backlog can be summed. The Development Team tracks this total work remaining at least for every Daily Scrum to project the likelihood of achieving the Sprint Goal. By tracking the remaining work throughout the Sprint, the Development Team can manage its progress.

The Product Increment

The Increment is the sum of all the Product Backlog items completed during a Sprint combined with the increments of all previous Sprints.

At the end of a Sprint, the new Increment must be a working product, which means it must be in a usable condition. It must be in working condition regardless of whether the Product Owner decides to actually release it.



The Scrum Team needs to have consensus on what is considered to be an Increment. This varies significantly per Scrum Team, but team members must have a shared understanding of what it means for work to be complete. This is used to assess when work is complete on the product Increment.

The same understanding guides the Team in knowing how many Product Backlog items it can select during a Sprint Planning. The purpose of each Sprint is to deliver Increments of potentially releasable functionality.

Teams deliver an Increment of product functionality every Sprint. This Increment is usable, so a Product Owner may choose to release it immediately.

If the understanding of an increment is part of the conventions, standards, or guidelines of the development organization, all Scrum Teams must follow it as a minimum.

If it is not a convention of the development organization, the Scrum Team must define a definition of Increment appropriate for the product.

Each Increment is additive to all prior Increments and thoroughly tested, ensuring that all Increments work together.

As Scrum Teams mature, it is expected that their definitions of Increments expands to include more stringent criteria for higher quality.

Any one product should have a definition of Increment that is a standard for any work done on it.

Kanban

Kanban is a method for visualizing, managing and improving a workflow. It comes from Japanese Lean Manufacturing, but is now increasingly used in services, knowledge, and software development industries.

Is Kanban Agile?

Strictly speaking, no. Kanban started way before Agile and came from manufacturing rather than software development. Many of the underlying principles of Kanban are similar to Agile principles however, and many Agile coaches use Kanban as one of their tools. Visualization of work, collaboration/communication, feedback loops (inspect/adapt), experimentation, empowering and self-organized teams are concepts shared across both Agile and Kanban.

What are the fundamental Kanban principles?

There are 4 Kanban principles:

1. Start with what you do now.
2. Agree to pursue incremental, evolutionary change.
3. Respect the current process, role, responsibilities and titles.
4. Encourage acts of leadership at all levels.

When Should I Use Kanban?

An Agile framework like Scrum is very useful for product/software development teams where the work changes frequently, there is a

high degree of innovation and the team is working (usually) on a single product/system.

Kanban is useful for teams that have repetitive work that doesn't change much from week to week, day to day. Examples might be a call centre, recruitment, purchasing departments.

Work is repeated and high volume. Kanban can be used to improve flow by identifying and removing bottlenecks (where work gets bunched up and slows down) and blockers (which stop the work and prevent it from moving to the next stage in the process).

Advantages of Kanban

Kanban is easy to start and to learn. "Start from where you are". You don't have to create new roles or change the current way of working.

Teams can pick up new ways of thinking gradually and continue to improve once they understand and apply the principles. Kanban has simple metrics so you can visualize and monitor your work improvements.

5 Properties/Practices of Kanban

There are 5 core properties of Kanban

1. Visualize the workflow
2. Limit WIP
3. Manage flow
4. Make Process Policies Explicit
5. Improve Collaboratively

5 CORE PROPERTIES

1. Visualize the workflow
2. Limit WIP
3. Manage flow
4. Make Process Policies Explicit
5. Improve Collaboratively



How Do I Start To Use Kanban

1. Get your team to map your existing flow/way of work, exactly as it currently is on a Kanban board.
2. Ensure that your team understands the Kanban basics.
3. Each day, have a standup to get your team to move cards, identify bottlenecks/blockers and consider how to improve the flow
4. Empower the team to try new things, change the flow, to reduce bottlenecks
5. Use simple metrics to monitor how these changes improve the flow (or not)

Sample Kanban Board

Pool of Ideas	Feature Preparation		Feature Selected	User Story Identified		User Story Preparation		User Story Development		Feature Acceptance		Deploy ment	Delivered
	In Progress	Ready		In Progress	Ready	In Progress	Ready	In Progress	Ready				
Epic 431													Epic 294
Epic 478	Epic 444	Epic 662	Epic 662			Story 115-09	Story 115-09	Story 115-09	Story 115-09	Epic 401	Epic 609	Epic 094	Epic 386
Epic 552						Story 115-09	Story 115-09	Story 115-09	Story 115-09	Epic 468	Epic 577	Epic 270	Epic 419
Epic 439	Epic 589		Epic 302	Story 115-09	Story 115-09	Story 115-09	Story 115-09	Story 115-09	Story 115-09	Epic 362		Epic 339	Epic 388
Epic 379	Epic 651		Epic 335	Story 115-09	Story 115-09	Story 115-09	Story 115-09	Story 115-09	Story 115-09			Epic 521	Epic 287
Epic 287			Epic 512	Story 115-09	Story 115-09	Story 115-09	Story 115-09	Story 115-09	Story 115-09			Epic 582	Epic 274
	Discarded												
	Epic 606	Epic 606											
	Epic 606												

Policy
Business case showing value, cost of delay, size estimate and design outline.

Policy
Selection at Replenishment meeting chaired by Product Director.

Policy
Small, well understood, testable, agreed with PD & Team

Policy
As per "Definition of Done" (see...)

Policy
Risk assessed per Continuous Deployment policy (see...)



Making the Whole Enterprise Agile

Agile was originally devised in the software community. In 2001, a group of IT experts got together to create the [Agile Manifesto for 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 past 2 decades, many IT organizations moved to Agile teams, iteratively working on software product development. Recently, companies started adopting the Agile way of working to other functions. This is what is called 'business agility'. A good definition is provided by the [business agility consortium](#):

*Business Agility is the ability of an organisation to:
Adapt quickly to market changes - internally and externally -
Respond rapidly and flexibly to customer demands*

Adapt and lead change in a productive and cost-effective way without compromising quality

Continuously be at a competitive advantage

In most organizations, this agility is implemented by adopting ways of working that were invented in the software community.

Mostly people will adopt scrum or kanban for agile marketing, agile sales, agile finance, agile human resources or agile operations.

For us, the term business agility initially refers to a 'silo' or 'department' adopting the agile principles. They start working as a team in agile iterations or sprints of 1 or 2 weeks.

They create visibility and transparency by creating work items that can be written on 'sticky notes' and pasted to a board on the wall.

They will plan events similar to software teams: sprint planning, daily standup, sprint review and sprint retrospective.

Most teams will adjust the number, contents and cadence of these events.

The agile team may also create roles like product owner or scrum master, but in most functions this is not the ideal set of roles.

Teams can experiment with having a role like a business owner (who decides together with the team what the team will work on).

We can then add the role of Agile coach to ensure that the team adopts the right habits and agile mindset.

Within the team, the roles can be based on what existed before. If a team adopts kanban, kanban suggests starting with the roles that were already in place.

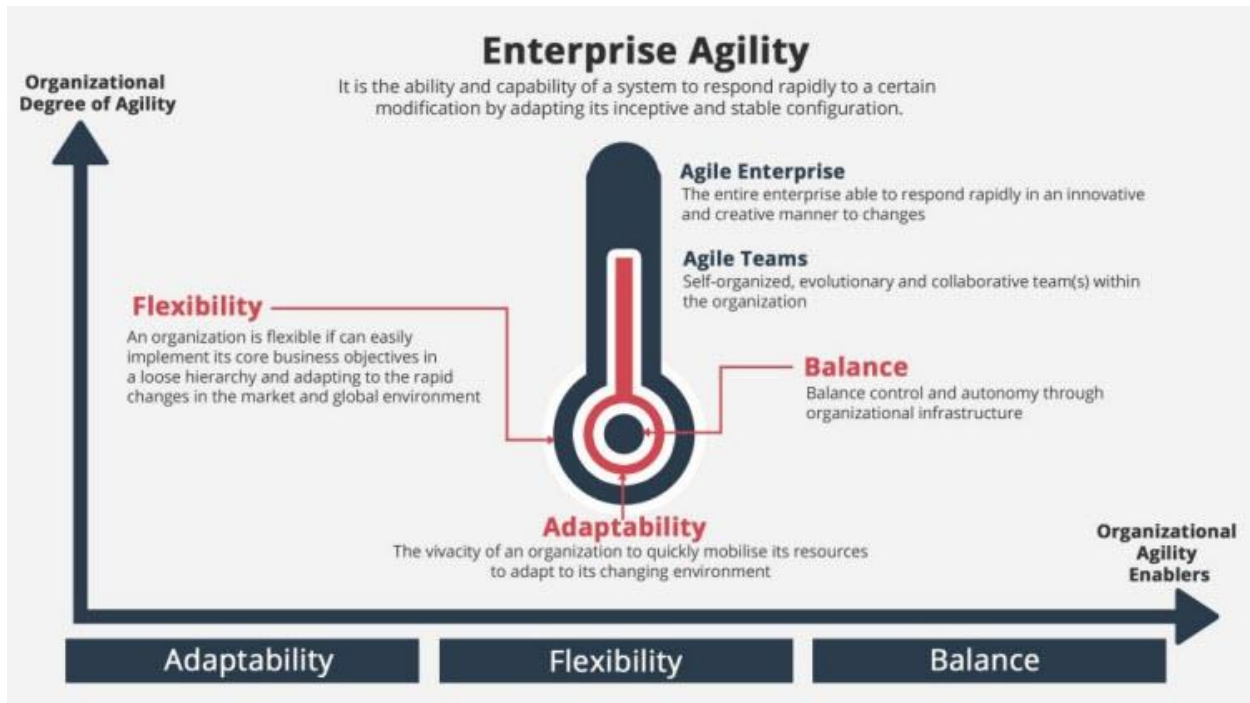
Enterprise agility essentially is the same as 'business agility', but in our perspective refers to adoption of Agile on a bigger scale.

Instead of 'doing agile' in specific departments, the goal is to remove the silos that have always existed in an enterprise (marketing, sales, finance, operations, etc).

Instead, we create 'cross departmental' teams, often called 'squads'. The squad works on a specific 'value stream' or 'customer service' or 'product'.

All the functions sit together and have a common goal. An interesting example of enterprise agile adoption is [the ING way of working](#).

[Modern analyst](#) shows the essence of enterprise agility in below image:



Modern analyst defines enterprise agility as:

Enterprise-level agility – *The enterprise-level agility involves the ability of the entire enterprise to respond rapidly in an innovative and creative manner to changes in the market conditions.*

It is the ability of the enterprise to facilitate dynamic decision making that supports the trending direction and any emerging competitive advantage.

The level of agility in an enterprise is contrary to that of the traditional organizations where there is adherence to sustained competitive advantage.

As can be seen from the diagram, the keywords in enterprise agility are adaptability, flexibility and balance. To reach enterprise agility,

we need to reshape the way we deal with 'power' and 'control' versus 'autonomy'.

We need to empower agile business teams to re-group and re-plan what they work on, on a continuing basis. This means that we must trust the enterprise agile teams to do 'the right thing'. We can add the role of agile coach to ensure that the team also does 'things right'.

How to Create an Innovation Engine?

It seems common sense to see 'startups' as independent entities, disrupting traditional businesses and bringing innovation.

But the past few years, enterprises have started adopting 'the startup way'.

While an enterprise can never BE a startup, they can adopt the principles that underlie startups. With this, they can create startups within their larger organization.

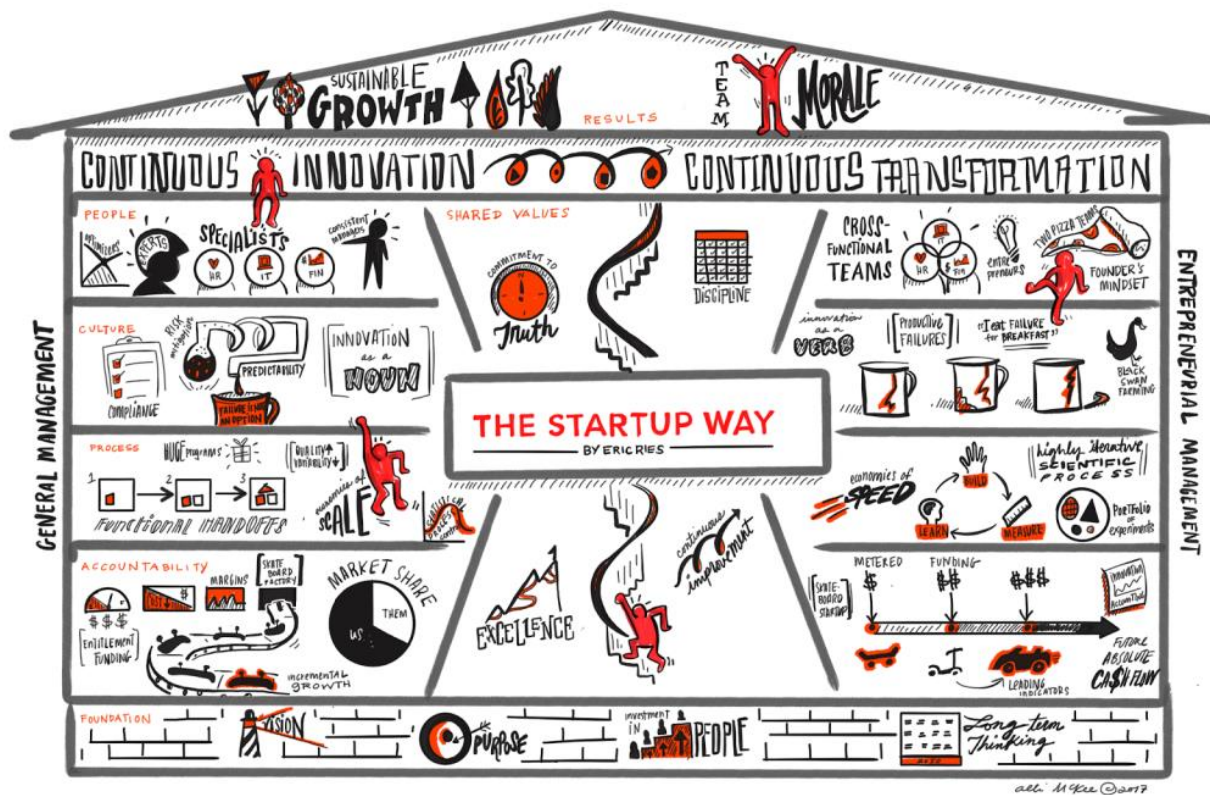
Either as atomic units inside the existing structure or (eventually) as separate entities outside that structure.

While the answer to the question 'how to do lean startup in an enterprise' is quite complex, there are some patterns. The below image is from 'the startup way', written by Eric Ries.

He identifies the main patterns that need to change from 'traditional management' to 'entrepreneurial management'.

Usually, enterprises will need to go through a transformation with a structured program to create corporate startups.

The building blocks of such transformation can be derived from the below main patterns.



1. From 'Specialists' to 'Cross Functional Teams'

Most organizations have teams of specialists, typically organized in departments. People see themselves as belonging to one department or function.

Usually they are hired for a role in that department and restrict themselves to the responsibilities laid out in their job description.

Startups function differently; in general, they need generalists in the early stages. People who can 'get things done' and don't look at what they are hired to do, but what needs to be done.

Typically, the best way to organize this in a traditional organization is in 'cross functional teams'. While people can still work on their main function, the team can tackle everything that needs doing.

A startup team can have a CEO, CTO, a CMO and then people who support the rest (finance, sales, development, design, etc).

2. From 'A Culture of Compliance and Predictability to A Culture Of Speed And Embracing Failure

Traditional management is based on compliance and predicting the future. Managers want to control what is happening.

They want to control finance, people, products, so their mutli-year plans work out as described in their original business case. For a predictable and mature business this may still work (although a startup might suddenly pass by to disrupt that predictability).

For a startup, this doesn't work. A startup is trying to find out what business is must go into. It is trying to find out who its customers are and what business model can work. Nothing in a startup is predictable until it has 'figured out' what product-market combination works.

To enable intrapreneurs to innovate, corporates must let go of the need to control things. Where traditionally, big business cases and long-term funding were the norm, we must let go.

Startup funding works in a staged approach. For the initial 'exploration' period, a startup gets some business angels to provide them with a small runway.

Typically 6-12 months with enough money for the entrepreneurs to sustain themselves, hire some initial people and fund their experiments.

They must report progress to their investors. This progress is based on learning, on initial feedback and Minimum Viable products which they used to learn what their users really want.

Once they have proven enough and achieved some positive outcomes, they can go to some professional investors to get their series A funding.

To get this funding, they must get positive feedback from their original investors. The new investors will want to hear stories around learning, initial traction, proven business models and other early stage results. Again, they will get sufficient funding to run through the next cycle of making their initial business work out and if they show outcomes, they can get series B, etc.

Now a corporate can adopt a similar method of funding. Instead of VC's, they can have an investment board representing leaders from various levels and functions to decide on the funding.

The internal startup teams must show progress and results to secure the next round of funding. They don't get the funding because they have friends, a long track record or the best business case; they get funding because they show results. This is what Eric Ries calls 'metered funding'.

3. From Functional Handoffs (Waterfall) to Build-Measure-Learn Cycles

In traditional project management, we have 'silos' or 'functional departments' that work on a project at different stages of its lifecycle. Once a group of people has finished working on a specific part, they hand it over to the next group.

If we look at digital product development, a group of analysts starts the work and develops requirements. They then hand it off to the designers who start creating pages and screens.

Once those are confirmed, they hand over to the programmers who in turn hand over to testers once they created the required software.

This is what the software industry calls 'waterfall'. Methods like Prince2 and ITIL all believe in 'predictability'. While in reality, projects nowadays are much less predictable than a decade ago.

Modern organizations adopt 'build-measure-learn cycle'. Instead of making requirements upfront, they run 'experiments' or 'cycles'.

They define what can be developed in a cycle of 2 or more weeks. They build and launch it to their users. That gives them feedback (data and quantitative inputs), from which they learn what really needs to be built for the users.

The paradigm shift here is big. In traditional project management, we assume we can define everything upfront. We assume projects are like building a house: the architect designs the whole structure, it's signed off and then the construction teams sequentially finish the house.

Built-to-spec. In the lean startup way of launching new products, we assume that we're moving in uncharted territory. We are developing something new and hence we don't yet know what the users or market really needs.

Therefore, we make hypotheses, based on the assumptions (what we think the users need). We validate those hypotheses before we really build anything. Only when we validate, we build the 'real thing'. This ensures we build the right thing and we avoid all the waist associated with building the wrong thing.

This paradigm shift requires leadership. We need to create a culture of experimentation. A culture in which people are okay with failure (because if we falsify our hypotheses, we're basically failing).

In which we prefer to learn that we should NOT build something or should KILL a certain project, before we invest our money and time into it.

4. From Accountability and Entitlement Funding to Metered Funding 'Startup Style'

In enterprises, we have strict procedures for funding projects. We require people to write business cases, which are then sent to some line manager and if he agrees, it will be sent to the finance department.

We usually spend months to get a business case approved. Once it's approved, the person applying for the budget is 'entitled' to the funding.

That entitlement might be for 6 or 12 months. Once a project is funded, it might get funded again at later stages, because the person applying has the right political connections or because the leader sponsoring the project does not want to kill it. That's what we call 'entitlement funding'.

In startups, funding happens in stages. A startup initially gets some 'angel investors'. Those are usually people using their personal funds to support a startup idea.

There is personal trust between the investor and the startup founders. The founders get for example 200.000 USD and with that money they must show results to the (future) investors.

Upfront, we define what we expect in terms of outcomes. In the early stages of a startup, 'outcomes' mean 'learning' (what users need, whether the product is viable, what the pain points are in the market, etc).

After learning, the outcomes become 'user acquisition' or 'members' or something similar. The team has learned enough to launch some initial version of the product and is now trying to get users to adopt the new product or service. In later stages, the outcomes are turnover or profit.

For each stage, the team will receive some funding. And securing funding for the next stage depends on the outcomes the team has shown as well as the relationship with the investors from the previous round.

The relationship also indicates whether the team is strong, whether the investor believed that he invested in the right people.

Now it's not easy to change this type of funding in a large enterprise. Depending on the industry, there are (government) regulations.

There are procedures in place to ensure that the people in the company invest the companies money (which is the shareholders money) wisely.

Essentially, all the regulations are made to ensure accountability with the people making investment decisions. To protect shareholders (which are private people or institutions in publicly traded companies) from their money vanishing in the wrong projects or pockets.

Becoming a modern company means that we need to adjust the attitude of the company towards risk. To change the way we look at investment and budgeting.

[Eric Ries](#) said that In Silicon Valley, the money you raise is yours. You can spend it on what you like with minimal oversight (especially in the early stages).

But Lord help you if you try to raise more money and you haven't made any progress. Seed-stage funding provides an excellent balance between risk mitigation and freedom to innovate.

The structure of the startup limits the total liability of the team to the total money raised. And it strictly limits the amount of time and energy the team has to invest in acquiring and defending its budget.

But at the same time, it creates a strong incentive to keep investors informed when there's something newsworthy to share so that they will want to continue investing and provide a positive reference to the next set of investors.

What is the Startup Way?

Innovation has been a big topic for centuries. Creating new products and services, improving processes and other things in a company are all important.

Today, because change and disruption are speeding up, the importance of innovation is different. Startups are popping up all over the world.

Entrepreneurs seek problems in the market and set out to solve them by inventing new products and services that didn't exist before. Technology enables these entrepreneurs to build new products faster, cheaper and easier than ever before.

The bigger companies are often left behind when startups suddenly disrupt their traditional business models. There are the obvious examples like Airbnb, Gojek, Uber, Facebook. But this happens everywhere in all industries today.

There are thousands of books and theories about innovation. Companies have built R&D labs, innovation labs, innovation programs for decades. In 'the startup way', big enterprises try to adopt the mechanisms used by startups. The startup way is a phrase coined by Eric Ries.

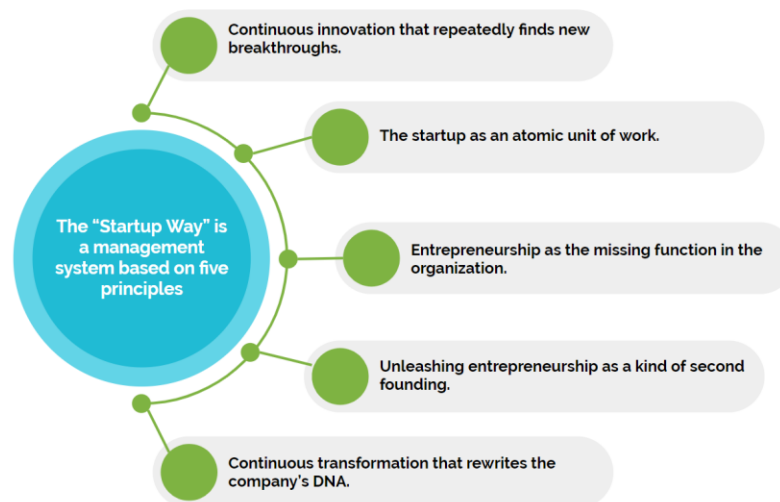
In a similar style, Dan Toma and friends wrote 'the corporate startup'. They describe how corporations can run innovation 'like a startup'. Now the important thing to note here is that an enterprise

can not 'become' a startup. A large company employs thousands of people. It has accrued habits, culture, regulations, hierarchies.

A startup is still very agile and everything can be changed overnight. The incumbent structures in a big company can't be changed overnight. But the central idea here is to see what **CAN** be changed. **HOW** can an enterprise step by step change those structures, learning from what works in startups.

In his book, Eric describes a set of principles. Those are good starting points for an enterprise to learn what can be changed, to define the direction innovation can take in a modern company.

5 principles of 'the startup way'



Continuous Innovation That Repeatedly Finds New Breakthroughs

This principle relates to the culture we have within our enterprise. A traditional company has lots of people who like 'status quo' and stick to 'how things have always been done here'.

They accept how things (don't) work. They accept that their current business is x and not y. In a modern, innovative company, people are continuously (stimulated to) looking for improvement and innovation.

The company runs programs and events to stimulate ideation everywhere (not only in a 'lab'). It provides support and resources for people to execute those ideas. It educates people continuously on the modern tools of innovation.

The Startup as An Atomic Unit of Work

Traditionally, innovation happened in 'labs' or a special department. In the startup way, we see the 'startup team inside the corporate' as the entity 'running' their innovation.

We gather a group of people in a team (a startup). Those people are all employed by the enterprise, but they are 'allowed' to work on their startup idea. The structure of the company is such that this startup has the freedom to pursue their startup idea similar to an outside startup. They get mentoring, funding, access to customers, access to existing tools and frameworks, access to talent, education. All the support structures are there to empower the startup team and help them towards success.

Entrepreneurship as The Missing Function in The Organization

Without entrepreneurship, innovation will not take place. We need entrepreneurs at different levels in the enterprise, both at the (top) leadership level as well as on the execution level.

We need to make 'entrepreneur' or 'intrapreneur' an accepted role in the company. There are [three levels of leadership](#) in modern, innovative companies.

3 types of leadership

- 1. Entrepreneurial leaders**
create value for customers with new products and services
- 2. Enabling leaders**
make sure the entrepreneurs have the resources and information they need
- 3. Architectural leaders**
keep an eye on the whole game board, monitoring culture, high-level strategy, and structure



An enterprise needs to create the right environment in order to identify and keep entrepreneurial talent. Most entrepreneurs will jump at an opportunity.

But if the enterprise doesn't allow them to work on their ideas (independently), they will leave the company to do it on their own.

The key is to create entrepreneurship as a 'normal function' in an enterprise, so we keep this talent and support them to launch their ideas.

[Excerpt from 'youexec':](#)

The entrepreneurial function also supports other functions in the organization in doing their work more effectively. Traditional management tools are focused on planning and forecasting.

Identifying and managing entrepreneurs requires a new style of leadership. It is particularly important to realize that 'entrepreneurship' is not some special quality possessed only by a few people.

In fact, you never know who the entrepreneurs are going to be; and even the non-entrepreneurs will benefit from this new way of working.

To take advantage of its latent pool of entrepreneurial talent, the organization has to make the entire employee base aware of the possibilities of entrepreneurship as a career path. This means meeting a series of challenges:

- *Creating space for experiments but with liability constraints.*
- *Learning to make investments on the basis of evidence, experimentation, and vision, not just ROI forecasts.*
- *Creating milestones that can work even when there's no accurate forecast.*
- *Providing professional development and coaching to help people get better at being entrepreneurs.*
- *Provide internal and external networks so that people know what it means to say, "I'm an entrepreneur."*

- *Recognizing that high risk and uncertain projects need a separate and rational way to attract talent.*
- *Creating new incentive and advancement systems.*

Unleashing Entrepreneurship as A Kind of Second Founding

The second founding is essentially ‘reinventing the organization’. We transform everything in order to embed the entrepreneurial function and continuous innovation.

We must look at what Ekipa calls the 5 building blocks of innovation:

LAP 5: ENTERPRISE TRANSFORMATION

Building blocks of the corporate startup engine:



Finance: Budgets, funding and risk



Governance: Metrics and milestones (to make investment decisions). How to select and form the right teams. searching versus executing. 3 horizons.



HR, mindset, culture, professional development and coaching for the entrepreneurial teams; incentives and career



Ecosystem: networking, matchmaking, learning



Leadership: who leads this and how? Growth board; committee. 3 types of entrepreneurial leaders

In ‘the first founding’, a company comes to life. As it grows, it establishes patterns, culture, structure, governance. The bigger the

company gets, the more ingrained all of this becomes. In the second founding we 'start all over again'.

We imagine our enterprise to recreate the way we fund projects and products. We recreate our culture, in alignment with the values of an innovative, modern enterprise.

We re-create hierarchical structures and the according leadership structure. Governance is 'broken down' and we shape the way we measure success of any innovation project. And we create a new ecosystem in which we operate, opening the doors of our company and collaborating with startups and intrapreneurs 'out there'.

Continuous Transformation That Rewrites the Company's DNA

This last principle relates to the idea that transformation doesn't have a beginning and an end. If we want to unleash innovation and entrepreneurship, we must keep reinventing everything we do.

We should keep transforming, because the market and world are not static today. Everything keeps changing, disruption will come from any direction all the time. And so our enterprise needs to keep moving along.

'The goal of all these tools and techniques is to move the organization from a state of continuous innovation to one of continuous transformations; an ongoing cycle of change that transforms not just a project or a team but the structure of the enterprise itself.

Every organization should have an active program of experimentation in new organizational forms and methods, populated by those who will become the founders of the next company-wide transformation.

These people will need a skill set similar to what is needed to build a new startup from scratch. They need career paths, accountability, and a rigorous training system.

This engine of continuous transformation should be seen as a permanent organizational capability, one with responsibility for the entrepreneurship function.' [Youexec](#).

The 5 building blocks above show the main things that must change. It's not sufficient to re-invent the organization once. We should see our enterprise structure and the 5 building blocks as an evolving structure.

The things we change are to be changed in a never ending cycle of 'build-measure-learn'. We keep looking at what needs improvement. We define a simple new version of that improvement.

We launch that, learn what is really needed and iterate until it works. And we keep doing that for everything in our enterprise (people, process, technology, structure, policies, etc). Never ending, continuous transformation.

5 Laps to Kick Start an Innovation Program

In Ekipa's perspective, a corporate startup is a team inside a larger enterprise, working together to pursue innovation (an improvement to an existing process, a new product, a new service).

The team is fully dedicated to that innovation and has the autonomy to execute as they see fit, with clear boundaries and accountability defined.

They have the freedom to spend the money invested in their startup, but they must show progress (learning, users, money).

So a corporate startup is not an enterprise operating as a startup. Rather, it is a structure for innovation. It is a way to spark strong ideas for improvements and future products.

The startup is the mechanism through which the innovation is executed. This mechanism is based on the way startups (outside) operate.

By looking at startups, enterprises can adopt (elements of) the way a startup gets funded, educated and supported.

How to Do Innovation in Large Enterprises

How to do innovation is a big question. There is no easy path to improve what we do or launch new products and services. The starting point is always finding the right balance between 'capabilities' and 'execution'.

Capabilities include: having leaders who understand how innovation 'works'; a clear vision and strategy to roll out innovation; the right type of funding (process); and education plus mentoring to support innovation teams.

Execution is where the rubber meets the road. This is where innovation teams generate ideas and start implementing them.

To get here, we need to organize ideation (through hackathons, competitions, innovation boxes, etc). And we need to identify the right people to work dedicated to innovation initiatives.

To find this balance, organizations need to run through what Ekipa calls 'laps' (they are not steps, because innovation isn't a linear process and so is a change program to become more innovative).

LAP 1: leadership awareness

Big question: WHY? WHAT?

- ✦ Session 1 with BOD to develop a high level vision for innovation
- ✦ Session 2 with C - 1 to create a high level roadmap for execution
- ✦ Creates alignment and shared vision to start transformation
- ✦ Identifies executive sponsor(s) to drive change
- ✦ Identifies barriers, challenges and opportunities for innovation



In the first lap, we need to identify who in the leadership team will drive innovation and entrepreneurship as a core function.

Who in the leadership team is supportive and who is skeptical.

In this lap we also need to define the 'why' of innovation; why do we want to innovate, where do we want to go and what are the things that will help us or hold us back?

We need to identify the right leaders at all three levels to start a long-term innovation program.

- ✦ Develop 'innovation thesis': what products, problems and markets?
- ✦ Portfolio overview: core business versus disruptive innovation
- ✦ Clarity on leadership principles and roles to drive innovation
- ✦ Vision on the 'sweet spot' for innovation
- ✦ Capabilities development plan (people, infrastructure, tools)
- ✦ Personal coaching for leaders to grow the entrepreneurial function

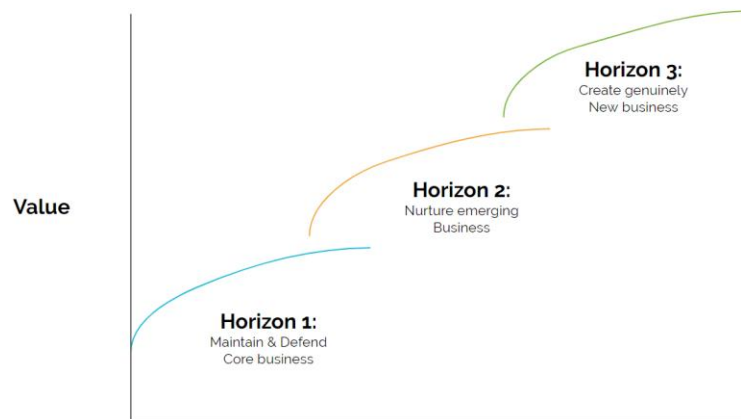


Once we have identified leaders at all three levels and we have some higher level direction, the leaders need to learn the traits of an entrepreneurial innovator.

Based on the high level direction, we also need to define what types of innovation we want and don't want (our innovation thesis). If we are a telco, we may want to develop new IoT solutions, but not start selling cars.

To develop the innovation thesis, we can look at the different types of innovation that [McKinsey defined in their 'three horizons'](#):

Prioritization: Mckinseys-three-horizons-of-growth



Once we know what type of innovation we want and we have the right leaders to support the innovation efforts as well as the innovation teams, we need to start pilots.

The first question is how to source ideas. There are many well-known formats that can be used, like hackathons, idea boxes, incubation spaces, idea competitions.

Once we have identified some initial innovators or innovation teams along with ideas, we need to have the right incubation process to help them get started.

This means we need to support the teams with the right knowledge (education) and mentoring (peers, leaders or external experts).

- ✦ 2 or more pilot teams, supported by an entrepreneurial leader
- ✦ Build dedicated, cross-functional teams
- ✦ Teaching people to design experiments (hypothesis, next action, risk containment)
- ✦ Coaching teams to execute 'the startup way'
- ✦ Engage 1 executive sponsor to clear away bureaucratic obstacles; work by exception
- ✦ Create new ways to measure success (metric system)

These initial pilots can serve as a starting point to develop an internal 'framework' or 'incubator' for innovation teams.

Once we have such a framework in place, it can be used to scale across the organization.

We run batches of innovation teams through our incubator, all spinning out improvements or new services.

- ★ Implement a widespread rollout, supported by enabling leaders
- ★ Identify the challenges faced by pilot teams
- ★ Architectural leaders start designing the deeper systems
- ★ Identify and make use of executive-level champions
- ★ Train representatives of all internal functions
- ★ Establish an in-house coaching program
- ★ Set up the mechanisms of metered funding and growth boards
- ★ Design systems to manage startups as a portfolio of investments



This is also the phase in which we develop strong governance. The innovation teams need to be held accountable for outcomes. We need the right set of metrics at the team, program and enterprise level to monitor what works and how it's contributing to the larger strategy of the organization. We also need an [innovative way to fund the innovation efforts](#) (as described elsewhere in this site, this is about moving from 50 page business plans and 'entitlement funding' to a stage gate approach where teams get funded based on their achievement, called 'metered funding').

The investment decisions are made by a committee or a board; senior leaders who monitor the team and its outcomes and decide whether they will go to the next 'funding round'.

If an enterprise becomes able to replicate the success of these innovation teams, it can start to transform the whole organization towards entrepreneurship.

LAP 5: ENTERPRISE TRANSFORMATION

Building blocks of the corporate startup engine:



Finance: Budgets, funding and risk



Governance: Metrics and milestones (to make investment decisions). How to select and form the right teams. searching versus executing. 3 horizons.



HR, mindset, culture, professional development and coaching for the entrepreneurial teams; incentives and career



Ecosystem: networking, matchmaking, learning



Leadership: who leads this and how? Growth board; committee. 3 types of entrepreneurial leaders



Some of the 5 building blocks have already been outlined in previous laps. The key in this lap is to architect all the element in a balanced effort. And to keep re-inventing them, for an enterprise transformation should be eternal. In the past, innovation could be done by a lap, isolated from the rest of the organization.

Today, we want everybody at all levels to continuously look outside at threats, opportunities and new product ideas. And we want them to look inside for improvements (radical or incremental).

The enterprise continuously reinvents the way it funds innovation and how people are engaged in innovation teams. It should stimulate entrepreneurial leadership at all 3 levels continuously.

What's Next?

Thanks for reading through our entire book on starting agile! Unfortunately, this is just your starting point. Agile grows on you as you 'get into it'.

There are many habits that need to be changed on the road to agility. Examples of habits that need to change are giving candid feedback to each other, using 360 degrees feedback.

Another is decision making; although simple in theory, this is a long-lasting process that requires change from both leaders and people 'on the floor'. Setting common goals, using for example OKR is another big change we see in organizations.

Fortunately, you've come to the right place: a group of seasoned agile entrepreneurs and coaches ready to get you on the right track.

We are on a journey to touch 1.000.000 people with the agile spark. The spark has three components: entrepreneurship, self-organization and a balanced workplace. We hope this book has sparked you and we can go on this journey together.

Now that you have read Start Agile, there are few things that you can do:

- **Send me your comments, insights, or questions**

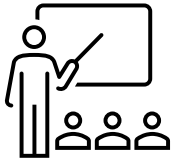
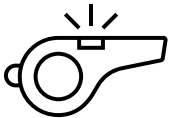
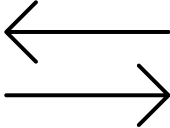
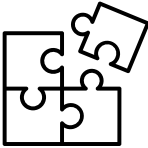
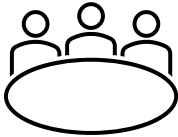

I would be very happy to hear from you. You can follow my social media:

Linkedin: [Hugo Messer](#)

Email : info@ekipa.co

- **Start Your Journey in Agile With Ekipa**

Agile is a long journey. If you intend to start your journey in agile, but you don't know where to start. Ekipa is ready to be your best partner. We provide service and product that helps you being agile.

 Agile Training	 Agile Coaching	 Agile Transformation
 Digital Innovation Strategy	 Digital Product Team	 Agile Certification

 <u>Scrum Master Certification (Bahasa)</u>	 <u>Scrum Master Course (English)</u>	 <u>Product Owner Certification</u>
---	---	---

- **Join Our Exclusive Community**

Learning alone sometimes drain you. So, you need a community. Ekipa provide exclusive community with interesting benefit. Here are our exclusive community:

[Agile Circles Indonesia](#)

Agile Circles Indonesia is one of the largest agile communities in Indonesia. We create a space for our members so that we can discuss, share knowledge & experiences, and meet up so we can grow together.

EXPLORE AND GET YOUR QUALITY COURSE NOW!

AGILE FUNDAMENTAL

BEGINNER

SPECIAL OFFER

**60%
OFF**

~~1,25 JT~~

500 K

**FOR 25 FIRST!
PURCHASES**



What will you learn:

- Understanding of what it means to be 'Agile'
- Understanding iterative vs continuous workflow
- Practical experience in applying Agile Methods via stimulations & workshopping client scenarios
- Benefits of implementing Agile frameworks & practices
- Knowledge of Kanban, Scrum, Lean, Principles
- Action plan to implement Agile frameworks & practices

What will you get:

- 24 Amazing Lectures
- Access on tablet and phone
- Certificate of completion
- 1:55:06 On-demand video

Student Feedback:



Rating scale: 5.0
"Great to start Agile journey"

CLICK HERE TO GET IT NOW