改善

Ángel Medinilla

Agile Kaizen

Managing Continuous Improvement
Far Beyond Retrospectives

Springer

Agile Kaizen

Ángel Medinilla

Agile Kaizen

Managing Continuous Improvement
Far Beyond Retrospectives

Springer

Ángel Medinilla
Proyectalis
Mairena del Aljarafe
Seville
Spain

ISBN 978-3-642-54990-8 ISBN 978-3-642-54991-5 (eBook)
DOI 10.1007/978-3-642-54991-5
Springer Heidelberg New York Dordrecht London

Library of Congress Control Number: 2014941731

Printed on acid-free paper

Springer is part of Springer Science+Business Media (www.springer.com)

Life is growth. If we stop growing, technically and spiritually, we are as good as dead.
– O Sensei Morihei Ueshiba

You don't have to be the Dalai Lama to tell people that life's about change.
– John Cleese

Here's to the crazy ones. The misfits. The rebels. The troublemakers. The round pegs in the square holes. The ones who see things differently. They're not fond of rules. And they have no respect for the status quo. You can quote them, disagree with them, glorify or vilify them. About the only thing you can't do is ignore them. Because they change things. They push the human race forward. And while some may see them as the crazy ones, we see genius. Because the people who are crazy enough to think they can change the world, are the ones who do.
– Steve Jobs

Preface

This book is based on my experience while working with companies as an external trainer and consultant. I have helped all kinds of companies, from 12 to 10,000 employees, to successfully implement Agile frameworks. Additionally, I have trained several thousand managers and developers on topics like Scrum, Kanban, Lean, Agile, Agile management, team coaching, Lean Startup, Agile product management, and change management. Client profiles include companies in the following industries: telecommunications, banking, videogames, software factories, mobile application development, government, logistics, retail, dot-coms, online services, start-ups, and media companies.

My previous book, *Agile Management*, received very good comments and appraisals. I'm happy about that, but many of the comments mentioned that it was a good book "on how to manage *software* companies." That's probably one of the problems with relying too much on your own background, using personal stories or using some key buzzwords like *Agile*. I'd like to assure you that this book is addressed to **any human organization** that feels the need to improve and obtain better results—no matter what kind of organization, market, product, technology, vision, goal, or size.

Nearly everyone I meet knows the famous Albert Einstein quote that defines insanity as "doing the same thing over and over again and expecting different results." It doesn't matter how much sense this quote makes; I feel that the vast majority of companies are stuck in the same process, methods, tools, practices, and behaviors, yet they expect to obtain higher productivity, bigger market shares, better quality, and shorter Time to Market.

If we want better results, we have to make change happen. We live in times of constant change, and even if we feel fine with the current state of things, we will probably find sooner rather than later that the environment, customers, competitors, technology, employees, or markets have changed and our current state of delight and complacency is no longer sustainable. As Edward Deming said, 'It is not necessary to change. Survival is not mandatory.'

But, again, it was Albert Einstein who said that problems can't be solved with the same mindset we had when we created them. In order to improve our companies, we have to improve ourselves. That's why I believe that the foundation of improvement is not found in processes, practices, techniques, or tools (although I'll provide plenty of them in this book) but in embracing the right mindset—values and principles.

Beyond process improvement—quality, productivity, time, profits, costs, etc., I believe that there is a higher moral implication behind Kaizen. A Kaizen culture, as any culture, starts with a common purpose, a "noble cause". As Dan Pink points out in his famous Ted Talk about motivation, when companies just focus on profit and this profit gets unmoored from a noble purpose, bad things happen.

Mass production and the Consumer Society have created a world of waste. Our economy is based on an endless loop of buy–break–discard–buy a new one. Companies plan for obsolescence and accelerated consumption. A whole bunch of companies have been created around the concept of producing crap—cheap, affordable crap that will break or be out of fashion soon so we can persuade our customers to buy more crap. Crap they did not need to begin with, but by the time they find out that the crap they bought is not making them happy, we will be throwing a new, cooler crap into the market.

The main focus of many companies, on the other hand, is cutting costs. But instead of making their companies more efficient, which is difficult, they move companies to third world countries where labor is cheap, unions are banned, and they are able to contaminate instead of investing money in filters, cleaning devices, and waste disposal or recycling processes.

The result is that we consume far more than what the Earth is able to provide, and we produce more waste than the Earth can process. The Earth's population is predicted to double over the twenty-first century. Urban areas expand and there's less land available for farming. We are contaminating the water we drink and the air we breathe. According to scientists, we are experiencing the sixth mass extinction, and there's undeniable evidence that we are causing it. Despite the effort of many to discard the proofs, there's global warming, and right now we don't know if we will be able to reverse it.

For heaven's sake—there's even a Garbage Patch in the Pacific Ocean, a gyre of marine debris made of plastics, chemical sludge, and other garbage. It is not visible from space, as plastics are suspended underwater, but the current estimates are that it is *twice the size of the United States*. And there's a similar one in the Atlantic!

We are basically destroying the future for our children, and we just hope someone will do something in the future—since 'it's just the way things are'. A Kaizen mind wouldn't allow such behavior, and the more people embrace the Kaizen culture, the closer we will be to a really sustainable society.

As a part of the Kaizen Army, you are now enlisted to fight for a new production paradigm based on efficiency, collaboration, respect, sustainable processes, built-in quality, and waste removal. As you will see, the expect results go far beyond increased production, more profits, or faster times to market, but you should expect those also.

There is also a personal, important goal of improving and becoming the best person you can be—learning to see your faults and areas of improvement and being able to engage in this without remorse, guilt, or frustration. And of course, the need to create better, more humane companies remains. Companies that instead of just seeking profit at any cost, let us strive, shine, and explore all our potential.

Seville, Spain
October 2013

Ángel Medinilla

A Note on Drawings

I personally did all the drawings in this book in a one shot, no further editing approach. The most complicated took me no more than 10 min. I know the results are not especially awesome, nor what you would expect from a professional illustrator (which I'm not). But I wanted to make a point out of it: everyone can draw.

More and more people are interested these days in how to make their work more visual. Books are published on how to draw business plans, *sketchnote* meetings, or introduce visual facilitation tools in the work environment. We are basically relearning to draw in order to make a more engaging experience out of our dull note-taking processes and to help better process information.

All my drawings were done on an iPad^{tm} using the NoteShelf[1] App and a regular stylus (no fancy stuff here). They were exported as images and uploaded to Tumblr,[2] from where I copied them and pasted into the document. You can search for them and others of my sketches at http://learningtosketch.tumblr.com/

You will find more information on visual facilitation and sketchnoting in the Resources section at the end of this book.

[1] http://www.fluidtouch.biz/noteshelf/

[2] http://www.tumblr.com/

Stay in Touch!

There are several ways you can stay in touch after you've read this book. Most of them are listed in my contact page, http://www.proyectalis.com/en/AngelMedinilla; everything from my e-mail to my LinkedIn profile, Twitter account, blogs, slides, videos, and more is listed there.

I would especially suggest that you join my monthly(ish) newsletter, *Agile Angel*, where I update information on the conferences I'll be attending, new videos and materials available, or the training courses I'll be delivering in the next few months. I also try to give my best Agile advice as well as a lots of information on books, articles, events, courses, and all Agile—in a fun and positive style. Plus, you can unsubscribe whenever you want, and we are 100 % spam free.

Acknowledgments

Of course, the first acknowledgment of this book must be for Taichii Ohno, Shigeo Shingo, Masaaki Imai, and all the fathers of the Kaizen concept. We truly walk on the shoulder of giants. *Domo Arigato Gozaimashita*. Thanks also to James Womack, Daniel T. Jones, Daniel Roos, Jeffrey Liker, Mike Rother, and all the other Lean authors who helped to bring the Kaizen mindset to the world.

Tribute must be paid to the two High Ladies of Agile Retrospectives, Esther Derby, and Diana Larsen. They cleared the path for countless Agile teams looking for learning and improvement. We can never be thankful enough.

I want to thank all the people that assisted to the Agile Kaizen sessions delivered at several conferences, including Scrum Gathering Las Vegas, XP2013, ALE2013. The feedback received helped in the process of writing this book—we also had some good times.

Once again, this book would not have been possible without the faith and encouragement from Ralf Gerstner and the Springer crew. My first book, *Agile Management*, was easy to write for me, but this one was a different kind of beast, and their patience made it possible.

As always, for everything I achieve in life, I must thank my wife and son for providing a safe place to return after every travel, a motherland for which to yearn, and all the love a heart can host without exploding. But I would like my wife to let me get a 'Kaizen' tattoo and my kid to stop shooting at me when we play videogames on the same team.

And I also want to thank Gonzo, my old Cocker Spaniel, for interrupting the typing every once in a while to demand a good ear scratching. Happiness is truly in small things.

Contents

Part I

Agile Kaizen

1 Understanding Kaizen

A Brief Introduction to Continuous Improvement Cultures

Friday morning—Michael stopped his car at the company's parking lot and turned off the engine. Then he waited for a full minute before going out. He wasn't in the mood for what he had to do that day.

A couple of years before, Michael's company had adopted Scrum, a framework for rapid product development. Scrum consisted in assembling cross-functional customer-focused teams that were fully empowered to develop a product from concept to cash. It worked in small iterations of 2 weeks, at the end of which they had to deliver some increment of the product to the customers in order to obtain some feedback. After each iteration or 'Sprint', the team had a short meeting to review the process in search of impediments. Then, they were expected to propose improvements to the process. This was called a retrospective. And today was retrospective day.

Michael was a 'ScrumMaster' for his team: he was responsible for the Scrum process and was considered to be the 'Impediment Removal' guy. At the beginning, Michael attended some Scrum seminars and read some books, then incorporated what he learned with his team—and the results were good. Everyone was happy with some improvements, like having visual tools (boards with a lot of information on what was happening) or short daily meetings to debrief, synchronize, and communicate. Even some techniques and tools were imported, and everyone was happy about it.

But after some time, Michael started to feel like something was not going right. He felt stuck. Some people started to complain about 'time wasted at retrospectives' and the team even skipped a couple of them when project deadlines were near.

Á. Medinilla, *Agile Kaizen*, DOI 10.1007/978-3-642-54991-5_1,
© Springer-Verlag Berlin Heidelberg 2014

Above all, Michael felt like nothing had really changed that much. They had adopted a bunch of practices, techniques and tools, but the old culture was still alive and strong. Michael had attended a couple of conferences and user groups, and he started to feel that they might be doing 'Cargo-Cult Scrum'—mimicking the practices, tools, and ceremonies of a real Scrum team, but without actually changing anything. They just threw the Scrum liturgy on top of the existing mess.

Michael had decided to change something at last. He played with the idea of a big improvement event—hiring some consultants, putting everyone together, playing some games, and then obtaining some buy-in. That would definitively help to kick-start change. But he feared that, once the event spirit was gone, everything would revert to the natural state of the company.

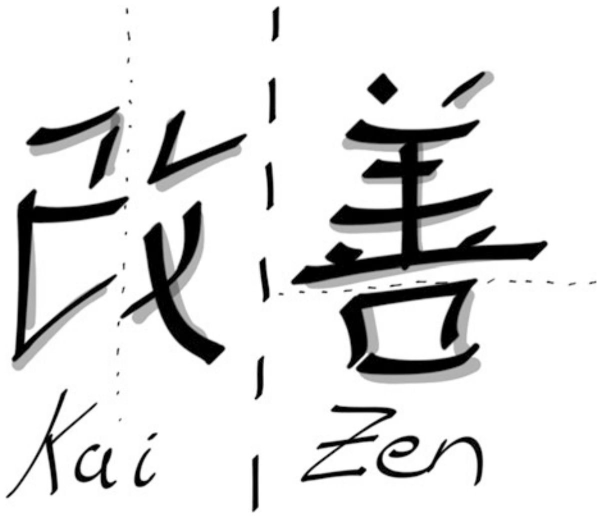
He wanted to grab the bull by the horns and start changing things. But of course, if he just did it on his own, he was afraid that the change would not last, or would last only as long as he was able to push against the cultural resistance. He knew that, in order to make a sustainable change, his role was not to implement the change himself, but to make others conscious of the importance of such a change.

Michael had decided to go for a real Kaizen culture. And it started with empowering the team and making them own the process.

Yes, today was going to be an interesting retrospective day.

What's Kaizen?

The Japanese word 'Kaizen' can be split in two parts: 'Kai', which can be translated as good, continuous, and 'Zen', which can mean wisdom, change, improvement. Thus, 'Kaizen' can be translated as 'good change', 'change for good', or 'continuous improvement'. It is written using two *Kanji* or ideograms, Kai and Zen):



These *Kanji* can also be split: Kai is composed of two drawings. The one on the left represents a man, 'self', and the right part represents the man whipping himself on the back; so Kai (change) uses the idea of a man whipping himself.

On the other side, the Zen *Kanji* has a bottom part that represents an altar, and an upper part that represents a lamb, which means that Zen (good, continuous) uses the idea of sacrifice.[1]

Thus, we can say that in Japanese culture, it is believed that improvement is obtained through constant self-whipping and sacrifice in search of a better state. I've trained martial arts for many years, and this brings me good memories of all the beating and hurting that I've suffered in order to learn and improve.

In fact, I consider that true Kaizen comes from a background much broader than process re-engineering or removing impediments, two very usual explanations from the Lean and Agile worlds.[2] It is true that Kaizen could then be easily translated as change for good or maybe continuous improvement, but the real meaning of the concept is rather complex. Kaizen is used to describe a state of being constantly uncomfortable with the way

[1] Source: http://www.leanblog.org/2012/10/guest-post-kaizen-and-passion/

[2] I discussed Lean and Agile in my previous book, *Agile Management: Leadership in an Agile Environment*. For the sake of this book, it is not necessary to go into detail on the meanings and implications of Lean in modern industries or how it relates to Agile genealogy.

things are right now. It means that we strive for an ideal state of perfection, even though we know that such a state is not achievable: walking the learning path is the real goal and the intrinsic reward. It implies that today we should be doing things better than yesterday, but tomorrow we ought to be doing things better than we do them today.

In many ways, Kaizen is rooted in Japanese culture. People devote most of their lives to perfect swordsmanship (*Kendo*), archery (*Kyudo*), calligraphy (*shodo*), or even the subtle art of serving tea (*chado*), arranging flowers (*ikebana*), folding papers (*origami*), or folding people into impossible forms (*Aikido,* my personal practice). From this perspective, and since Kaizen is the cultural root of Lean, it is no surprise that Lean was born in Japan.

> *They are an intriguing people. From the moment they wake they devote themselves to the perfection of whatever they pursue. I have never seen such discipline. I am surprised to learn that the word Samurai means, 'to serve'.*
>
> – Algren, *The Last Samurai*, Warner Bros. (2003)

If you stop and look at some of these life-long practices in search of perfection, especially when it comes to martial arts, you will find that many of their names end with 'do'. 'Do', for Japanese people, means 'the path' or 'the way'. Thus, 'Aiki-do' means 'The Way of Aiki' or 'The Way of Merging Energies'. The same applies with Sho-do, Cha-do, Karate-do, Ju-do, Ken-do, and so on. This idea of The Way or The Path is crucial in order to understand the Japanese mindset and the principles behind Kaizen. Kaizen is a path; there is no goal, end state, or target. There is no place where we can stop and say 'see, we are perfect now, we can stop improving'. Improvement in a Kaizen culture, again, is like breathing: you never stop doing it. And that's good.

In fact, the object of the practice becomes far less important than the practice itself. In a Tea Ceremony (*chado* or 'The Way of Tea'), the *actual* tea is not important. The important thing is that the practice of perfection fosters harmony, disciplines the mind, and helps the practitioner to seek the purity of enlightenment.

> *The perfect blossom is a rare thing. You could spend your life looking for one, and it would not be a wasted life.'*
>
> – Katsumoto, *The Last Samurai*, Warner Bros. (2003)

For western mindsets, this can be somehow frustrating at the beginning. If there was a western manager around a tea ceremony, he would probably end up shouting to the Tea Master 'Hey, what happens with that tea? I needed it YESTERDAY!!'.

Many western people have mindset conflicts when we start some Kaizen transformation at their company and I explain that there is no 'end state' or 'deadline'. Over time, they admit how frustrated they were at first adopting this 'perpetual uncomfortability' state of mind. But once they saw the parallels with training martial arts, running, or any other life-improving activity, they seemed to relax and better understand the importance of the path.

Kaizen at the Company: The Toyota Way

When Japanese people started using this mindset at their companies, they basically changed the world. Over the span of 30-plus years they forced all western companies to adopt their production methods or disappear. Kaizen-improved companies were four times more productive and obtained quality that was 12-fold better than that of their western counterparts!

This happened because, after World War II, there was no way that a devastated and resource-scarce Japan could compete with western companies unless they dramatically changed the rules of the game. Many Japanese companies, Toyota being the most well-known example, engaged in an endless path of identifying and eliminating waste in their production lines in order to maximize the value delivered to the customer per unit of effort. In order to do that, instead of hiring smart consultants or asking their managers for solutions, they moved the heart of the Kaizen transformation to the *Gemba*, that is, the place where the actual work was being performed.

Teams were formed and put in charge of their production cells. They were not only asked for performance, they were also asked and made responsible for overall improvement. Management was redefined as a way to support the teams, not to give orders and think for them.

Soon, several tools and techniques were defined in order to support this improvement process. Work was standardized, and everyone was responsible for following and honoring the standard, which was visible for all at the production cell. In addition, these standards were changed several times a year in order to introduce improvements.[3]

Every piece would be inspected against the desired quality standards, and there would be no passing on of poor quality or defects. *Andon* devices were designed to let the team stop the line when defects or major problems were spotted. There was a huge cost every time the production line was stopped, something that happened constantly at the beginning, but over time these traumatic events started to be more and more rare as the quality of the process improved. Over time, the benefits that came from perfection by far covered the initial investment of the stop-the-line policy.

At any stop-the-line event, cross-functional task forces supported by managers would analyze the root causes of problems (not just the surface) in order to decide how to not only fix it on the short term, but to provide long-term definitive solutions so they should never need to worry about the same problem twice. The knowledge obtained from such an initiative would then be spread to other production cells in the company.

Kaizen did not happen just in the presence of obvious defects or impediments—regular Kaizen events were scheduled in order to analyze the workflow and spot improvement opportunities. In order to support this, a Visual Management policy was enforced—making all the data and relevant information of the production line visible by everyone, always. At any production cell, Kanban boards would show the state of inventories, production, lead times, cycle times, flow of unfinished goods and materials, and so on.

Teams would own their production environment, taking care of their tools, sorting everything, standardizing their work, streamlining activities, and enforcing team discipline through peer review. They would also care about safety and security procedures, and would be held accountable for the improvement of all of them.

They would analyze their production cells and spot sources of 'waste'—anything that did not add value to the product and was susceptible to removal or reduction. Waste included unnecessary transportation, inventory excess, overproduction, idle times, waiting or queuing times, handouts, over

[3] I consider that most western companies did not understand this concept, and standards are defined, certified, and then never updated because the cost of doing it is so high. Toyota standards were basically sheets of paper hanging on the production cell, and changing them was essentially cost-free and instantaneous.

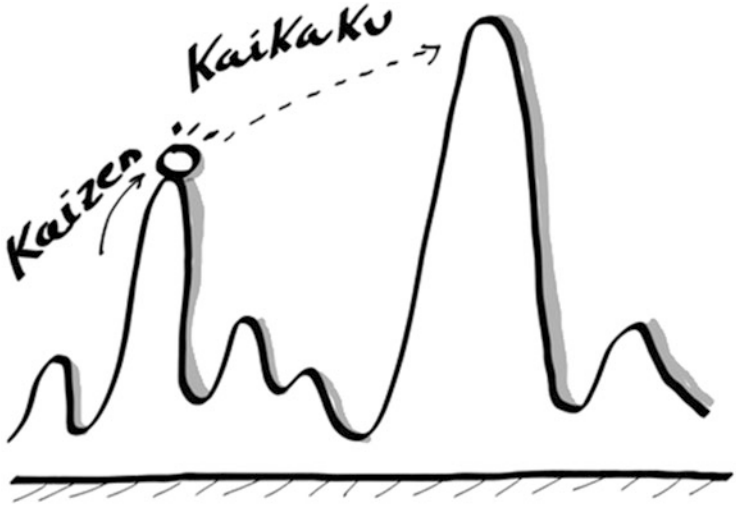
burden, and bureaucracy. Not a single time was this waste labeled as ‘impossible to remove’, ‘the way we do things around here’, or a ‘necessary evil’—such attitudes would probably have resulted in intense coaching by the Kaizen Sensei.

Regular analysis of the value stream (the sequence of activities from concept to cash) would result in the identification of bottlenecks, which would then be the subject of optimization in search of better flow of work batches. The value stream would hence be optimized for shorter lead times, faster times to market, higher capacity, and lower production cost.

Kaizen and Kaikaku

Although Kaizen is the first idea that comes to mind when talking about improvement in Toyota-style production systems (usually referred to as Lean Production), sometimes the situation calls for a big, hard, one-shot change instead of a more regular, soft, and continuous process of improvement. We refer to this kind of traumatic change as *Kaikaku*—Radical Change during a limited time.

Radical change might be needed when we reach a ‘local maximum’—if we have improved up to the point that we are not perfect, but maybe optimal enough, and there’s no efficient way we can improve our current situation unless we change big foundations. A new approach, technology, or process might be needed in order to improve from our current state.



In fact, many Lean experts prefer the Kaikaku approach when it comes to Lean transformation. Their argument is that Lean is so difficult to learn that only in a life-or-death crisis situation will people take the effort, investment, and risks to do it.

One form of Kaikaku are so-called Kaizen events—one-time workshops where we gather a lot of people and ask them to analyze a problem and design a solution or improvement plan.

Kaizen events might be a good practice when we face some long-lasting problems that need an extra boost of energy in order to be engaged. But the problem with Kaikaku and the one-shot events is that, usually, once the energy created during the event is gone, the culture remains and drags any initiative toward the initial state. In order to obtain the most out of Kaikaku or Kaizen events, a continuous follow-up plan must be designed before the change and observed with strong discipline after it, thus making the improvement more continuous over time.

Another problem with the Kaikaku approach is its traumatic nature. Change is always hard, but radical change might be too hard for many cultures. No matter how seductive the 'one-shot fast change' idea might be, you should remember that any change is always unconsciously perceived by the human mind as an aggression, and a big aggression might not be well understood even for the most rational mind.

According to my own experience, cultural change takes time and sustained momentum. So my advice would be consider Kaikaku another tool for building your transformation, but not to place all your hopes and chances on it if you really aim to create a Kaizen culture.

Kaizen Culture

Culture is a complex term, and I tend to define it the easiest possible way—'culture is the way we do things around here'. A corporate culture can be defined, we learned, by a common purpose (the noble cause), shared values (defined by what we do and do not do, and not by a fancy sign in the lobby) and some shared artifacts (both physical and behavioral). We also learned that the best way to assess and define a given culture is through storytelling: stories about the company define what we are and what we are not.

Culture can be an enabler of a Kaizen transformation if we have established a powerful purpose and fostered the right values of quality, improvement, and excellence.

At the end of World War II, Matsushita stood up before a gathering of his dejected, demoralized workforce, in an occupied country, with all the company's inventory taken by the occupying power, and said 'I've been thinking about purpose.'

He then painted a word picture that spoke to everyone, about how taking the lead in quality and innovation and low prices would force competitors to do the same and 'in 250 years would eliminate poverty in Japan.'

He sat down to silence. Then, one by one his employees stood up, some with tears in their eyes, and said 'I think I could dedicate my life to this.'

Much of the 'Japanese way' that conquered the world's economy in the 1980s can be traced back to that moment.

– Phil Dourado, *The Leadership Hub* blog[4]

But most times culture is more of an impediment than an enabler for Kaizen. The reason is that a successful Kaizen transformation challenges the status quo and demands change, and this usually conflicts with existing culture—'this is not the way we do things around here'.

There are several enablers for a Kaizen culture. You should frequently conduct assessments all of them in your environment and have open conversations around them. The main enablers I usually check for are:

- **Purpose**: if you are going to ask your people to excel and improve, to put their energy into making something awesome, you need to show them a noble cause, a global purpose beyond profits, company growth, and stakeholder wealth—who would really care about those?
- **Long-term vision**: every Lean, Agile, or Kaizen transformation means a huge investment of energy, work, time, and resources. Sometimes we will need to change things that seem to be working in a moderate way in order to make an improvement, and that change will cost money and time. People must be conscious of the effects of investment over time and the

[4] http://www.theleadershiphub.com/blogs/how-inspire-people-tough-times-kotter-matsushita

expected better state. Learning must be a real priority, especially over short-term results.

- **Whole system approach**: Show them the whole picture and avoid the temptation of suboptimization—trying to make the system better by working only at a local scale. Systems Theory and common sense show that, in order to make the whole system better, some parts of the system must operate at loss. Just focusing on 'doing your stuff' instead of caring about everyone will eventually break your system. Lean, Kanban, Theory of Constraints, and other frameworks can help your people embrace this concept.
- **Constant communication**: constant communication of course means communication in all ways, not just from managers to employees. Communication is part of our work, not just additional work. People interacting with each other are not to be seen as wasting time. If meetings are not efficient, the answer is not to kill the meetings, but instead to teach people how to be more efficient and productive in their communication.
- **Quality first**: in more industrial environments, the paradigm that 'quality always pays off' is well understood, but when it comes to knowledge workers it still amazes me how many talented people are asking their teams not to document, not to test or to just drop quality in order to meet deadlines. They are thus creating technical debt that will cost more in the future than the cost of building quality into the product up front.
- **Courage and the absence of fear**: in order to create a Kaizen environment, we must foster a culture of no fear, but even in the presence of natural fears (fear of risk, fear of uncertainty, fear of blame) people shall be able to overcome those fears and point out what has to be done. Even if it's not in their own personal responsibility area, everyone should be able to call other people's attention to what they consider to be an impediment, a defect, or an improvement opportunity.
- **Transparency**: people should be able to question everything. There is no way to look at the whole system and point to others' impediments or improvement opportunities if there is no transparency. It's not by chance that Visual Management is at the core of Lean thinking. In order to enhance transparency, every trace of a 'blame game' culture must be eradicated. When there is no transparency, most of the time it is because people fear that they will be scorned or blamed for what they show. In that case, you will never be able to engage into those problem areas in order to improve them. By the way: Visual Management means that

information is constantly visible and easily manageable—electronic tools are not quite as good for that.[5]

- **Empowerment and ownership**: these two words have been in the consultancy jargon for decades, but there is a reason for that—if you have seen the effects of real empowerment and true ownership, the feeling that improving the system is everybody's job and you have the authority to 'stop the line' and call for changes, you know how powerful those concepts can become. Of course, hierarchies, perceived power, and 'command and control' kill the sense of empowerment. Ownership also means responsibility and accountability. These must usually be grown, as people are not used to working in environments where they don't have to just care about what the manager says and instead are expected to propose ideas, to take risks, and to be responsible for their actions. Managers sometimes are afraid of empowerment, as they fear their role in the company might be undermined, but a new model of co-management, and a new role of the manager as a Team Gardener/Team Supporter/System Warden must be developed in order to foster Kaizen cultures.[6]
- **Teamwork and self-organization**: empowered individuals should actively seek to collaborate with each other. Discipline and team standards must be observed, and boundaries, constraints, guidelines, and goals must be provided by the organization—usually through the managers. But the team must be able to self-organize in search of those goals. Complex environments can't be managed optimally with simple approaches or central intelligence—they call for self-organization as a way to deal with complexity. In order to achieve real teamwork and real collaboration, people must learn how to discuss, engage conflicts, listen actively, reach consensus, respect others' opinions, and communicate in a nonviolent way.
- **Recognition**: a lot of literature has pointed out in the last decade that money rewards and bonuses do not work in creative knowledge-based environments. But we must never forget the need to use constructive feedback and, especially, to give recognition for individual and team contributions to company improvement. The father of behavioral psychology, B.F. Skinner, found that positive reinforcement is much more

[5] 'Yes, but I have distributed teams and blah blah blah…'. If you really must use distributed information, then use Visual Management devices AND replicate that information in the electronic tool. The Physical Team Board is irreplaceable!

[6] More information on the manager's role in such an environment can be found on my previous book: Medinilla A (2012) *Agile Management*. Springer.

powerful than negative reinforcement or punishment, yet I sometimes feel like the most common way of treating employees worldwide is through shouting, yelling, scorning, controlling, and punishing.

The old school made the amazing mistake of supposing that by removing a situation a person likes or setting up one he doesn't like – in other words by punishing him – it was possible to reduce the probability that he would behave in a given way again. That simply doesn't hold. It has been established beyond question . . . We are gradually discovering – at an untold cost in human suffering – that in the long run punishment doesn't reduce the probability that an act will occur. [. . .] Now that we know how positive reinforcement works and why negative doesn't, we can be more deliberate in our cultural design. We can achieve the sort of control under which the controlled, though they are following a code much more scrupulously than was ever the case under the old system, nevertheless feel free. They are doing what they want to do, not what they are forced to do. That's the source of the tremendous power of positive reinforcement – there's no restraint and no revolt. By a careful cultural design, we control not the final behavior, but the inclination to behave – the motives, the desires, the wishes.

– B.F. Skinner, *Beyond Freedom and Dignity*

Overall, the most important enabler for a Kaizen transformation is the organization making it a real priority—a 'national emergency'. Calling something a priority does not turn it into a real priority. The Kaizen transformation must be more important than short-term deliveries, quarterly financial goals, existing structures, company politics, personal agendas, or pre-existing commitments. Any attempt to improve that gets blocked by a 'not now, there's something more important to do' automatically kills the Kaizen spirit that we are trying to put into people's hearts.

Kaizen Roles

Let's be clear: there should not be any kind of formal roles in a Kaizen transformation. The moment you print business cards with the 'Kaizen Manager' title on them, everyone will get the message that 'Kaizen is what the *Kaizen Dude* does'. What we are trying to do is create and foster a culture of collaboration and distributed responsibility. Roles, at least formal roles, work against this fundamental idea of a Kaizen culture.

On the other hand, it is a human condition that, when something is everybody's responsibility, nobody feels personally responsible. It is believed that economist Adam Smith, who linked economy and psychology when studying the dynamics of microeconomics, was the first to describe the 'tragedy of the commons': if everybody just works in his own interest, the collective will deplete any shared resource even if that's against the group's—and hence the individuals'—long-term interest. You can see examples of this sad economic behavior everywhere, from excessive fishing to the systematic destruction of habitats and forests.

This is just another case of suboptimization: if every part of the system just looks to their own interest, the result for the whole is worse than optimal. The same can be applied to positive efforts and dedication: if everyone expects the rest to be the ones improving, then nobody does so.

In order to enforce collective ownership of the improvement process, ride the change, and implement the roots of Kaizen culture, there are several informal roles you can use:

– **Kaizen evangelists**: the innovators. These are the people who learn about Kaizen—by reading this book, for example—and who bring new ideas to the company. They will constantly remind people of the importance of Kaizen for the company future, and will train or support people on their learning of the Kaizen mechanics. Evangelists will make information available to everyone and deal with skeptics when it comes to questions about the Kaizen culture.
– **Kaizen agents**: the early adopters and real change agents. These are the people actually responsible for getting things running. They will facilitate the Kaizen events, maintain momentum, support teams in their improvement efforts, and keep energy in the improvement plans.
– **Kaizen champions**: these are Kaizen agents up in the corporate ladder. A tipping point in the Kaizen transformation comes when you find someone in the upper-management layer who is empowered enough and committed to the Kaizen transformation. They will make resources available, evangelize the upper managerial levels, help to remove corporate impediments, care about cultural change, and provide cross-departmental collaboration and whole-system improvement goals.
– **Kaizen leaders**: these are the role models. Leaders are everywhere—it is not a management-appointed position. Kaizen leaders will lead by action, and will show others how to improve by actually improving and being

uncompromising about moving out of the comfort zone. Be careful to understand ‘Leader’ as ‘Role Model’ and not as ‘boss’, ‘manager’ or any form of ‘I will tell you what to do’. The old paradigm of managers giving orders and workers performing them kills a Kaizen culture!

Other than the informal Kaizen roles, it is also important that you care about creating **Kaizen communities**. There are two kind of Kaizen communities: functional communities—like, for example, when all the engineers meet in order to improve engineering practices, or all the project managers meet in order to improve project management; and cross-functional communities—groups of people from different departments who care about whole-system improvement. From those, the Kaizen agents and evangelists’ functional community is especially important in order to keep and maintain the Kaizen spark going forward.

Above all, keep in mind that Kaizen agents are not the ones responsible for improvement, but are the ones accountable if there is no Kaizen culture of collective ownership and empowerment to improve.

Kaizen and Agile

Continuous improvement is at the core of Agile—the Agile Manifesto. It talks about change—both responding to it instead of blindly following the existing plan and embracing changing requirements. It talks about empowerment and motivation—creating self-organizing teams and trusting them to get the job done. It talks about continuous attention to excellence and about removing waste—maximizing the amount of work not done. And of course, it talks about meeting at regular intervals and reflecting (in Japanese *Hansei*) on how to become more effective (improve, *Kaizen*), and then tuning and adjusting behaviors accordingly.

This idea of frequent and continuous improvement is, for me, the most Kaizen-like idea behind the Manifesto. Other frameworks mentioned improvement and learning but, for example, they prescribed things as ‘project post mortems’—finding out best practices and lessons learned at the end of the project, *when there is nothing we can do to steer or change it*. On the contrary, Agile frameworks enforce the constant reflection (*Hansei*) and tuning the process *during* the project.

We can also drew a parallel between Agile constant delivery of product increments and Kaizen constant delivery of small, cumulative, evolutionary improvements. In both cases, product increments and small improvements are delivered by a cross-functional, self-organizing team that measures progress as value delivered—meaning that, in Kaizen, it is important that we measure actual improvement beyond the effort, tests or even failures that we might incur during the improvement process.

Two good examples of Kaizen in Agile environments are Retrospectives and Refactoring. The Retrospective is a team activity aimed at identifying team impediments and how to remove them. Impediments can include anything that is preventing the team from being more successful, and the removal of these impediments depends on the team's capability to correctly identify the impediment's root causes and propose correction plans, and to make themselves responsible and accountable for executing those improvement plans. These plans typically include actions to make others conscious of the problem and asking for managerial support.

On the other hand, refactoring is an improvement activity prescribed by eXtreme Programming that consists on rewriting working code in order to make it more efficient, clear, simple, robust, modular, or flexible. Refactoring is considered one way of building quality into the product, and the time spent refactoring is an investment in reducing technical debt.

Unfortunately, there is plenty of literature on how to manage continuous delivery of product functionality, but there is not that much when it comes to managing continuous improvement. It is worth mentioning the famous and brilliant *Agile Retrosp*ectives[7] and maybe some books about refactoring code, like Uncle Bob's *Clean Code*[8] or Martin Fowler's *Refactoring*[9]—which were basically very technical and engineering-oriented books—but then... Nothing?

As you will see in the next chapter, this lack contributes to many teams' failure in their efforts to move beyond Agile processes, practices, tools, artifacts, and roles (which are usually confined to the team level) to a real Agile/Kaizen culture of improvement, quality, excellence, and growth. That's why I felt compelled to write this book.

[7] Larsen D, Derby E (2006) *Agile Retrospectives*. Pragmatic Bookshelf.

[8] Martin RC (2008) *Clean Code*. Prentice Hall.

[9] Fowler M (1999) *Refactoring: Improving the Design of Existing Code*. Addison-Wesley Professional.

Summary

Far from processes or tools, continuous improvement is a cultural core that must be enforced through common purpose and values. Kaizen is not a time-bounded initiative with a goal or a deadline, but a lifetime path of searching, reflecting, learning, and improving. The prevalent mindset of twentieth-century western-style companies can prevent Kaizen through the strong maintenance of the status quo and general human resistance to change. But in order to improve, change is inevitable. In order to unleash Kaizen, several enablers must be observed, including a long-term vision, collaboration and ownership by every person in the company, putting customer value and quality first, taking a whole-system perspective, and fostering courage and transparency.

Although Kaizen takes the form of small, continuous and incremental improvements, sometimes reaching a local maximum or the appearance of a crisis calls for radical change or Kaikaku. A common-sense approach to combining both is the preferred strategy, but when in doubt, Kaizen is less traumatic and gives you more opportunities to steer your improvement project than Kaikaku.

In order to maintain the Kaizen transformation, fostering informal leadership in the form of Kaizen evangelist, agents, champions, and leaders can also be a huge enabler. Functional and cross-functional communities can also be crucial to keeping improvement happening at the personal, local, and systemwide scale.

Things to Try

- Start gathering stories about improvement in your company. Identify big improvements or changes in the past and ask people about them and how they felt before, during, and after those changes. Try to spot common themes in those stories and tie them to your definition of culture. Compare them to the definition of Kaizen/Kaikaku found in this chapter: how many of the Kaizen enablers were actually present? Ask people what made change possible, who helped it and who obstructed it, and what mistakes were made during the change. Overall, try to learn how change takes place in your own environment.
- Use the previously gathered stories to spot company and team values—the things people really care about, not a consultant-baked management-

forged definition of values. Use them to start your corporate culture assessment.

– You can also use those stories to define a desired state. Another way of envisioning the desired state is using some of the games and activities in the second part of this book, or benchmarking your company against market leaders and companies you admire.

– Assess the Kaizen enablers list with your people. Ask them to rate from 0 to 10 how they feel about them—for example, if they feel empowered, if there's ownership, if there is transparency, if they put quality first, and so on. Spot the main gaps and ask people how could they improve their score.

– Ask managers how and where your company could start a stop the-line policy: the closer to the beginning of the actual production line and the more control points you place it near, the better. A negative to implement even an end-of-the-line major quality control should be addressed by all stakeholders as a low level of commitment to quality and customer value.

– Engage in Kaizen conversations—remember the Agile value of 'individuals and interactions over processes and tools'. Find some time to hang out with individuals and teams and ask open questions about improvement: a better company, a better process, a better product. Ask them how are they learning and improving at their craft. How is their process or their environment different now from a year ago? How can we measure improvement, or even know if we are improving at all?

– Start a Gemba Walk routine: wander around the company observing how people work. Ask them questions about what they are doing and the impediments they are facing. If there is some information you cannot see instantly, encourage teams to develop visual tools to make that information available to everyone. Pair with some other managers, leaders, experts, or team members in Gemba Walks in order to have different perspectives about what is happening. Try to spot any reports or bureaucratic processes that you can eliminate through the use of Gemba Walks. Make sure that people understand the nature and goal of the Gemba Walk and do not see it as control, audit, or unnecessary interruption of their working routine.

– Start designing regular Kaizen events. They can take the form of daily meetings, bi-weekly Retrospectives, or monthly all-hands meetings.

2 Why and How Kaizen Fails

Impediments to a Successful Kaizen Culture

As Michael approached the team's retrospective room he mentally reviewed several of the retrospectives he had conducted with the team over the last couple of years. At the beginning, the team was so shy and silent—like a rabbit flashed by truck lights in the middle of the night. They didn't know what to expect from retrospectives, and some of them secretly feared that this was a new management tool to control and audit them, finding all the mistakes they were making and using these mistakes against them at the next performance review.

With patience and trust, Michael was able to break the team's emotional dam—impediments started to be identified, and soon the team was excited with the possibilities of the retrospective. They were also glad to have some secure, private space to speak intimately about the team's environment and even some personal feelings.

Michael was so happy at that time. He was sure that he was playing the Agile game by the book. But several months later he realized that nothing had really changed. The same impediments were listed over and over. Despite the changes made by the team—project boards, sticky notes, daily meetings, new frameworks, and tools—the customer kept complaining about the low quality of the product. Trouble reports were still coming and coming, and there never seemed to be enough time to build quality into the product.

Managers kept acting bossy, making all the real decisions, and maintaining the status quo. Teams tried to maintain the spirit of iterations, but priorities kept changing and new stuff kept coming all the time. They were asked to work on several things at the same time—which everyone

Á. Medinilla, *Agile Kaizen*, DOI 10.1007/978-3-642-54991-5_2,

agreed was a bad idea, because it introduced a great deal of waste and context switching—but of course, when things got blocked, which happened constantly, what could anyone do but start working on something else?

On the other hand, he did not see team dynamics changing that much. Yes, they talked and interacted more, but when it came to collaboration there was little change. Mostly everyone preferred to work on their own, and design decisions were made in order to divide work more comfortably instead of according to product, quality, or customer best interest.

He confronted the team with this uncomfortable truth, and they just complained that it was not their fault if they identified impediments but then the company did nothing about them.

Michael tried to remove some of those impediments on his own, and for some time the team was pleased with that. After some more time, Michael came to the conclusion that his attitude was just making the team even more comfortable with the situation—he acted as an overprotective 'Scrum Mom', caring for her kids and providing everything they needed, and this was not helping them grow and mature.

Michael also tried to engage management in the retrospectives, but every time a manager cared to show up, the retrospective was silent with fear—nobody wanted to speak truth to managers and be blamed. Of course, this backfired badly, as managers complained that retrospectives were a waste of the team's time and that any impediments should be identified and fixed by their ScrumMaster, team leader, or manager.

Michael knew that this kind of behavior would definitely kill the chances of getting true self-organizing empowered teams who were able to kick the company up to the next step.

What was going wrong?

Michael had thought deeply about it...

Failing to Improve

If you ever decide to take the Kaizen path, but then look back and see that you are doing things exactly the same way you were doing them a year ago, don't have any doubts—you are failing.

People seem to believe deep in their brains that, even if we keep doing things the same way, results will improve over time due to practice. But of course, there is a big difference between mere repetition and deliberate practice, the kind of practice where you are constantly trying to find new, better ways of doing things.

On the other hand, just changing things constantly without a clear purpose or some baseline does not guarantee that the changes that are introduced are helping. As with self-organization, change for the sake of change is not necessarily a good thing. Change is a necessary condition in order to improve, but not all changes will make you improve.

Of course, it would be great if we established some improvement-related metrics—productivity, quality, or even team happiness. But never trust metrics too much—use them just as an indicator of what might be happening. The reason not to take metrics too seriously is because metrics are hard—if not impossible—in a knowledge-based environment. How do you measure an architect's productivity? Is it all about how many hours she works? Then, if two architects work the same number of hours, are they equally productive?

Think about the most productive person in your environment—how do you *know* it? Is there something you can actually *count,* or is it instead something you *feel*?

> *Counting sounds easy until we actually attempt it, and then we quickly discover that often we cannot recognize what we ought to count. Numbers are no substitute for clear definitions, and not everything that can be counted counts.*
>
> – William Bruce Cameron, Professor of Sociology

Also be aware: what you measure is what you'll get. If you measure 'working hours', you'll get a bunch of them—and that's not always a good thing, unless you can charge for them no matter what the results are and you don't care about the ethical implications of such a business.

Anyway, we just defined the most important way to know you are failing at Kaizen—nothing changes. You can also use the list of Kaizen enablers in Chap. 1 and score your environment against them: do we feel like we are empowered? Learning? Self-organizing? Can we say that there are no blame games in the organization? What about transparency? The Kaizen enablers list gives you an all-purpose 'desired state' that you can compare yourself against. Of course, over time, it is crucial that you are able to describe your particular, specific desired state and assess the differences with the current or past states.

Don't make the mistake of going for the whole transformation too quickly: one of the most common pieces of advice you'll get from Kaizen experts is 'don't make your Kaizen scope too big'. Even if you have a clear image of where you want to be in the future, it's better to agree that it will take more time than you expect. That way, you are able to focus your short-term efforts on the first step toward the full implementation of the desired state.

Over all, in a life-long Kaizen transformation—and there is no other kind—you should be able to look back in time and tell the difference between how you were performing a year ago and how are you performing now, even with no explicit metrics. Do not fool yourself trying to measure improvement after every iteration, month, or quarter—it builds up, and it's difficult to identify the tiny increments.

Top Ten Reasons Why Kaizen Fails

1. **Absence of a real culture**. Kaizen is seen just as another process, tool, or even fad. There is no action aimed at changing people's behaviors or value system; even worse: actions and behaviors inconsistent with the Kaizen Culture are not argued back. There is no clear desired state and there is no noble cause behind the Kaizen initiative. Existing and prevalent culture and processes will also prevent Kaizen, especially if there is a culture of 'you cannot touch that', 'this is the way we've always done it', or 'this is not my/your duty'.
2. **Politics and blame games.** This is a specific case of cultural conflict: managers are more concerned about 'hiding the garbage' and who is to be blamed for any problem or defect than actually engaging in a constructive debate on how to improve. There is no real management buy-in. Sometimes it even takes the form of passive-aggressive behaviors—managers will swear they are into the Kaizen transformation, but then will undermine it. There is usually no transparency and there is a fear of communicating and making information available to everyone.

3. **General resistance to change.** People just refuse to change because any change—especially a big one, as in a Kaizen initiative—implies moving out of their comfort zone. By default, unless there is a strong reason (the noble cause behind Kaizen) people won't understand the need to run the extra mile or engage in the improvement process.

4. **Lack of momentum**. There is one big-bang improvement event, training course, or communication campaign, but then there is no follow-up. People are told to improve, but then there is not enough communication, no visibility, no energy, no champions, no quick wins, no internal marketing of the initiative, no metrics, no process, no training, and no support.

5. **No sense of ownership/no empowerment**. Not everyone is involved in the Kaizen initiative—maybe just some managers or some management-appointed Kaizen roles, who will be seen as the 'folks responsible for Kaizen'. People will feel that Kaizen is just another burden thrown at them. People won't be able to talk back to managers, make decisions, or, metaphorically speaking, stop the line. When confronted with impediments, the regular answer will be 'it's not our fault' or 'we already told the managers, go talk to them'. Sometimes, a symptom of no real ownership and empowerment can be identified when you see the company creating 'Kaizen circles' or 'Kaizen squads' at management levels instead of enrolling everyone into the Kaizen initiative (so, again, Kaizen ends up being something that the 'Kaizen folks' do).

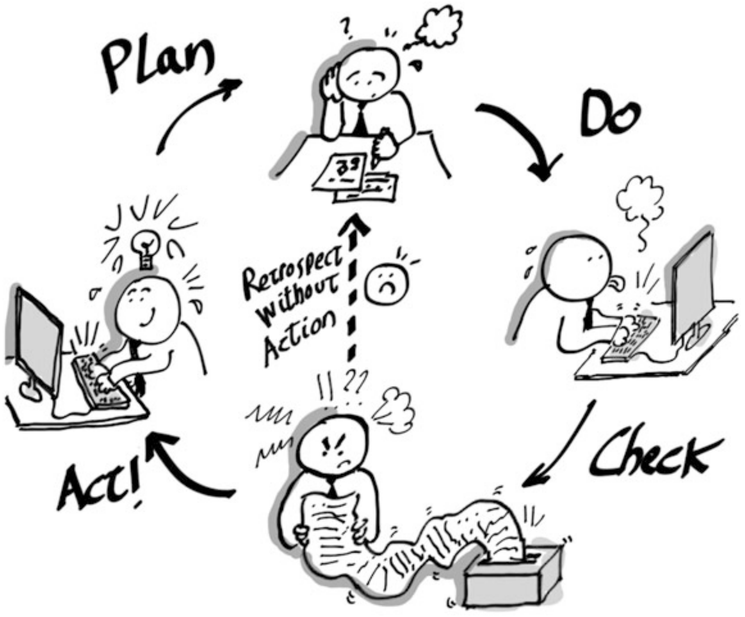
6. **Short-term vision**. Kaizen is not a real priority. The company suffers from short-sighted vision, so financial goals and project delivery dates are considered more important than Kaizen initiatives and their related investment. In other cases, Kaizen is seen as a short-term project with an end date—an example of the 'Silver Bullet' western mentality.

7. **Failure to identify problems.** Japanese Lean Sensei usually asked western workers what problems they were able to spot at the production line, and very frequently the answer was 'We are doing fine here! No problem!'. The Sensei always answered back: 'No problem *is* a problem'. It means that your people have lived so much time with their problems that now they are white noise, like music in the background that you no longer notice. A second stage comes when your people identify big impediments but then are unable to divide them in smaller parts that they can tackle piecemeal. Sometimes, a variant of this problem is that people fail to prioritize their actions, and sometimes

they try for very important—but difficult to solve—problems instead of going for the ‘low hanging fruit’ and creating quick wins to encourage the improvement initiative.

8. **Failure to see root causes.** We just apply shortcuts and workarounds, but the real problem causes are hidden and are never solved. Problems are repetitive—the focus is on short-term solutions instead of investing enough time to understand all the causal relations and all the implied factors. As the original causes are not solved, sometimes the same cause can span different problems in an unpredictable way.

9. **Failure to plan and execute**. A desired state is defined (plan) and then some action is performed (do). Impediments are identified (check), but then nothing is really done in order to remove impediments and redefine the system (*act*!). This kind of ‘failed Deming cycle’ is what you see when Retrospectives end with a list of things we are doing right and things we are doing wrong, but there is no clear plan for action in order to correct those problems.



10. **Lack of resources**. Not necessarily in the sense of physical resources: sometimes there is just not enough time nor skills to approach Kaizen in a productive way. On many occasions teams will be asked to improve, but then there will not be any structured time to analyze, reflect, and

plan for improvement, nor any time or resources to dedicate to the designed improvements.

Change Management 101

In order to change your culture, you have to understand change dynamics. I made a similar statement on my first book and introduced several tools and resources you can use in order to better understand how change takes place. If you want to go deeper into those, I recommend that you read *How to Change the World,*[1] *Fearless Change,*[2] *Switch,*[3] *Crossing the Chasm*[4] and, above all, go watch the *First Follower: Leadership Lessons from a Dancing Guy* video on-line—seriously, this is one of the best lessons on leadership and change you will get for free and in under 3 min!

Anyway, for the sake of this chapter, I give some of the most important reasons any change initiative fails.

Probably, the first reason change fails is rooted in the beginning of your change initiative. Almost every time I have consulted for a company that is trying to change, they started top-down (management message to the company), and the change was addressed to everyone in the company at the same time.

If you are familiar with Moore's 'Crossing the Chasm' chart, which was adapted from Rogers' *Diffusion of Innovations*[5] text, you probably know that any new idea starts with innovators and then is rapidly accepted by the early adopters. But then there is a big, scary chasm that divides them from the majority and the laggards.

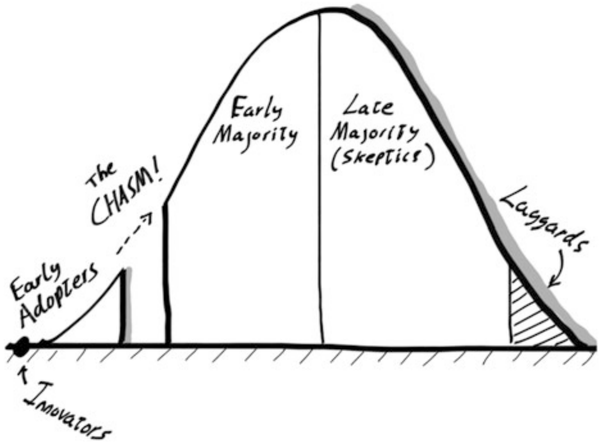
[1] Appelo J (2012) *How to Change the World: Change Management 3.0*. Jojo Ventures BV.

[2] Manns ML, Rising L (2004) *Fearless Change: Patterns for Introducing New Ideas*. Addison-Wesley Educational Publishers, London.

[3] Heat C, Heath D (2010) *Switch: How to Change Things When Change is Hard*. Crown Business, New York.

[4] Moore G (1991) *Crossing the Chasm: Marketing and Selling High-Tech Products to Mainstream Customers*. Harper Business Essentials.

[5] Rogers EM (1962) *Diffusion of Innovations*. The Free Press.



When you try to explain an idea to the whole set of people, the early majority will stay silent, and the late majority will be skeptics. Laggards, on the other hand, will attack any attempt at change, no matter its nature. So basically skeptics and laggards will outnumber the innovators and the early adopters, and the idea will be killed.

In order to maximize your chances of success with the introduction of any idea, you have to incubate it in the safe environment provided by the early adopters. Then, you have to provide small wins and facts to the early majority to make the change attractive and safe for them. You should listen to skeptics, as they will identify resistance arguments—'I don't like this idea because A, B, and C'. As for laggards, you should ignore them. Laggards rarely give rational arguments, and they just resist change because of its nature. But laggards will—reluctantly—follow the majority when it is convinced, or they will move to some other place where they don't have to adopt the change.

The second most important reason why change fails is, as identified in the reasons why Kaizen fails, the inability to plan and execute the change initiative. Change needs constant energy—it's thermodynamics! In absence of energy, the system will always drag to chaos, disorder, and the lowest-effort state. There are several ways to sustain energy in your change initiative, but probably the best ones are based on 'individuals and interactions', i.e., having people acting as change agents/change evangelists.

Being a change agent is hard. It means people will constantly ignore you or even throw stones at you[6]—get used to it! But no matter how much you embrace the idea of being a change agent, never underestimate how it can affect your motivation. It is important that change agents create a community of practice where they can share their pain and get some support.

It is important to understand that change agents are not meant to actually implement change. Their duty, on the other hand, is to make everyone aware of the importance and desirability of the change, to move them, and motivate them. Change agents tell stories and make people want to take part in building a better future.

There are several other reasons your change initiative might be failing. If you study Jurgen Appelo's work in *Change Management 3.0*, you will see that he combined several change models (Plan-Do-Check-Act, Rogers and Moore's chasm diagram, the ADKVAR[7] model, the five I's,[8] and others) and turned them into a deck of questions you can use to evaluate your change plan.[9] Some of my personal favorites are:

– How do you plan to communicate change, or make everyone aware of the initiative? How are you radiating information and easing communication? How are you enforcing dialogue and participation?
– Is there any way you are incentivizing the change and making it desirable? In other words, what's in it for them?
– Is there any ability people need in order to make the change happen? If that is the case, are you training people and letting them practice?
– How are you tying this change to the group's identity? Are you relating change to the team's purpose, values, goals, behaviors, and needs?
– Are you aware of the barriers and impediments to change? What are you doing to remove them?
– Have you addressed the rule-makers in search of early adopters or even change champions?

[6] Usually metaphorically speaking. Usually.

[7] http://www.change-management.com/tutorial-adkar-overview.htm

[8] http://www.newscientist.com/article/mg20327225.700-triumph-of-the-commons-helping-the-world-to-share.html

[9] http://www.slideshare.net/jurgenappelo/how-to-change-the-world-9444890

A commonly overlooked reason for change failure is to not consider how the change is affecting everyone. Our environment and our behaviors determine in great part who we are, the idea of self we forge over time. Thus, the emotional parts of our brains consider any attempt to change our environment or behaviors as an attempt to change our selves—therefore, it is seen as an aggression. In other words, our brains are hard-wired to react aggressively and negatively to change, especially if it comes from the outside.

Sometimes we are so excited about our own ideas for change and improvement that we fail to see how this change affects other people's ideas of security, status, power, influence, comfort, or knowledge. A new process can be a great improvement for the company, but it can also undermine people's confidence on their own skills, and they might react negatively to it. That does not mean they have to be immediately labeled as 'laggards' or 'not team players'; there might be a valid personal reason for change avoidance, even if we fail to see that reason or even if they are not aware of their own personal reasons to resist change.

Whatever anybody says or does, assume positive intent. You will be amazed at how your whole approach to a person or problem becomes very different. When you assume negative intent, you're angry. If you take away that anger and assume positive intent, you will be amazed. Your emotional quotient goes up because you are no longer almost random in your response. You don't get defensive. You don't scream. You are trying to understand and listen because at your basic core you are saying, 'Maybe they are saying something to me that I'm not hearing'. So 'assume positive intent' has been a huge piece of advice for me.

– Indra Nooyi, Pepsico Chairman and CEO

Finally, one of the simpler reasons change fails is because people don't sustain the effort long enough to make change happen. There's a myth about 'overnight success' and it is nothing but that—a myth. If you study some of the most-mentioned examples of overnight success, you will find that there was a lot of work and sustained effort behind it. In my previous book, *Agile Management*, I mentioned several examples and described my personal recipe when it comes to building enough momentum and sustaining the change energy:

– **Courage**: a fearless approach. Change agents will often be afraid of judgment, failure, or loss of status if the change fails. They have to overcome their fear in order to sustain change.
– **Commitment**: devoting enough energy, time, and resources to personally keep the change process going.
– **Discipline**: maintaining the effort over time, never losing focus.
– **Proactivity**: overall, you can't just tell people to change. You have to walk the talk, inspire others, and be the first to represent the change you want.

So. . . What Do I Do Now?

Basically, you must realize and make friends with the idea that you (reading this book) will need to review all these Kaizen problems and change impediments in order to just get things going. As a small briefing of what you will be doing is:

1. You need to convince everyone that Kaizen is, in fact, a culture and not some kind of process, tool, or framework. As we will discuss in next section, you need to gather all the innovators and early adopters and start working on a shared purpose and common values. You need to start gathering stories about transformation, stories about the future, stories that people can tell to each other. Remember that stories are everywhere: customers, suppliers, employees, managers, stakeholders, and community. It's not necessary that these stories talk about your organization at the beginning—you can tell stories about Toyota or any company people admire; the idea is to give them a desired cultural state to get to. You will start giving recognition and positive reinforcement on everything that is aligned with the cultural state that you defined through purpose and value—catch them doing something right! You should also start spreading Kaizen artifacts—A3 Problem Solving charts, Visual Kaizen Boards, and Standards.
2. You will start to communicate the message that we are all on board in the Kaizen process, and will stop any finger-pointing or blame games when talking about problems. You will create cross-functional improvement teams and will watch for fear of management or lack of participation. You will have one-on-one coaching sessions both with people trying to act on their own, managers abusing their power, and people

lagging behind. You will enforce transparency and the use of visual management.

3. You will design and maintain a constant change management project. In order to do that, you will learn about change and constantly review the change management frameworks, updating and adapting them to the current reality of the company culture. You will encourage people to get out of their comfort zones and celebrate learning (through both failure and success). You will constantly remind everyone about the common purpose, the desired state, the noble cause that we are pursuing.

4. You will constantly inject energy into the Kaizen process. You will care that everybody observes the Kaizen events and that there is enough structured and protected time for learning, research, innovation, and personal development. You will create and encourage more Kaizen evangelists, agents, champions, and leaders. You will sustain and nurture Kaizen communities. You will provide people with training, support, literature, and mentorship. You will communicate and celebrate all improvements. You will care to establish the adequate improvement metrics and make them visible and important to all.

5. You will make sure that everyone is included in the Kaizen plan. You will foster progressive delegation of authority on production teams. In order to do so, you will identify key decision areas and define the current level of delegation.[10] You will coach teams in order to make them more mature and, hence, capable of more authority. You will grow responsibility and accountability, first through a culture of trust, transparency, and absence of blame, and then by asking for commitment and follow-up. You will support teams when they advance and grow in responsibility, and make sure they make good use of their empowerment by setting adequate contexts and alignment with corporate goals. You will prevent relapses when a team makes mistakes: you will work with both team and managers to make sure they learn from their mistakes and prevent similar mistakes from happening in the future.

6. You will care about the long-term vision of your Kaizen initiative. You will point out any occasion where short-term goals are going against the Kaizen investment. You will defend the ‘Kaizen budget’ of time and resources devoted to continuous improvement from short-term crises.

[10] A good way of doing so is using Management3.0 Delegation Boards. Learn more at http://www.management30.com/workout/delegation-boards/

7. You will coach teams in identifying problems and impediments. You will teach them how to divide those problems into smaller ones and to prioritize them according to their impact on the team and the ability to cope with them. You will help the team identify the easier impediments in order to have quick, small wins that encourage their Kaizen attitude. You will show teams how to escalate problems when they feel unable to deal with them. You will make sure that the Impediment Backlog is maintained and that it evolves over time.
8. You will help teams to see root causes by introducing techniques and tools. You will constantly communicate the need to get rid of defect sources forever instead of just fixing defects. You will identify attempts to use quick fixes, shortcuts, and workarounds instead of developing long-term definitive and sustainable solutions. You will make sure that knowledge bases are maintained so the impediments do not appear again in the future. You will foster the exchange of knowledge between different teams and the transfer of knowledge to new employees.
9. You will constantly update the Kaizen plan. You will make sure that improvement goals and Desired States are properly defined, and that key metrics are updated and communicated. When the plan goes wrong (and it will), you will check the reasons for the deviation; then, you will update the plan with new activities and initiatives to correct the deviation causes. You will make the Kaizen plan available and visible to everyone.
10. You will make sure that the resources needed for improvement are made available to the teams, both in terms of time, skills, tools, budget, and material resources.

Seems hard work? Something that requires your full time? That's because that is exactly what it is. So instead of trying to do it all by yourself, again, it's time that you start getting people onto the Kaizen bus through a well-designed change management initiative.

A Framework for Agile Kaizen

There is still a last-but-not-least reason why Kaizen fails, although it might be implied in many of the previous topics. In many cases, Kaizen is applied to the company production processes, but that just means that we are able to produce double the crap, twice as fast, at half the cost. In order to make the company successful, Kaizen must also be applied to the product or service we are offering to our customers.

We could then group most of the Kaizen failure reasons in the following Kaizen framework:

- Lack of structure—**Retrospectives and Kaizen events**. This includes maintaining momentum and providing Kaizen resources.
- Lack of culture—**Team Kaizen**. This includes building the right mindset, purpose, and values into your people. Values include the Kaizen enablers: long-term vision, whole system view, quality first, and teamwork.
- Failure to identify and remove impediments or waste—**Process Kaizen**. This includes the use of Kaizen tools, definition of standards, and improvement plans in order to build things right.
- Failure to improve the product—**Product Kaizen**. Learning about value and customer expectations in order to build the right thing.

The order is not casual: if there is no Kaizen 'space' in the form of Kaizen events, it will be difficult to invest resources on the other elements. Then, if the team culture is not right, any investment in improvement in process or product will be suboptimal. Once we have the resources and the right attitude, we can engage in process: there is no way to build the right product if we are not building it right—remember, build quality first.

In the next chapters, we will study each of these framework elements and propose several tools to enhance them.

Summary

The number of things that can go wrong in a Kaizen transformation is so big that it turns true successful Kaizen into a huge competitive advantage for the companies that are able to achieve it, both because of the power of a company that improves continuously and because it is so difficult to imitate. These include the absence of a real culture, the presence of politics and blame games, general resistance to change, lack of momentum, no ownership or empowerment, short-term vision, failure to identify problems and root causes, failure to plan and execute, and lack of resources. Again, there

are many things that can go wrong, and the job of the Kaizen agents is to keep the process going.

To maximize the chances that the Kaizen transformation takes place, it is important that Kaizen agents understand the mechanics of change. Again, it is not about making change happen on your own, but making others take care of it. Finding the early adopters, nurturing the communities of practice, selling small wins to the early majority, and making it safe for everyone to join the Kaizen culture are key points for success. Having a Kaizen champion on your side will improve your chances dramatically.

Finally, to structure the Kaizen transformation, a Kaizen framework has been proposed—it covers Kaizen events, Team Kaizen, Process Kaizen, and Product Kaizen.

Things to Try

– This whole chapter can work as a checklist or guideline to improve the design of your Kaizen plan. Start by assessing the Kaizen failure reasons and see which of them are more likely to be found in your environment. Remember that several activities and strategies have also been proposed throughout this chapter in order to reduce the chance of failure.

– Debate how are you going to measure and evaluate improvement. Identify the key improvement metrics (quantitative and/or qualitative) and make sure they are measured frequently and made available to all. Identify the actions and events that move those metrics positively and the ones that affect them negatively.

– Even if there is no measurable improvement yet, you can assess the state of your improvement process—are people discussing new ways of working? Are there proposals on how to improve? Are there resources and support available for teams in order to seek improvement? If all the ingredients are in place and there's no improvement, review the list again and try to spot the most likely reason that Kaizen is failing.

– Learn about change. Review the proposed books and download the Fearless Change patterns available online,[11] then engage in storytelling exercises where people are able to identify stories that show examples of those patterns, especially if they took place in your company. Try to

[11] http://www.fearlesschangepatterns.com

foster those examples of change and see what the enablers were for those changes.

- Make sure that everyone is participating in the Kaizen initiative. Is there someone who is not included? Is there someone who is actually showing up to the Kaizen events but then not participating much? Engage in conversations, both with people not participating and with those around them, and try to spot hidden causes for this lack of participation.
- Widen your Kaizen scope to include the whole proposed Kaizen framework—events, team, process, and product. Identify the right way to approach every element of the framework. Maybe in order to start a Product Kaizen you need to enroll marketing and product people, whereas for Team Kaizen you might need to count on team leaders, coaches, or human resources.

Retrospectives and Kaizen Events

3

Improving the Way You Improve

Michael had scheduled a full day-off for that day's retrospective. When he arrived at the room, he was glad to see that not only the two teams he was coaching were there; several managers had come too. Michael had been working with them for a whole month in order to explain to them the importance of this event.

His teams were uncomfortable. Retrospectives were supposed to be a safe place to talk about impediments, and with all those managers around they were not sure if something had gone terribly wrong and they were going to be punished. Michael noticed the irony of the situation—this kind of behavior and values was exactly what he had to solve.

He started the meeting thanking everyone for their attendance. Then, he dedicated some time to make sure everyone understood that it was a safe environment: they were going to be talking about pretty uncomfortable things that needed to be talked about, and nobody was to be blamed for the problem, defects, impediments, or improvement areas they identified.

He then explained why there were more people than usual in that day's retrospective: he felt that the company culture was a barrier to true Kaizen transformation. He presented some stories about awesome companies achieving incredible transformations and asked everyone to rate their company against those stories. As expected, the marks were very low.

Michael was excited—they all agreed that something was wrong, but what? And what could they do about it? This was exactly the point he wanted to reach.

After a couple of hours, some conclusions started to emerge: it was not a matter of tools or practices, but more of a cultural problem. But where was culture? How did people learn about it? Why did they adopt this kind of culture?

This was a moment Michael had feared, but he decided to take a leap of faith. He confronted them with the uncomfortable truth: culture was set by example. They acted that way because everyone else was acting that way; and, despite the fact that many of those behaviors originated on middle management, the truth is that another big part of them was peer-induced on the production line.

There was a change in the meeting's mood. A moment ago they were engaged and excited, finding and learning, but suddenly most of them were on the defensive—'surely it was not ME who was setting a bad example!'. Michael broke the trend by introducing a couple of exercises and games designed to increase trust and vulnerability. Everyone had to step in front of the group and tell a story on how he or she was enforcing the old culture—the rest of the team had to clap and cheer for him or her as a matter of appreciation and acceptance. To say that it was awkward and uncomfortable would be just a polite way of describing it. But strangely enough, by the end of the exercise the finding-learning mood was back—the guilt and blame moment was over!

Michael felt a huge release. Things were actually going better than he expected. After a short break, they dedicated the next couple of hours to describe a desired cultural state and the purpose of such a change. They described the behaviors they would like to see on their peers and clustered them to identify common values.

Everyone was cheering up and then a manager said 'Well, that was a good job, now it seems time to call the meeting over'.

Michael kept silence for 10 s. The atmosphere turned tense. Then he smiled and said 'No... I have made that mistake before, and will try not to make it again. The meeting should not be over until we have an action plan. Right now we have described an improvement goal, but not WHO is going to care about it, HOW are we going to do it, WHEN are we going to be able to work on it, and WHAT is needed. In fact, in order to make that plan, we need to understand not just the goal, but the reasons we haven't been able to meet *that goal in the past, the root causes and impediments that we will be facing. The meeting is not over until we craft such a plan'.*

'But this isn't all'—Michael continued—'we've crafted plans in the past, only to see that there was not enough time to make them real. The same plan was proposed over and over until everyone forgot and lost interest in it. That needs to end today. The plan we decide together, we are going to make real together. To make sure that it happens that way, I will need you all to commit to a given, structured time, maybe every day, maybe every week. During that time, your duty will be to perform the improvement actions we decided together. Project managers will need to make capacity available for improvement actions, and managers will have to commit to not interfering in the improvement process, no matter how urgent or important they feel about doing something else. In fact, when some people are on vacation, don't we manage to carry on with the production by adjusting workload, capacity, and deadlines? This will be the same. We will have 'spare' time for improvement, no matter how little, and we will consider this improvement time strategic for the company's future, or we will just stop pretending we want to be Agile or Kaizen-minded'.

The look on their faces was a mix of emotions. They could not believe Michael was actually trapping all of them into his vision, but they were excited to see that someone was actually taking the bull by the horns. Was this the beginning of something?

Michael wished it really were.

The Use and Aim of Kaizen Events

Once again, I must reiterate that Kaizen is not about isolated events and good intentions. The most common Retrospective failure cause I see when coaching Agile companies is that there is no action plan and no follow-up. It reminds me of the good old 'strategic planning' events, where a huge investment of time and money was spent to craft a master strategic plan for the company, and then nobody cared about making the plan real—until next year's strategic planning event. Another good example is the hyperdetailed Gantt chart crafted at the beginning of a project that never gets updated after the kick-off, and thus becomes obsolete and useless.

In preparing for battle I have always found that plans are useless, but planning is indispensable.
– Dwight D. Eisenhower, 34th US president

Kaizen events are an important part of Kaizen, but they are not the whole Kaizen idea. As we established in the first chapters, Kaizen is more a mindset and a culture than a set of defined techniques, processes and tools—which, again, does not mean that there are no Kaizen techniques, processes and tools, or that they are not important.

Still, if you want to start a Kaizen change program at your company, after you've had a couple of hallway conversations with early adopters, probably one of the first things you should be doing is to institute some kind of periodic Kaizen event and make Kaizen official.

Kaizen events are used to structure, plan, and check the Kaizen efforts. If we use Deming cycle (plan, do, check, act) as a metaphor, Kaizen events could probably host every activity except the 'do'. The actual Kaizen improvements must be performed on the production line on a daily basis. You could or even should also plan, check, or act on improvements on the production line: Kaizen is not a one-shot, confined and time-bounded activity, but a whole-life continuous attention to excellence, improvement, and perfection.

In Kaizen events we create improvement plans, check their results, and correct them in order to keep searching for the desired results. We identify impediments, break them into smaller parts, analyze root causes, and imagine ways of getting rid of them. We plan how to implement those ways, and we define strategies for that implementation. Then we can use the same periodic Kaizen events to follow-up on the implementation. We can also use Kaizen events to analyze improvement metrics and define new, better metrics that better suit our needs when the improvement focus switches.

Kaizen events can be compared to the Sprint Planning/Sprint Review meetings in Agile frameworks like Scrum: there's planning, checking, and correcting, but the plan must then be implemented during the whole Sprint. No actual development is made in those meetings, yet they are important to reflect on and steer the project.

Kaizen events, as opposed to Kaizen, are limited in scope, time bounded, and usually include only a small subset of the company. The aim of Kaizen events is reduced, and they usually seek to build enthusiasm, synchronization, information, and knowledge exchange, bringing different perspectives together and reaching consensus, alignment, and commitment.

It is important to stress the importance of having multiple points of view when looking to improve your company. Companies are complex environments, and so the Law of Requisite Complexity applies: *If a system is to be stable, the number of states of its control mechanism must be greater than or equal to the number of states in the system being controlled.*[1] This means that a complex environment (your company, formed by many individuals) cannot be controlled or improved by a mechanism that is less complex (i.e., just one individual).

Different Kinds of Kaizen Events

Several different kinds of Kaizen events can be used to enhance your Kaizen transformation. There are Kaizen events more suitable for project improvements; others are aimed at functional, local improvement; and still others can have a cross-functional, system-wide focus. There are periodic Kaizen events and one-shot ones.

Some of the Kaizen event formats are:

– **Kick-off meetings**: used at the beginning of projects to create a common understanding of the project scope, goals, success criteria, and so on. Kick-offs can be used to define improvement goals over previous projects and to remember our lessons learned and how to apply them to our next project. The goal here is not to perform new projects the same way we have been performing them in the past, but to search for new, better ways of both managing and executing projects.
– **Project post-mortems**: once the project has ended, there are few things we can do to improve them, but a post-mortem analysis is a good opportunity to learn what we did well, where we failed, and what can be improved in the future. Then, we can prepare ideas and improvement plans to be used at the next project kick-Off.

[1] Ashby WR (1956) *An Introduction to Cybernetics*, Chapman & Hall.

- **Inceptions[2] or joint application development meetings**: these meetings are a great opportunity to improve our products and our value-generation process. The idea is not to create a product design, THEN show it to the customer, but to create a product design WITH your customer. These workshops include several activities to generate a common vision and gather insights, and they are a great opportunity to define success and improvement from the client perspective.
- **Client feedback events**: similar to inceptions, we only perform them once the project has already started or even when the product is already on the market. Sometimes they take the form of focus groups or customer interviews. The idea is to gather insights on how to improve our service and product.
- **Labs**: one of my personal favorites. The idea is to define a time-bounded space during the iteration where the team can stop working on their current project and spend some time doing research, innovating, learning, or developing their own skills. I usually start with 4 h every 2 weeks, which represents as little as a 5 % time investment. During lab time, there is an endless list of things the team could be doing in order to improve, including the design of new products, proofs of concept, or prototypes; implementing new tools; learning new technologies or techniques; working in pairs in order to exchange knowledge; developing and improving their work standards; reading books and presenting them to their peers; fixing impediments they are not able to fix during regular work time; reviewing each other's work; or developing and maintaining knowledge bases.
- **Backlog grooming**: another interesting activity, especially in very technical or complex environments. The idea is to get some structured time to review the team's backlog and prepare for the next iteration planning meeting so there are no surprises. Sometimes we detect some work package that is coming that we don't fully understand, so we decide we'll be using some time to investigate it before iteration planning, or sometimes we realize that the work packages are not well described, include dependencies with other teams, are too big to be managed, and so on. The idea here is to improve the team's backlog over time and before it's too late to react to the backlog defects.

[2] A good resource on inceptions is Jonathan Rasmusson's *Inception Deck* at http://www.slideshare.net/dleyanlin/99-inceptiondeck

- **Root cause analysis (RCA)**: these can be special events where we focus on one given problem and try to find hidden root causes that we have not been able to understand yet.
- **Workouts**: similar to the RCA or even complementary to it, the idea is to get several people together to work on a given problem until they are able to propose an improvement plan. Then, they will usually be asked to implement and follow up the plan. Sometimes they are referred to as 'Kaizen Blitz'—blitz meaning lightning in German—because of the fast improvement that is sought through this kind of event.
- **Off-sites**: again, similar and complementary to RCA, workouts, and some other Kaizen events—you can combine them all in different ways. The idea of the offsite is that, in order to be focused on Kaizen and not be interrupted by the daily short-term 'noise', we move the improvement team to an offsite location, typically with restricted access to mobile phones, laptops, the Internet, and daily tasks.

Agile Retrospectives

Of course, the canonical Kaizen event for Agile teams is the iteration Retrospective.

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

Retrospectives are short meetings at the end of the iteration where we typically review what Agile teams call 'Pluses and Deltas': things that we did well and we must enforce in the future, things we failed at and we should correct in following iterations. The team or their coach will log pluses and deltas in an Impediment Backlog; then, they will use that backlog to perform improvement activities over the next iterations. Some Agile frameworks even call for some given capacity of the team to be spared over the iteration in order to work at the Impediment Backlog. This 'improvement capacity' can take the form of backlog space, lab time or out-of-iteration-focus activities.

Retrospectives are usually an intimate event. If managers, product owners or other stakeholders are present, they can alter the Retrospective results because the team might not feel confident enough to talk about failures, defects, impediments, problems, or bad behaviors. The need for this intimacy depends on the company culture: very mature companies might want an all-heads Retrospective, but assume that this is very rare and most organizations will benefit from more private Retrospectives—just the team and their ScrumMaster. On the other hand, if nobody knows what is happening at a Retrospective, a feeling of 'wasted time' can happen at managerial levels. A good start is to perform the Retrospective in a closed, secure environment, and then create a charter of impediments found and improvement actions planned. This charter can then be disclosed to others.

Teams might choose to invite someone to a given Retrospective. This must not be understood as a lifetime invitation pass. Maybe they need a manager for one Retrospective, but they might not feel that need any more after the Retrospective. Agile coaches must act as Retrospective gatekeepers and make sure that nobody is offended for not being invited to a Retrospective or even for being kicked off.

As said, in a more mature environment—Agile maturity, that is—people can feel confident enough to declare Retrospectives open to all. Agile coaches must make sure that, if this happens, Retrospectives remain honest, transparent and that no topics are actually taboo.

The typical Retrospective dynamic is:

– **Opening of the Retrospective**: welcome the participants, explanation of the set-up, Retrospective focus, and expected behaviors or attitude. It might include an opening exercise—an ice-breaker or energizer.
– **Telling the iteration story**: what happened during the iteration, things we did well, things we did wrong, things that might be done differently. Several information sources can be used for that: the burndown chart, cumulative flow diagrams, notes on the team's iteration board, happiness indexes,[3] or personal team member notes.

Unfortunately, many Retrospectives just end here. Of course, that's a wrong way of performing Retrospectives. Some more advanced teams will include a couple of steps:

– **Identify impediments**: it might take the form of brainstorming, or they might already be identified and clustered from the iteration analysis.

[3] Check the second part of this book for some team activities and tools that will help.

– **Add impediments to the backlog**: impediments will be logged and prioritized. The team will choose some of them and will work on them during the iteration.

And again, this is a good improvement on how to perform Retrospective, but there are still a couple of key points missing.

Retrospective Failures

Of course there are several ways a Retrospective can go wrong, but here are the most usual ways I find when coaching Agile companies:

– **Pluses and deltas**: the team identifies strong practices and weak spots. . . And then what? There is no actual analysis, no plan to enforce the latter and reduce the former, no follow-up mechanism.
– **Groundhog Day Retrospective**: in the cult movie *Groundhog Day*, Bill Murray was forced to live the same day over and over. In this Retrospective failure mode, something similar happens: all Retrospectives seem like the same Retrospective. The same problems are identified and reported over and over, and nothing really changes.
– **Playground Retrospective**: the team uses Retrospective time for playing team-building exercises, Agile games, and so on. But there is no real improvement effort. It's just the team's way of dissipating some stress after the iteration, or just some form of 'cargo-cult' Retrospective, meaning that they mimic what other Agile teams are doing or what the literature describes, but they don't really understand why. Hence, they end up performing empty, useless activities, or they just pursue some other goals not necessarily related to continuous improvement.

In the South Seas there is a cargo cult of people. During the war they saw airplanes with lots of good materials, and they want the same thing to happen now. So they've arranged to make things like runways, to put fires along the sides of the runways, to make a wooden hut for a man to sit in, with two wooden pieces on his head to headphones and bars of bamboo sticking out like antennas – he's the controller – and they wait for the airplanes to land. They're doing everything right. The form is perfect. It looks exactly the way it looked before. But it doesn't work. No airplanes land. So I call these things cargo cult science, because they follow all the apparent precepts and forms of scientific investigation, but they're missing something essential, because the planes don't land.

– Richard Feynman, *Cargo Cult Science*, 1974 Caltech commencement address

- **Wailing Wall Retrospective**: people will complain about how everything is so wrong, but they won't propose action plans. They will usually argue that problems are not their fault and blame management.
- **Weak planning**: the team will prepare some action plan, but the plan will be weak. Usually they will use statements as 'we are going to solve this', 'we are going to work smarter' or 'we will put some effort on it', but then there will be no clue on who, what, when, or how.
- **Leader talks**: the coach or—even worse—some manager will make all the iteration analysis, problem diagnosis, impediment identification, and solution proposal. The team will not feel empowered, nor they will feel any ownership in the improvement process. Kaizen is limited to what the leader is able to achieve on his own. A variation of this happens when the facilitator is too involved and creates a bias in the analysis and the proposals—he will discard ideas that do not match his own.
- **Few people talk**: a variation of the previous type of failure. In this case a few people participate, but some will always lag behind or just be idle. The facilitator is not fostering the collaboration and commitment of the whole team. Maybe there's a hidden team dysfunction.
- **Artificial harmony**: the team's fear of conflict makes them hide real problems. They feel like pointing out impediments or defects will bring conflict and make others feel bad, so they prefer to just let things be as they are.

– **Bad feeling**: Retrospectives are seen as uncomfortable events because all the talk is about failures, defects, problems, and things we did wrong.
– **Too short**: Retrospectives are too short usually because of a vicious cycle—there is not enough time to perform a proper Retrospective, so results are weak, so people think Retrospectives are a waste of time, so they don't use too much time on the Retrospective—and the circle is closed. Some Agile coaches recommend a minimum of 30 min per iteration week (2 weeks equals 1 h). According to my own experience, that is too short and will cause any of the failed Retrospectives modes I mentioned before. My own rule of thumb is closer to minimum 1 h per iteration week (2 weeks, 2 h Retrospective), and usually even more.

Improving Retrospectives

In order to improve Retrospectives, first you have to make sure that you are creating the right environment:

– **Good facilitation**: because of their intrinsic nature, Retrospectives will move people out of their comfort zone and will ask them for answers they don't have at the moment. In such a situation, it is easy for the team to get stuck or to talk in circles around the problem without reaching any meaningful or useful insight. Good facilitation can reduce this risk by time-boxing activities, ensuring everyone participates, providing frameworks and tools for constructive reasoning, teaching everyone how to discuss issues positively, or making teams deal with conflicts. Some good facilitation resources are mentioned at the end of this book.
– **Make it secure**: if people have the feeling that something they say can be used against them, can harm some people's feelings or might create conflict in the team, it's likely that this issues will never come to surface. Facilitators must also make sure that there will be no finger-pointing or blame games during Retrospectives. Many teams start their Retrospectives by writing on their boards Norman L. Kerth's Retrospective Prime Directive.[4]

[4] Kerth NL (2001) *Project Retrospectives: A Handbook for Team Reviews*. Dorset House.

Regardless of what we discover, we understand and truly believe that everyone did the best job they could, given what they knew at the time, their skills and abilities, the resources available, and the situation at hand.

– Norman L. Kerth, *The Retrospective Prime Directive*, http://www.retrospectives.com/pages/retroPrimeDirective.html

– **Positive intention**: create a climate of nonviolent communication. Watch for aggressive gestures, bad words, name-calling, or personal mean-spirited attacks. Try to maintain the Retrospective in a fact-based state of genuine curiosity—trying to learn, understand and find ways of doing things differently—and don't let the team run into frustration or defensive modes. Teach them how to give good feedback—a good message can be spoiled by a bad form.

– **Enforce participation and collaboration**: everyone must participate in the Retrospective and, even if they don't fully agree on the action plan, some consensus must be reached. This means that sometimes people will be asked to work into something they don't fully believe in. To do so, a climate of trust, collaboration, and teamwork must be grown: sometimes I will trust the team and go for something they believe in; sometimes it will be the other way round. Looking for 100 % buy-in and agreement might make Retrospectives eternal.

– **Solution focused:** be solution focused more than problem focused. The motto here is 'problem talk creates problems, while solution talk creates solutions'. If you focus too much on the problem inevitably sooner or later you fall into negativity. With a focus on solutions, you foster creativity, excitement, and enthusiasm; even more: you will be moving people into action and changes—at least in their minds.

A good Retrospective environment can also dramatically improve the Retrospective results—or kill the Retrospective completely! Some of the environmental factors that can help your team get into 'Kaizen Mode' can be:

– **Isolated space**: safe environment where the team feels free to talk about anything with no 'unwanted ears'. People outside the team coming to Retrospectives without being invited can kill Retrospectives.

– **Out of their regular space**: moving to a different space from the one they occupy during the iteration helps the team move their minds to an observatory/inquiry state.
– **Lot's of idea space**: use whiteboards, walls, even desks. Don't let your team run out of space when brainstorming and expanding their minds! Some space to walk around or stretch is also nice, as is having sofas, armchairs, and other ways of feeling comfortable and even cozy.
– **Props**: sticky notes, adhesive tape, markers, paper, and scissors. And never let the team forget to bring their notebooks—there will be homework at the end of the Retro!

On the other hand, in order to maximize the chances of having productive Retrospectives, there are some steps to be added to the usual Retrospective structure we discussed in the previous section. An improved ten-step Retrospective process aimed to reduce all the failure modes and enforce the desired Retrospective mindset would look like this:

1. **Prepare**: I usually ask both the team and their facilitator: 'don't show up naked at the Retrospective'. This means that, during the iteration, everyone should be preparing topics to be discussed at the Retrospective, measuring improvement, following up on previous plans, etcetera. All this information should be reviewed before the Retrospective, and the relevant data should be brought to it. If there is currently a special or specific improvement focus, the Agile coach—Kaizen agent, team facilitator, leader or similar role—should remind people about it and engage in enough conversations to ensure that there will be something to discuss at the Retrospective. Daily synchronization meetings—stand-ups, daily Scrum, or similar—are a good opportunity to remind people to prepare for the next Retrospective.
2. **Set up**: make sure everyone understands the Retrospective goal and the desired environment of trust, collaboration, transparency, and positive, constructive energy. It's also usually a good idea to 'test the waters' before going on. Sometimes, it's just not the right moment to retrospect and could be a good idea to suspend the Retrospective and do some team coaching instead. If the team is too frustrated or too depressed, you might want to change the Retrospective dynamic on the run (or just postpone it and buy the team some beer).
3. **Remember, remember**... My personal rule is to start reviewing last Retrospective's action plan and see if we were able to perform at least

part of it. It not, there is probably no sense in running a new Retrospective the same way—we should go back to the last one and see what are we failing at when it comes to actually performing the improvement plans! If this means that we don't have enough time to review the last iteration, that's fine: if the Retrospective process is broken, there is no use in doing more broken Retrospectives until we fix them. If we were able to perform some part of our improvement plan, we might spend some time following the plan's roadmap and designing new actions in order to continue improving that way.

4. **Tell the iteration story**: assuming we were able to at least try part of the improvement plan, then it's time to review our last iteration. Facilitators must make sure that, during the iteration, enough information is recorded so we can actually reconstruct the iteration story. People usually rush to 'pluses and deltas' mode, but I always find that some narrative helps the team to put them into the right context—that is, tell me the *story* of the iteration: how did we start, what was our goal, what happened then, how did we end...
5. **Highlight events:** after the full iteration story has been told, try to spot the most relevant events that impacted on team dynamics, project performance, product quality, and the overall improvement process. You can cluster them in similar categories, but don't spend too much time analyzing yet.
6. **Divide and conquer:** the first thing is to set a relative priority of the detected impediments. Priority can be established through a corporate strategy, a team-decided improvement focus, or through the use of several frameworks. I usually start by assessing impact on the team versus ability to remove the impediment—high-impact easy-to-remove items come first! Once you spot the top-priority items to improve or, if we are talking about positive impacts, enforce and evangelize to other teams, it might happen that they are too big to be dealt with. In such cases, trying to divide them into more suitable, actionable problems might be a good strategy.
7. **Root Cause Analysis (RCA):** once we have decided to work on a particular item, we should make sure that we attack the roots of the problem and not its symptoms. Read more about RCA in Chap. 5—Process Kaizen.
8. **Plan to improve:** now that we have found actionable issues we can work on, it's time to brainstorm actions we can take to eliminate or reduce them. The plan should, at least, tell us *what* are we going to do,

how are we going to do it, *who* is going to take care of it, and *when* will we perform this actions. Success criteria, improvement goals, metrics, risks, strategic partners, needed resources, or a desired state might also be defined in the improvement plan. The plan should have a special focus on what are we going to do during the following iteration, but it should also include a roadmap after that. Once the plan is set, if we have enough time, we can try to root-cause analyze another issue, following the priority order, and craft another improvement plan for the iteration.

9. **Closure**: for a great Retrospective closure, ask for appreciation and feedback. Appreciation consists of asking everyone to thank one or more team members for their contributions during the iteration. It can be about their work, their attitude, how they collaborated, how they solved an impediment or even about the day they brought doughnuts for everyone: the idea is to enforce behaviors and work on team spirit. Feedback, on the other hand, is about this Retrospective: are they happy with the results? What can be improved in future Retrospectives? How can we make Retrospectives better? What did we do differently that we liked? It is important nevertheless that we assess the team's feelings about any conflict related to the last iteration. Facilitators must try to bring real closure: we reflected about the past, we talked about conflict, we found ways of dealing with this kind of conflict in the future, and now the conflict is over. If the team starts the new iteration still feeling frustrated about the last one, very probably there will be new conflicts—or even worse, we will repeat the last ones.
10. **Follow-up**: as with preparation, daily meetings are a great place to ensure that the improvement plan is being performed. Some teams even create an 'improvement lane' on their iteration boards to keep track of the desired improvement activities, so people get to report on daily meetings about these activities and the impediments they are facing.

A Note on Positive Issues

As I wrote before, 'problem talk creates problems', and the whole Retrospective literature is heavily biased towards deltas—things that we have to solve and problems to analyze. In many cases, it seems that we only list pluses to celebrate and balance the negative feeling created by all the impediments and defects that we detect over the Retrospective process.

The idea, of course, would be to enforce the things we are doing right and to ask ourselves how can we do more of it, what were the root causes of the observed improvement, and what other similar actions could we perform in order to maximize those improvements. We should also care about sharing that knowledge with other teams, something we can do through the communities of practice or with the help of Kaizen agents/Kaizen evangelists.

Another important use of the pluses is to communicate all the small wins to the company as part of the Kaizen change initiative/Kaizen plan. We are trying to build a culture by telling stories, and the stories about what went well, what worked, or what we were able to improve are important stories—stories about success.

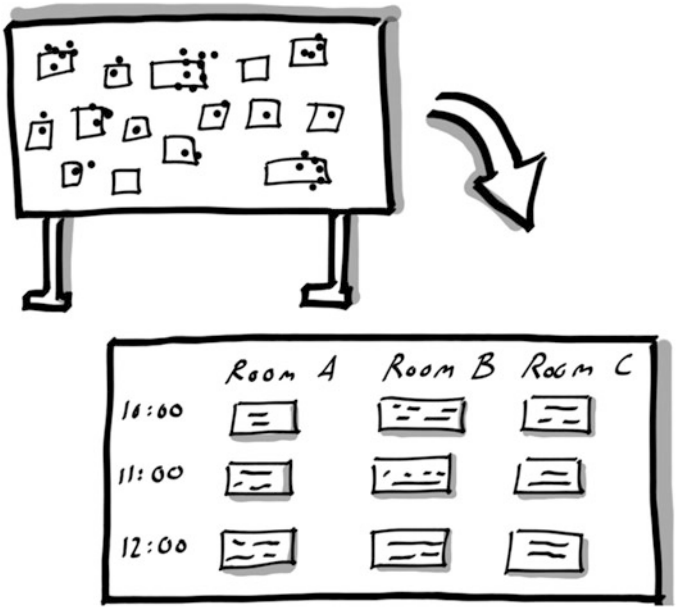
Some examples of positive approaches to Retrospectives and continuous improvement are solution-focused Retrospectives and appreciative inquiry. Solution-focused Retrospectives have a pretty self-explanatory name: the main goal is to explore and define solutions to any problem identified. A solution-focused Retrospective starts with positive formulation—'I want...' and 'We should do...' instead of 'I don't want', 'We should stop', which would be a negative formulation. Then, it moves into an exploratory stage on what would be the consequences of the positive change we are trying to achieve—how it would affect others. It also tries to figure out how close we are to our desired goal, what made us advance in the past, how can we increase that, what separates us from the goal, and how to overcome that gap.

Appreciative inquiry is somewhat similar in the sense that it focuses on exploring what is the best possible future state for all. This future state should be so compelling for everyone that no incentives, rewards, or persuasion would be needed to make everyone work toward it. The core idea of appreciative inquiry is that just asking people to 'fix problems' is not powerful enough to spark and sustain change throughout human organizations, and a commonly accepted idea of what could be—a noble cause, if we think in corporate cultures, or a purpose if we look at it from the motivation perspective—is needed to encourage and support change.

Special Retrospectives

In some occasions, special Retrospectives can be used to enhance the overall Kaizen process:

- **Cross-Retrospective**: there are several ways of doing a cross-Retrospective. One is to run a Retrospective with members of different teams so we can reflect on cross-team collaboration, dependencies, and issues. Another one would be to switch facilitators in order to bring new perspectives and techniques to teams that might be stuck in their thinking process. There's even another way: each team makes a Retrospective on some other team's issue, with no interference from them, so new insights can be generated 'outside the box'. In all cases, knowledge sharing and a whole-system view are the main goals of running such an event. A cross-Retrospective can be followed by a cross-cross-Retrospective, where different cross-teams inter-exchange their results or visit each other's Kaizen boards to learn about their insights.
- **Meta-Retrospective**: or 'Retrospective on Retrospectives'. The goal of this event is to review the Retrospective mechanics and evaluate if we are actually improving of if there is something missing. This whole chapter can serve as a checklist to assess the current Retrospective process of the team and to see if there is something we can fix, change, or add to it.
- **Focus Retrospective**: in this kind of Retrospective the team does not review the whole iteration or the existing improvement plan; instead they focus on just one issue for the whole timebox. It is a useful approach when there is some big, hairy issue that we never seem able to divide or analyze. Focus topics can include impediments, roles, behaviors, processes, tools, technologies, or even cultural issues. Bringing in experts on the topic or other kind of stakeholders can be useful in this kind of events.
- **Connecting the dots**: sometimes the team gets so frustrated with the current state of things that it might be interesting to create a 'roadmap from the past' describing how we have been able to improve over time. The main goal is to identify, revive, and analyze past milestones and crucial improvement events from the past and see how can we achieve similar improvements in the future.
- **Open Space**: this is a massive all-hands Retrospective that can be conducted using the Open Space format. Several topics are proposed by participants. They are dot-voted and prioritized, and the ones with the most votes are placed in a schedule to be addressed during the rest of the session. People are then free to join any group they find interesting, and the proposers of the topics are responsible for hosting the session and wrapping up the results.

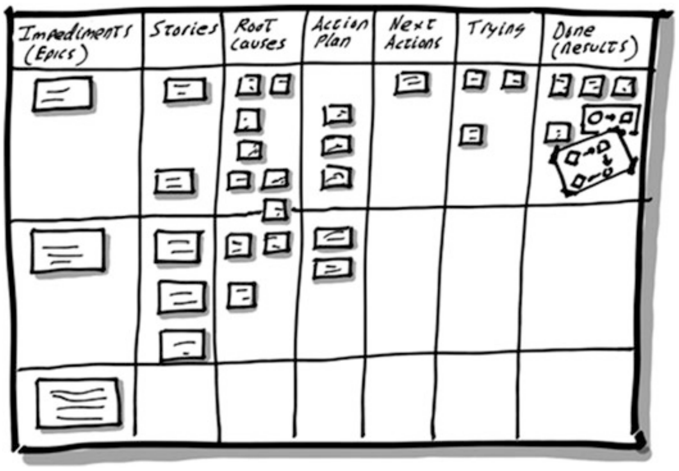


Of course, feel free to combine, modify and experiment on different Retrospective formats, techniques, or scopes that suit your needs! Remember that the goal is not to 'run Retrospectives' or even to improve—the goal is to create a *culture* of improvement and, in order to do so, change must be the rule—no change equals no improvement. If you ever feel like you are doing the same Retrospective over and over, do not hesitate to spice it up with any of the approaches discussed in this chapter or the tools and techniques described in the second part of this book.

Communicating Your Improvement Plan

Even if there are still no improvements to show, it is very important that the team communicates the impediments they are finding, the root causes spotted, the action plans defined, and the actions they are actually performing. Sometimes Retrospectives are kept secret for the sake of team trust and team collaboration; but if results are not shared, there is a chance that managers—and the rest of the organization—will start feeling that Retrospectives are a waste of time and that they are only an excuse for the team to goof-off.

A good way of communicating your team's Retrospective results is to ask your team or their facilitator to craft a poster during the Retrospective, listing all the major insights that were pointed out. Another way would be to maintain a Kaizen board, were the team lists the Impediment Backlog, root causes detected, improvement plan, and actions performed.



Several other methods can be used to communicate your plan, both push communication—making information reach your audience—and pull communication—having information ready for anyone who wants to consult your improvements. Some other tools and communication formats you might want to try include:

- **Improvement wiki page**: you could easily set up a wiki page and log your Retrospective results, impediments detected, and so on.
- **Improvement newsletter**: a short description of every Retrospective could be e-mailed to a list of interested people and stakeholders.
- **Improvement forum**: an online forum could be created and managed to allow discussion on improvement topics.
- **Kaizen corner**: a fixed place and time to give a short speech about the team improvements to anyone who wants to attend.
- **Coaches community**: all team coaches could perform a short stand-up meeting after all Retrospectives and share a short briefing of their own Retrospectives.

Summary

Kaizen is not about one-shot time-boxed events, but Kaizen events are an important part of any Kaizen initiative. They provide structured time to focus on long-term improvement and can help us build buy-in, grow enthusiasm, bring different perspectives together, reach consensus, and plan for improvement. Of course, Kaizen events must be followed-up, and enough daily capacity must be ensured to carry on the changes and improvements designed at Kaizen events. Daily meetings are a good way to build up momentum, make sure the Kaizen plan is being performed, and deal with unexpected impediments.

Retrospectives are a typical Kaizen event found in Agile environments, and they usually focus on detecting good and bad things that happened during the last iteration (typically, the last 2 weeks). To have really useful Retrospectives, several environmental factors must be present—collaboration, trust, transparency, ownership, security, positive attitude, good facilitation—and a solution-focused, action-driven approach must be enforced. A good physical environment also helps.

Instead of just finding impediments, a good Retrospective focuses on the most important impediments in terms of impact and ability to be solved. The most important impediments are divided and root-cause analyzed in order to create action plans that identify both short-term actions and mid- to long-term improvement roadmaps. These action plans include details such as what is to be done, who is going to do it, what is needed, and when it will be done.

Again, follow-up is key. Communicating the plan and making others conscious of your results and key insights is also very important so everyone can see the benefits coming from the Retrospective. Kaizen boards are a good way to begin communicating your improvement plans, and they help to build a Kaizen culture, while becoming cultural artifacts.

Positive wins and detected improvements are also an important part of the Kaizen initiative, as they can help sustain the improvement momentum and help us define new improvement stories that build the Kaizen culture. Cross-pollination of ideas and knowledge sharing are two factors that Kaizen agents and team coaches must observe—a cross-Retrospective is a good way of approaching this need.

Things to Try

- Make a formal kick-off for your Kaizen initiative, or if you did that in the past consider re-founding the Kaizen plan. Prepare it in advance—you should find plenty of ideas of what you want to tell in this book. Gather all the interested stakeholders and use the kick-off as a first Kaizen workshop to define your desired state, main impediments, long-term goals, short-term action tracks, and follow up structures.
- Ask your people to rate their own Retrospectives. How happy are they with dynamics, scope, and results? Is there anything they are missing? What's going wrong at the Retrospectives? What was their best Retrospective so far, and why? Use this information to foster dialogue around new ways of improving Retrospectives.
- Check the Retrospective failure modes and see if they could be happening in your environment. Ask your people to rate their Retrospectives against them (0—no, we never do this; 10—yes, that's our Retrospective all the time). Ask them how they feel about it—maybe it's not a big deal—but if they feel like improving their Retrospectives, ask them why they are falling into this behavior and how can they move out of the identified dysfunction.
- Review your teams' Retrospective environment and see if it matches the guidelines explained in this chapter. Ask your teams to rate their environment and propose changes or improvements to it in order to have better Retrospectives.
- Introduce the full, improved Retrospective process described in this chapter to your teams. Ensure that they allocate time in the Retrospective for prioritization of issues, RCA, and improvement planning. Make sure they follow up during the iteration—use daily meetings for that. Review the RCA tools in the next chapter.
- Introduce 'Special Retrospectives' as needed—for example, at the beginning of new project (kickoff, inception), the end of project (post-mortem, client feedback), or for dependencies spotted (cross-Retrospective).
- Improve your team facilitator's toolbox and skills—check the list of resources at the end of the book.
- Find a local Agile group—they are pretty much everywhere, and every major city has one. If you cannot find a group or are not able to join one for whatever reason, join an on-line Agile forum, list, or social network. You can then exchange knowledge and information on how to improve

Retrospectives with other Agile practitioners—maybe even invite some-one to your Retrospectives or be invited by them in order to learn new strategies, tools, frameworks, and games.

– Design your Kaizen board. Play with the idea—don't stick to a predefined design. Try to make it dynamic and collaborative. It is also important that people outside the team are able to understand what are the team is working on: use results-oriented language and make sure that the terms are not too technical.

4 Team Kaizen

Improving the People Who Make Your Stuff

Two weeks had passed since the 'Foundational Retro', as everyone now called it, where Michael had moved the Kaizen plan of the company to a new level. Michael was really happy with the results, and nearly everyone was sharing a sense of novelty and enthusiasm.

Of course, some people were skeptical and believed that the excitement would last only until a new crisis kicked-in and everything returned to the old-style way of working. And of course, there were some people who just refused to take part on the improvement plans and claimed to be 'too busy to waste time in improvement meetings'. When thinking about those people, Michael always remembered the story of being 'too busy chopping wood to waste time sharpening the axe'.

Michael decided to bring his efforts to the majority and expect that the laggards would follow sooner or later: if there wasn't a majority of people on the Kaizen bus, there would be no reason to argue with laggards about their need to jump on.

Michael was in fact more concerned about something he was noticing in the current iteration. Managers had indeed joined the Kaizen initiative, but they were still basically working on their own—there was no real team spirit once he looked above the production line. Old structure, politics, and power games would die hard; he already expected that.

But even when he looked at the production line, he was worried to see how quickly the teams divided any work into parts that they could then assign to each other on their own without really collaborating. Sometimes, when staying with a team for a morning, he could not see anyone talking to *others except for the 10-min daily synchronization meeting; and these meetings looked more like individual reporting than real team collaboration. He could see how individuals pulled some task from the team's iteration board and then worked on it silently until it was over, reported about it, and then just pulled the next available task.*

Now that he was starting to understand the roots of a Kaizen culture, Michael felt like there was no real 'team' in his company—just a bunch of people working at the same corner and with some similar duty. He read and read, but he always bumped into the same statement: there is no Kaizen without real collaboration of every employee.

On the other hand, Michael had tried to start a 'Lab Time' project: he convinced a manager to try it on his people, but then when he brought the good news to the teams, they refused to try labs because project deadlines were too close. Michael could understand their commitment, but he couldn't see how they were going to improve their skills if there was no structured time where they could devote themselves to deliberate practice, learning, experimenting or sharing knowledge with each other.

He could also see how meetings were very poorly facilitated and how people were not able to quickly reach consensus. The same topics were discussed over and over with no conclusions, and people seemed more concerned about being right than agreeing with something they could try. Teams also feared conflict, and they preferred to pass on uncomfortable issues—so these conflicts were not addressed, nor were they improved.

Michael made a mental list: technical skills, soft skills, team collaboration, personal communication, dealing with conflict… He knew that if Kaizen was to happen, he had to start worrying about People Kaizen—about Team Kaizen!

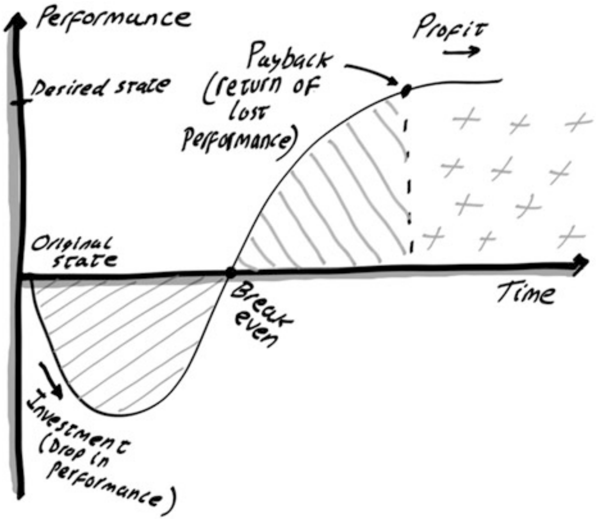
Kaizen and the Human Brain

Your brain is not your friend. Period.

The human brain has evolved to save as much energy as possible. The brain itself is a huge energy consumer (20–25 % of basal metabolism in humans) and is still hard-wired to behave as those of most primates do. Part of this hard wiring implies that the brain will always favor immediate reward when confronted with delayed gratification.[1]

In my first book, *Agile Management*, I included a brief explanation of the J-curve, a common behavior of systems when you introduce some change: performance first drops as the system adapts to change, then it rises when the change takes enough time to show some results. There will be some losses during the 'investment' stage, and we expect to recover those losses when the system starts to show some return on investment.

The J-curve can also be used to explain short-term versus long-term investment. In the short term, it is just better not to invest, as you will have more resources available. But in the long term, investing returns more profit.


Performance
Profit
Payback (return of lost performance)
Desired state
Original state
Time
Break even
Investment (Drop in Performance)

In the same way, when forced to decide between eating a doughnut now and looking good in swimsuit at the beach in 6 months, the brain will always prefer the doughnut—short-term thinking. The same happens when you have to decide between buying the latest gadget or saving money for retirement—and the examples can go on and on forever.

[1] Mischel W, Shoda, Y (1995) A cognitive-affective system theory of personality: Reconceptualizing situations, dispositions, dynamics, and invariance in personality structure. *Psychological Review*, 102, 246-268.

In a related experiment, Stanford University psychologist Walter Mischel conducted the now famous 'Marshmallow Experiments',[2] where he tested several kids by offering them a choice between a small instant reward and a bigger delayed reward. Kids who were able to tame their brains and wait for the delayed reward had better life outcomes over time, including better SAT scores, body mass indexes, jobs, and emotional lives. These results have been replicated by several subsequent studies.

Another example is change. The brain does not like change. Even when it's able to dramatically reconfigure itself (*neuroplasticity*[3]), the brain tries to avoid any changes in environment and reacts aggressively to external threats to our comfort zone.

Even when it comes to the human condition, we have been using our brains as hunter–gatherers for 90 % of our species' history—until 12,000 years ago, all humans lived that way, and a few contemporary societies are still classified as hunter-gatherers.[4] In this situation, the brain knows that you must save all available energy until, eventually, prey will appear and you'll need to spend your energy chasing and hunting it. The results are so strongly hard-wired that, in a series of experiments using MRI scans and similar technologies, scientists have found that the human brain prefers high-calorie junk food even when it tastes worse than healthy, low-calorie alternatives.[5]

For these two reasons (immediate reward and saving energy), it is always more desirable for the brain to lay down, watch reality shows and eat junk food than run in the rain—which is, of course, better for your health and, arguably, more enjoyable afterwards. Nobody seems to have told the brain that we, office workers, sit down for 70 % of our work day,[6] and that there is no need to save energy to run after buffalos any more. Thus, nowadays there

[2] Mischel W, Ebbesen EB, Zeiss AR (1972) Cognitive and attentional mechanisms in delay of gratification. *Journal of Personality and Social Psychology* 21 (2): 204–218.

[3] Pascual-Leone A, Amedi A, Fregni F, Merabet LB (2005). The plastic human brain cortex. *Annual Review of Neuroscience*, 28, 377–401.

[4] Lee, RB (2005). *Cambridge Encyclopedia of Hunters and Gatherers*. Cambridge University Press.

[5] Araujo IE, Lin T, Veldhuizen MG, Small DM (2013) Metabolic Regulation of Brain Response to Food Cues. *Current Biology* 23(10):878–883.

[6] Duncan M, Kazi A, Haslam C (2012) Office Workers Spend Too Much Time at Their Desk. *Science Daily– British Psychological Society (BPS)* (2012, January 15).

are more people killed by obesity than from malnutrition—three times more, if we do not count sub-Saharan Africa[7]

Coming back to our topic, the brain is not inclined to improve or change. Kaizen is hence an 'unnatural' state of mind—as are studying, doing homework, or working out. Only through tough discipline and strong will can we overcome the barriers of our lazy brains.

I've come to find that all physical and mental activities that require discipline and practice over time are a good way of cultivating a Kaizen mind. It doesn't matter if you train through martial arts, prepare for a marathon, or teach yourself new languages: the regular workout of your mind and body contribute to create a brain that is more used to long-term delayed gratification activities and, hence, more is prone to improvement and change through 'sacrifice and self whipping.

The Team Is the Thing

Everyone seems obsessed these days with forming teams. If you ask your company why are they forming teams, they will probably answer 'teams are better'. If you keep asking why, they might even suggest that teams are more productive. But ask 'why' once more, or ask them how do they know, and they will probably be shocked. And this is really sad, because if you don't understand the mechanics behind this so-called 'hyper-productive teams', you are more likely to form dysfunctional teams in a form of 'cargo-cult'—copying some other companies' structures, tools, and techniques, without really understanding them, and hoping that the magic will happen anyway.

Of course, several case studies over the last decades have shown the benefits of creating teams when facing complex situations, both in knowledge-based environments and in more industrial ones. We could mention the great results in Lockheed Martin with the Skunk Works approach[8]; the Scrum team strategy described by Nonaka and Takeuchi[9] and, later, adopted by Sutherland and Schwaber when creating the Scrum

[7] *The Global Burden of Disease: Generating Evidence, Guiding Policy* (GBD, 2010).

[8] Rich BR (1996) *Skunk Works: A Personal Memoir of My Years of Lockheed*. Back Bay Books.

[9] Nonaka I, Takeuchi H (1986) The new product development game. *Harvard Business Review* 65(1):137–144.

Agile framework[10]; the high-performance teams described in Jim Collins' *Good to Great*[11]; or the factory teams and 'cells' in the Toyota Production System.[12] We could also dive into the vast array of management books by experts like Kotter, Drucker, Friedman, Lencioni, Peters, Goldrat et al., only to find the same statement: teams are the best approach when facing complex environments and situations.

It is teamwork that remains the ultimate competitive advantage, both because it is so powerful and so rare.
– Patrick Lencioni, *The Five Dysfunctions of a Team*

The problem comes with the interpretation of the term 'team'. For example, in motorcycling there are 'teams' for every sponsor or maker, but inside those team there are separate pilots competing with each other. There are chess, archery, and bowling 'teams', where team members are basically independent, and the 'team achievements' are determined by the addition of every team member's independent contribution—and all 'team members' perform similar actions, so they are basically interchangeable. In this kind of team, one member's actions have little or no effect on the other members' performance. Sales teams or—unfortunately—management teams are usually an example of this kind of configuration: every salesperson cares only about his own sales, every manager cares only about her own department.

On the other hand, using a rugby or football[13] metaphor, some teams are interdependent—the results depend not only on the everyone doing their own stuff, but on effective collaboration of all team members. This is the case for medical teams—nurses, doctors, specialists, and surgeons treating the same patient—or software teams, where analysts, user experience experts, designers, programmers, architects, and testers must collaborate on the same product.

[10] Schwaber K, Beedle M (2001) *Agile Software Development with Scrum*. Prentice Hall.

[11] Collins J (2001) *Good to great: why some companies make the leap…And others don't*. HarperBusiness.

[12] Womack J, Jones DT, Roos D (1990) *The Machine that Changed the World*. Free press.

[13] What we Europeans call football, of course, not the rugby-with-armor version of the American folks… ☺

What about complexity-facing, problem-solving teams? Of course, the independent strategy does not work very well. They relay on some external 'central intelligence' to provide them with guidance and tell them what to work on. Basically, when companies create an independent team and hope that it will improve their problem-solving capacity, they are making a statement of their amazing dysfunction and their ignorance of why and how teams are important to deal with complexity.

Some basic rules to assess if your teams match the usual structure of problem-solving teams (like, for example, the Agile teams, but not just them) are:

– **Small**: several studies have been conducted to determine the optimal size of teams, and every expert has his own favorite size. Problem-solving teams size depend on their circumstances (technology, market, product, company size) but range from 4 to 10 or 12 people. Above that size, the team tends to collapse under the complexity of communication and knowledge sharing. Below four, there's usually not enough critical mass, cross-functionality, creativity, or different perspectives. 'Team of two' is a managerial smell I find too often, and it never indicates something good: it is usually a sign of calling something a 'team' when it's not, or managers micromanaging their people and deciding who gets to work on what.
– **Cross-Functional**: functional teams—those where all team members have the same skills and work on the same tasks—will inevitably suboptimize with any improvement they make to their own field. Cross-functional teams are sometimes misunderstood as 'everyone knows how to do everything'—which is nonsense. The correct definition is a team where all the necessary skills and knowledge are present in order to develop an increment of functionality from concept to cash. The idea would be to take engineers, designers, workers, and customer representatives all together and let them drive the development process from start to finish.
– **Self-organizing**: there is little use in forming a problem-solving team and then introducing a 'leader', 'boss' or 'manager' (you choose the flavor) who will then be identifying problems, diagnosing them, deciding on the most optimal solution, and telling everyone what to do. Even if you take all your managers and call them leaders, they will be just wolves on sheep's clothing.

– **Feature-driven**: another way to describe it would be that they are product- or service-driven instead of technology-driven. The goal is that any team in the company can work on any feature or product, thus giving us maximum flexibility. This goal might not be 100 % achieved, but a Kaizen company will constantly change and adapt toward it, and will try to be as close to this desired state as possible.
– **Collaborating**: rather than *coordinating*—everyone working on different stuff so they don't overlap—or *cooperating*—working on the same stuff, but one at a time so they don't overlap—problem-solving teams will *collaborate*. This means that they engage the problem together and work on it at the same time. They exchange perspectives and share knowledge. Maybe not all team members might be collaborating at the same thing, at the same time, but still collaboration is the prevalent team mode: two people might be working together on a part of the problem, three might be working and some other part and the five of them are synchronizing, debating, and rearranging frequently, usually several times every day.

So many times I'm hired by a company that has too-small, functional, managed, technology-driven coordinated teams, and they ask me 'What are we doing wrong? How can we improve?' When you tell them to rearrange the team structure and dynamics they usually say 'Oh, no, we cannot do that... You don't understand'. Of course, doing the same things over and over leads them to the same kinds of results—and problems. Time to try something new?

Functional, task-driven, micromanaged teams with independent members are a form of Taylorism that is directly against the core of Agile development. This kind of team configuration might be good for algorithmic, effort-based, mass-production, repetitive operations. On the other hand, if your main concern is problem solving, if you are facing a complex environment, if every day you are performing tasks differently, then the Agile team configuration will provide better results.

Team Identity

Good, jelled teams grow a team identity. This identity is much more than the sum of all team members' identities, or the minimum common identity of them all—it becomes a 'third entity' that is independent of team members and shapes their behaviors when they are in team context. Probably, the best story about team identity that I've ever read is that of the Black Team on *Peopleware*[14]:

> *The most surprising thing about the Black Team was not how good it was at the beginning, but how much it improved during the next year. Some magic was happening: The team was forming a personality of its own. This personality was being shaped by an adversary philosophy of testing that evolved among group members, a philosophy that they had to want and expect to find defects. They were not rooting for the developers at all, quite the opposite. They were delighting in submitting the program (and the programmer) to a sequence that was not just a test, but an ordeal. [. . .]*
>
> *To enhance the growing image of nastiness, team members began to dress in black (hence the name Black Team). They took to cackling horribly whenever a program failed. Some of the members grew long mustaches that they could twirl in Simon Legree fashion. They'd get together and work out ever more awful testing ploys. Programmers began to mutter about the diseased minds on the Black Team.*
>
> *Needless to say, the company was delighted. Every defect the team found was one that the customers wouldn't find. The team was a success. It succeeded as a test group, but more importantly for our purposes here, it succeeded as a social unit. People on the team got such a kick out of what they were doing that colleagues outside the team were positively jealous. The black outfits and the silly exaggerated behavior were part of the fun, but there was something much more fundamental going on. The chemistry within the group had become an end in itself.*
>
> *Over time, members of the team moved on one at a time to other things. Since the team function was important to the company, departing members were replaced immediately. The continued until finally there wasn't a single member left of the original group. But there was still a Black Team. The team survived the loss of all its original staff, and it emerged with its energy and its personality intact.*
>
> – Tom de Marco, Timothy Lister, *Peopleware: Productive Projects and Teams*

[14] De Marco T, Lister T (1987) *Peopleware: Productive Projects and Teams*. Addison-Wesley Professional.

There are several factors that contribute to form a team identity, but team identity can be easily made equivalent to a microculture inside the company's culture, and hence you can use the same cultural model we proposed in the first chapters of this book: noble cause or purpose, shared values, and cultural artifacts, both physical and behavioral.

In the Black Team story, those factors can be identified as:

- **Noble cause**: make the software better. Find defects nobody but us can find. Be a high-performing team.
- **Values**: perfection, quality, technical excellence, collaboration, uncompromising nastiness when it comes to testing software.
- **Physical artifacts**: black outfits, moustaches (!!)
- **Behavioral artifacts**: cackle when defects are found, twirl moustaches, exaggerate behaviors, fun, designing awful testing ploys.

Once you understand those, there are several behaviors that you can check against the team identity or micro-cultural model to see if they are acceptable or not—would a Black Team member let a defect pass because it will compromise deadlines or because it is very difficult to find and fix? The team identity can work as a personal compass in order to make decisions in a better way than company procedures and norms can provide.

Team microculture should not be a problem unless it creates purposes, values, or artifacts that go against the common company culture. I don't see that happen too often—it usually happens the other way round: the company culture kills team identity. Kaizen agents must make sure that the efforts to foster a corporate Kaizen culture help teams to grow more Kaizen-oriented identities.

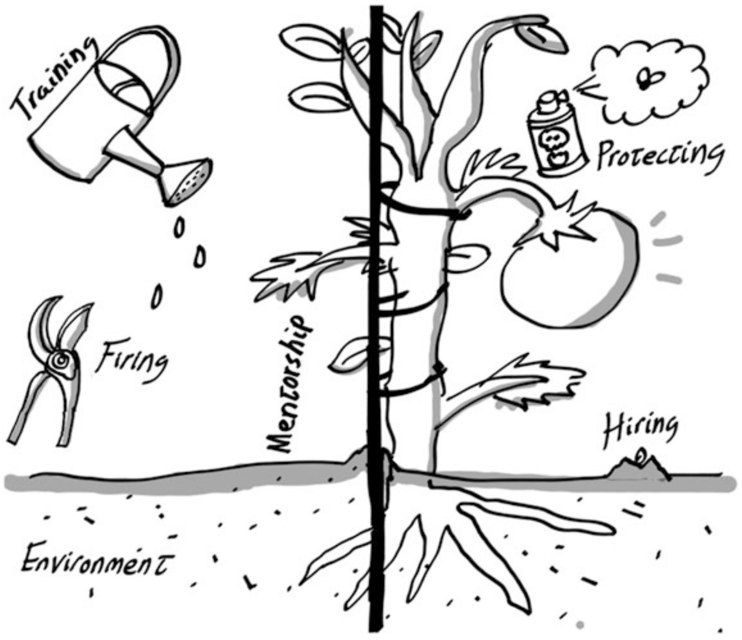
Environment is important to foster team identity, starting with the team's board. I've always defended the idea that the team's board is a totem—an emblem of the team's identity. A good board should radiate purpose and values—not just post-it notes and burndown diagrams. Team space can also influence team's identity: a team that has access to whiteboards, break rooms or discussion space, collaboration tools, and communication channels will grow a different identity than one where all team members are confined in cubicles.

Behavioral and 'mind artifacts'—like shared beliefs, mindsets, stories, and preferences—are also important when defining team identity. Defining a common way of working, having some rules on how to collaborate and manage information, or even some guidelines on how to communicate and discuss can improve the teams' pace to forming a common identity.

From Zeros to Heroes

Some team-related literature uses the metaphor of 'Team Building' when referring to the process of configuring and launching teams. For me, it gives the false sensation that teams can be built using some blueprints or procedure, as if we were playing with Lego bricks—which is absolutely not the case.

I prefer to use the gardener's metaphor.[15] Gardeners might select the seeds (hiring); prepare the soil (environment); nurture the plant (training, motivation); guide it while growing (mentorship, coaching); protect it from bugs (support, remove impediments); or prune the bad branches (firing or changing members). But even then, you are not guaranteed that you will get an award-winning tomato. Sometimes the magic just does not happen—time to seed again. That's why I prefer to talk about Team Growing rather than Team Building.

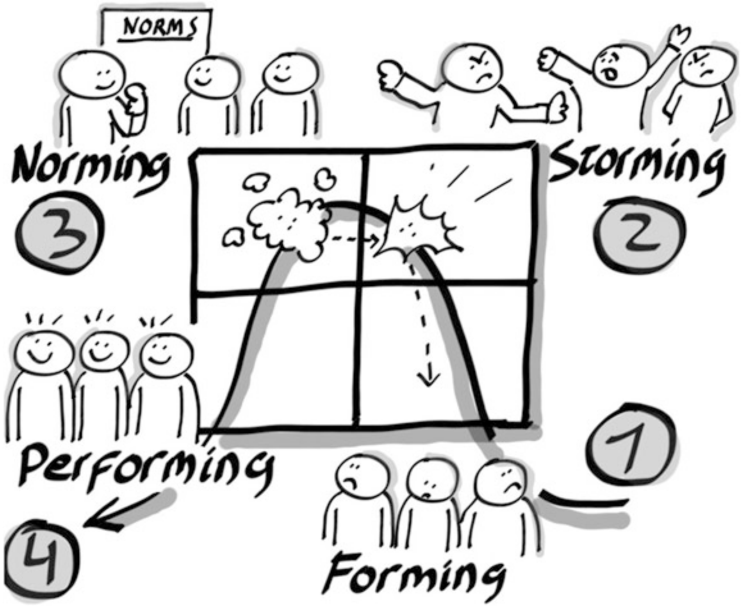


[15] Several people in the Agile community, including Henrick Kniberg and Jurgen Appelo, use similar garden metaphors to describe the process of 'managing' Agile teams and dealing with complexity. I couldn't find a specific origin, just would like to point out that I'm not the first one relying on it.

Besides the gardening model, another useful framework for team growing is Tuckman's[16] four-stage model for team development:

- **Forming**: team is created or assembled. People will try to be nice and avoid conflict, thus creating an 'artificial harmony'. Typically they will find ways of working independently on different things so they don't need to discuss or run into conflict. Daily meetings are more like status reports where everyone talks about their stuff, which really doesn't interest the rest of the team members. No conflicts or big team impediments are identified at Retrospectives—the team focuses on small unimportant issues in order not to run into big, hairy conflicts. There is no shared responsibility or ownership of the improvement process: everyone just cares about his or her personal performance. No real Agile team—or even team—is present at this stage.
- **Storming**: as team members get to know each other and are forced to collaborate (or even just cooperate), different perspectives and opinions are discussed. Team members must find ways of collaborating effectively without wasting a lot of time trying to impose their criteria. Facilitators might help the team start considering the big impediments and conflicts and try to give solutions to those. If the team does not find a way of dealing with conflict, or conflict is too big and violent, they will revert to stage one—artificial harmony. Team motivation might fall during this stage. Good facilitation is key for success, and lots of patience might be needed. Usually, facilitators might be more directive during this stage—like in 'we are doing daily meetings, whether you like them or not'.
- **Norming:** the team starts working on common goals—there is true shared responsibility and accountability. Team members find ways to collaborate constructively and share knowledge. Team identity starts to grow in this stage—ways of working, shared values, common goals, and team-enforced behaviors.
- **Performing:** the team grows in skills, capabilities, and performance. They work smoothly as an elite unit and love to be challenged. Motivation and team cohesion are very high. Facilitation in this stage might not be needed at all or should be less directive and more supportive. Some teams, though, just never reach this state of 'hyper-performance' due to impediments, bad processes, lack of support, or other environmental factors.

[16] Tuckman BW (1965) Development sequence in small groups. *Psych Bull* 63 (6):384–399.



It is interesting to see how this model is still very relevant for most team coaches, experts, trainers, and authors, even 50 years after it was first proposed. The model assumes a couple of principles that are not always well understood by Agile-wannabe companies (or anyone trying to grow problem-solving teams):

- **Agile teams need stability**: you can't manage the proverbial 'pool of resources', moving people around from project to project on a daily, weekly, or monthly basis, and call that 'a team'. Tuckman's model shows that teams need some time before they reach a performing state—if they ever reach it. Sensibly altering their composition reverts them to the 'forming' stage.
- **Agile teams need facilitation**: or, at least, they can benefit enormously from it and it can dramatically improve their chances to become a hyper-productive team.
- **Agile teams must collaborate**: it is not enough that they are coordinated or they cooperate. Agile team members must engage in common goals, common projects, and common tasks together.
- **Agile teams need to deal with conflict**: conflict is not something to solve or be avoided—it is an opportunity to reach new consensus, try new things, learn and grow further into the norming stage.

Self-Organization

The one thing an Agile team must be, beyond any discussion, is self-organized. You can discuss size, process, structure, feature-driven versus function-driven, or even cross-functionality, but there's little choice to discuss self-organization if you want to play under the Agile rule set.

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

Agile Manifesto – Principles

I discussed self-organization extensively in my first book. For the sake of team improvement, I'll remark that good, productive self-organization depends on three factors:

– **Alignment**: clear set of goals that benefit the company and that the team understands and agrees to.

– **Boundaries and constraints**: small set of rules that allows plenty of freedom on how to pursue the defined goals.

– **Progressive delegation**: teams shall be offered as much freedom as their maturity level asks for—not more, not less. Then, progressively, a bit more freedom should be granted and facilitated.

Smart goals and small sets of boundaries, constraints, and rules create a good context that the team can use to understand what is expected from them. They can then decide how to perform and deliver in order to maximize the expectations met.

Regarding delegation, if you give the team too much freedom they will probably not know what is expected from them or how to meet those expectations. On the other hand, if you don't give them enough freedom you will kill motivation, self-organization, and learning. The trick here is to be always 'one step further' from the team comfort zone, making them learn how to manage more complex decisions every time. In order to do this, managers and facilitators can experiment at the beginning, telling the team what to do and explaining the reasons of this. Later, they can ask the team for advice and see if their insights on what to do match their criteria. Then, they can start making decisions together so at the end a full delegation of responsibility (and accountability) can be performed. Of course, different decision areas can be delegated at different levels. A good way of understanding this and learning more is using Jurgen Appelo's Management 3.0 Delegation Boards.[17]

This process helps the team grow, improve, and obtain the most out of self-organization—just telling them to self-organize and hoping for hyper-productivity won't work. Think of this process as somehow similar to raising your kids: you start by holding their hands, later you walk with them, and some day you'll be able to send them to the convenience store alone. But if you just get them out of the cradle and give them a Visa card and the keys to the house… Well, don't expect good things coming that way.

Problems in Agile Teams

As with every Kaizen aspect, several impediments can prevent an Agile team from reaching the desired high-performance state:

– **Absence of trust**: the team is not open to talking about their problems.
– **No conflict**: the team does not want to enter into constructive discussions.
– **Too much conflict**: the team is always discussing, but does not know how to reach consensus.
– **Wrong characters**: sometimes the mix of characters is just not right. I haven't yet found a 'right' guideline or framework for choosing adequate characters, although most people recommend a balanced mix.
– **Individualistic approach**: there is no real collaboration; people just care about personal goals and do not help each other. Personal goals, performance reviews, bonuses, and rewards might trigger or encourage this behavior.
– **Specialization**: some people in the team may only engage in one type of task, or the whole team may have the same skills so there is no critical mass of skills and perspectives.
– **No knowledge sharing**: this can also happen as a form of specialization: some people will hold their knowledge for status and power.
– **Lack of skills**: if the team does not know how to perform the tasks they are assigned, they will be frustrated and will not advance.

[17] http://www.management30.com/workout/delegation-boards/

- **No time for the team**: which, again, can lead to no training, no support, no improvement, no discussions, and no learning.
- **No clear purpose**: if the team can't understand their role and goals or cannot relate to those, they'll fail to grow a team identity or will create a purpose on their own that might not be aligned with the company's expectations and goals.
- **No results**: teams need to see advancement and results in order to be motivated.
- **Too much management**: managers in any form making the decisions and assigning tasks to team members and coordinating them, so no self-organization happens.
- **No leadership**: meaning no personal or distributed leadership. Nobody takes action, and everyone just hangs around waiting for orders. There's no clear role model for team members.
- **Wrong culture**: people around the team discourage team behavior. There is no real teamwork they can imitate.
- **Bad environment**: lots of interruptions, noise, priority changes, power abuse, wrong tools, blaming, or fear.
- **Absence of facilitation**: team falls into bad behaviors and there is no help, training, or support to correct these—sometimes because nobody in the organization has the facilitation skills and managers fail to see the business case in hiring facilitators.

Seeing how many things can go wrong, you might start to understand why having high-performance teams is both so difficult and so powerful in terms of competitive advantage.

On the other hand, bad team behaviors can also kill the chances that a team will ever grow beyond the Storming stage. Some of the most usual bad behaviors and dynamics you can watch for are:

- **Hijacker**: takes most of the meeting time, answers all the questions, does not let other people talk, interrupts them, moves the discussion in his or her own interest; basically, tries to take control of the meeting.
- **Busy bee**: urges everyone to cut down the meeting so they can all go back to work. Wants to move on before conclusions are distilled. Sees discussion and collaboration as a waste of time. Does not see any value in meetings besides 'conformance to the Agile process'.

- **Ghost**: refuses to participate. Another usual case when someone does not see any value coming out of meetings. Would prefer to be doing something else—but is too shy or introverted to say so; or is so disengaged and demotivated that he sees the meeting as an opportunity to goof-off.
- **Dr. House**: arrogant and sarcastic. A form of passive-aggressive behavior. Discards other people's ideas, will even laugh at them or make mean-spirited personal attacks. If you call his or her attention, the answer would be in the line of 'hey, just kidding' or 'I was just asking—just asking'. Usually a sign of strong character and frustration—sometimes this person just does not know a better way of communicating.
- **Grandpa**: digresses and wanders away from the meeting topic. Tells stories, makes jokes... Another form of disengagement.
- **Laggard**: refuses everything with no real reasons. Sometimes just skeptical—will give some reasons, real or fictional, why he or she thinks it would never work. Never proposes alternatives or constructively discusses workarounds. Blocks decisions and will not support others' ideas even if they don't have better ones.

Facilitators should watch for these behaviors and work them at both the team and personal level. The team should take decisions on what to do if they find that someone is falling into some of these modes, and the facilitator should address repeat offenders in private conversations. Be careful not to scorn or preach to people in public: if they still don't feel comfortable being vulnerable in front of the team, they will feel attacked and will disengage even more from team dynamics.

Conflict

If everybody is thinking alike, then somebody isn't thinking.
– General George S. Patton

Conflict has become a bad word in corporate environments. Conflict is something to avoid, something to solve. Scores of people ask me at seminars 'How can I deal with conflict? What do I do if there's conflict around?' Nobody seems to be happy that conflict is showing up, and that's because, usually, when people talk about conflict they talk about a high-level conflict that has gone far away from control.

In her book *Coaching Agile Teams,*[18] Lyssa Adkins proposes a five-stage model of conflict intensity:

– **Level 1**: Problem to solve. Information sharing and collaboration. Language is open and fact-based.

– **Level 2**: Disagreement. Personal protection starts being important. Language is more careful.

– **Level 3**: Contest. Winning more important than solving. Language turns personal.

– **Level 4**: Crusade. Protect your group or idea. Language is ideological.

– **Level 5**: World War. Destroy the other. Little or no language exchanged.

Adkins uses this model to explain that, in order to engage in positive conflicts, levels above 'Problem to Solve' must be de-escalated, usually one by one. If some people are at the 'World War' stage, you must separate them in order to avoid further damage. Then, you can de-escalate them through level 4 with the use of diplomacy—acting as a messenger between the conflicting parties.

On level 3, you can start to turn the team to the facts and remind them that they are on the same boat. Appeal to their shared goals and values, to the team's identity. A successful approach is to ask each other to argue the other part's position—if you can't explain your opponent's point of view, you basically think he or she is stupid! Once you turn them back to the facts, you can start negotiating, facilitating, supporting, and discussing through level 2 all the way back to level 1—positive conflict.

If you are like most people, you don't like conflict. And the reason why is simple. You've never been trained to participate in meaningful conflict, so you likely think of conflict as scary, harmful, and hurtful. Conflict can be all three; done well, conflict can also help you accomplish your work mission and your personal vision. Conflict can help you serve customers and create successful products. Happy people accomplish their purpose for working. Why let a little professional courage keep you from achieving your goals and dreams? Make conflict your friend.

– Susan M. Heathfield, *Top 10 Ways to Be Happy at Work*

[18] Adkyns L (2010) *Coaching Agile Teams: A Companion for ScrumMasters, Agile Coaches, and Project Managers in Transition.* Addison-Wesley Professional.

There's yet another aspect of conflict that you need to understand, and it is again hard-wired in the human brain: the emotional or 'Amygdala' hijack. Coined by Daniel Goleman in his awesome book, *Emotional Intelligence*,[19] this term refers to an overwhelming emotional response to a situation that is perceived by our primitive brain as an aggression or a threat. In these cases, our reasoning capability is affected by the emotional state, and our brain can produce arguments that seem valid to us in that moment, but that would not pass a sanity check afterwards.

When emotions are present—fear, anger, frustration, jealousy—the facilitators must take care to de-escalate the conflict from the emotional state before trying to argue reasonably with the conflicting parties. Any attempted reasoning during an emotional hijack will go through so much emotional noise that it possibly won't work—save your arguments for later.

As with conflict levels 5, 4, and 3, emotional hijack must be addressed by making sure the parts do not harm each other, acting as a diplomat, and pointing them to their common identity, goals, values, and concerns. Above all, remind them that this is not the end of the world, nor is it a life-or-death choice—no matter what it looks like right at the moment, it never is. They will be forced to go into similar situations over and over during their career, and they need to understand that no personal aspects are being discussed when considering other team members' opinions.

Consensus

To be able to get the most out of positive conflicts, teams must learn how to reach consensus. Consensus is generally mistaken for 'unanimity' or 'vote of the majority', which is not.

Unanimity might be impossible to achieve in every case: people have different opinions, and waiting until all are convinced could make the team rather inefficient. On the other hand, majority usually means that some people win and a few people lose. Imposing criteria with the vote of the majority, even when some people on the team feel really uncomfortable about it, can lead to team dysfunction, disengagement of the unconvinced people, and lack of support from their side—even sabotage!

[19] Goleman D (2005) *Emotional Intelligence: Why It Can Matter More Than IQ*. Bantam Books.

Consensus means that the team agrees on trying some given option even if some people are not fully convinced. The idea is that, even if they don't fully agree or are skeptical about it, people reach consensus if they will support and back up this option wholeheartedly to see if it works or not.

To reach consensus, facilitators must make sure that everyone understands the others' points of view and feels like the other people on the team understand their own point of view. Then, when the decision is made, facilitators must make sure that no one undermines this decision and everyone does their best to support it—at least, until we discover if it works or not, or a new, better option is discovered.

A good way of testing for consensus is to ask for fast 'thumbs up' style voting on decisions:

– Thumbs up: I fully agree with this decision.
– Side thumbs: I don't fully agree, but I can live with it and would support it.
– Thumbs down: I can't live with it and won't support it.

If at any given moment there are no thumbs down, the team can stop arguing and go for action instead—thus making discussions fact-based and more efficient.

In case of any thumbs down, several questions can be asked in order to facilitate the decision:

– What's the part of this option that you can't live with?
– Is it an emotional issue or is it a fact-based argument?
– How is this option personally affecting you?
– Why do you think other people might prefer this option?
– Is there any way we could modify this option so you could live with it, even if you don't fully agree with or like it?
– What other options can you provide? How are they different from this one?
– Can you support a time-boxed test of this option?

Team Facilitation

As we have explained, there is a path that a group of people must follow in order to become the kind of complexity-facing, problem-solving team we are looking for. It is a learning path. Odds are that the several impediments and difficulties that the team will face during the 'Storming' and 'Norming' stages will send them into failure mode. In order to reduce these chances, facilitation might be needed.

The role of the facilitator is nonintrusive. He or she does not actually participate in discussions or engage in content—they just care about the consensus process. If you also consider the fact that facilitation skills are not easy to find, it makes it a bad idea to 'just rotate' the team facilitator role amongst team members: they will usually lack the skills and they will be too involved in the topics discussed.

Companies can very easily calculate the cost of one facilitator per every one or two teams, but they fail to assess the benefits of improving team performance—this is another reason why good facilitation is so difficult to find on most companies, even Agile ones. Without proper facilitation, it's more likely that different team dynamics, good or bad, are not identified or are dismissed in favor of 'more urgent' discussions on features, deadlines, estimates, progress, or other short-term project-related stuff. On the other hand, it is more difficult for the team to deal with conflict, make decisions, and reach constructive consensus, thus making them less responsible or accountable.

Team facilitators, ScrumMasters, or Agile coaches are not actually responsible for team dynamics—the team itself is responsible—but they should look after them. Some of the aspects team facilitators should care about include:

- Proper communication and respect amongst team members.
- Knowledge-sharing strategies and practices.
- Constructive and efficient decision-making processes.
- Team identity and culture—care for team purpose, respect for team values, and development of team cultural artifacts.
- Team learning and improvement.
- Whole team participation.

- Observance of ground rules and team agreements.
- Team performance.
- Understanding the team goals and stakeholders' expectations.
- Individual improvement and 'career paths'.
- Individual and team motivation, momentum, and engagement.
- Providing feedback and recognition on behavior and improvement.
- Showing the team their own results to encourage further improvement.

Team facilitators should be detached from current project outcomes or deliveries—they care about 'next year's team'. They are concerned on the long-term and should not be compromised by short-term boundaries and constraints, although they shall of course be aware of them. Some people worry that, if facilitators are not concerned with short-term goals, these goals won't be met. Please, be aware that a lot of other people—product owners, managers, sales people and even customers—are concerned with these short-term goals and will push for them. In fact, we don't need another short-term related role. What we need is at least *one* person in every team caring for the long-term improvement—Kaizen!

A usual over-simplification of the team facilitator's role is to say that he or she is responsible for removing impediments. At the beginning, this might be close to reality, at least with the most evident environmental impediments that prevent the facilitator from performing his or her role. Over time, the facilitator will switch his or her focus to making the team capable of identifying and removing impediments by themselves.

Team facilitators shall also act on an individual basis as personal coaches. My own rule of thumb is that team facilitators should be able to, for every 2-week iteration, have a one-on-one 30-min talk with each team member they are helping—*at the least*. During these talks, team facilitators can discuss personal goals, individual learning and improvement, private feedback not to be discussed openly, personal relations with the team, or even ask for help in changing team dynamics.

Providing enough facilitation tools and explaining their use is beyond the scope of this book, but some resources for further learning are proposed at the end of it. As a short list, some of the most usual facilitation tools are:

- **Brainstorming**: the team dedicates some given time to generate as many insights as possible with no discussion or comments on generated ideas.

Later, the team reviews them, trying to maintain a positive, constructive point of view, combining them and altering them in search for new ideas. Time boxing and no arguing are two very important rules in a well-facilitated brainstorming.

- **Clustering**: can be used as a part of brainstorming. Similar proposed ideas are clustered, and the cluster is then identified under a common theme.
- **Dot voting**: used for prioritization. Everyone in the team is given number of adhesive dots—three to six, for example—and then they will be able to distribute them through the available options. For example, they could give three dots to the idea they like the most, then two dots, and one dot to the following ones in priority order. After everyone has voted, all ideas are prioritized according to the number of dots they were able to collect.
- **Fast voting**: as proposed in the consensus section, a way of doing this would be to use thumb-voting (thumbs up: I support this idea; side thumb: I can live with this idea; thumbs down: I cannot live with this idea). Other ways of fast voting include finger-numbering (from 1 to 5, how convinced you are, or what priority would you give to this idea) or planning poker (everyone plays a card with a number at the same time, then discussing the number they proposed).
- **Silent writing**: everyone takes some minutes to reflect and write insights, then all insights are discussed. This technique ensures that nobody hijacks the idea generation stage and that everyone gets to participate and give feedback, even in the presence of shy or introverted people.
- **Group mind-maps**: mind-mapping is a way of graphically outlining information and generating ideas. A focus topic is written in the center and several related topics are radiated in sub-branches on a tree-like structure. Color, images, symbols, and other codes are enforced to make the process more visual, engaging, and creative.

Summary

All human beings need discipline and commitment to achieve their goals. Our brains tends to disregard long-term effort and favor short-term rewards, so we must train our brains in sacrifice and self control in order to reconfigure them and create Kaizen-oriented minds.

To deal with complexity and solve problems, teams offer better results than individualistic approaches. Teams are the building blocks of Agile companies. To take advantage of the team strategy, certain team configurations and structures should be observed: cross-functional, small, feature-driven, self-organizing teams. Not every team is an Agile team, and even worse, not every group of people is really a team, no matter what you call them.

Real teams grow a team identity through a process of conflict, discussion, and positive agreement. As a team matures, the delegation process can be pushed forward so the team is more empowered and can grow in their self-organization. Facilitation dramatically enhances the chances to overcome all of the failure modes, bad behaviors, and impediments that can prevent the team from reaching the performing state.

Things to Try

– How many of your people are engaged in hobbies, sports, or activities that represent an improvement process through disciplined, deliberate practice? Ask them about any long-term goal they would like to achieve—run a marathon, learn a new language, teach themselves to play an instrument... Why not help them pursue those goals? Disciplined, motivated people make better Agile team members!

– Introduce some exercise routine in the team's work day. It might be a small walk around the park during daily meeting, some morning stretching, or yoga sessions before lunch time. Light exercise is a great motivator, and teams who exercise together create a team identity more quickly.

– Assess your team structure following the guidelines offered in this chapter. Treat any divergence from the proposed model as a major alarm and try to analyze the reasons for such a singularity—they might be valid, but don't be indulgent with yourself. Discuss this singularity with people outside your company to see if your prevalent paradigm might be flawed.

– Enhance team identity in any possible way. Ask teams to choose their own name. Let them design their own team board. Suggest they create their own logo, celebrate 'Team Day', enforce their own ceremonies, or let them customize their workspace.

– Assess your team's stage: forming, storming, norming, or performing? What is holding them in their current stage? Analyze the stage characteristics and define ways to move them to next stage.

- Review the list of good and bad team behaviors. Discuss it with the team. Try to remember situations where they fell into some of them—both good and bad. Decide what to do in the future if they fall into those bad behaviors. Add them to the team ground rules or Way of Working (see the team exercises in Part II of this book).
- Ask the team to describe how they think they are expected to behave. Compare that to a similar description provided by managers and other stakeholders. Discuss any differences and ask the team if they wholeheartedly agree on the desired behavioral state—if not, engage in conversations on how to change expectations. Analyze how to improve a team's behavior to match a commonly agreed-upon desired state.
- Make sure the team understands its goal and the 'size of their playground'—the rules, boundaries, and context that define their self-organization context.
- Use Delegation Boards (as described in this chapter) to discuss the different decision areas of your company: who gets to make the decisions, and what level of delegation and collaboration should be enforced in the process of making them.
- Ask the team to rate, from 0 to 10, how likely they are to run into any of the Agile team problems described in this chapter. Use this as a basis for root cause analysis and improvement plan design.
- Craft some cards with the 'bad team behaviors' and ask the team to use them when they see someone falling into one of them by handing that person the card.
- Ask the team to create 'conflict rules' to be used. They should provide a way to define the conflict level and what to do to de-escalate conflicts—some ideas are provided in this chapter. Include consensus rules on how to agree on actions and decisions, and how to deal with decision blocking.
- Review the facilitator's role in your environment—who is performing it? What are his or her duties? How can you measure the facilitator's performance and improvement over time? Use the list of things the facilitator should be looking after and ask the team to rate their facilitator against it.
- Introduce your teams to the facilitation tools described. Tell your facilitators to research the resources list at the end of this book.

5 Process Kaizen

Improving the Way You Do Your Stuff

Time kept passing by. Facilitators struggled with their teams and had to fight with managerial layers so they could use more time on Team Kaizen. After some months, the compound interest of that investment started to show up. Small wins at the beginning, some big wins at last. The most noticeable results were enthusiasm and conversations.

Conversations were everywhere. Sometimes people were just discussing issues at the coffee machine, in pairs. Sometimes people would pull a flip-chart by while having lunch and start discussing problems. Most of these flip-charts ended up hanging on a wall – everyone liked visual management. Teams had crafted amazing team boards where they loved to share all kind of information, from project performance charts to team party pictures.

Teams had been working on their identities, exploring their values and the behaviors they wanted to encourage or avoid. A common culture started to emerge in the form of shared values. Even management started to articulate their message around the team-generated values.

Probably teams were still not fully comfortable with conflict and discussion – at least, not as comfortable as Michael would have expected. But they were improving. After years of working at this company, finally some of the big hairy issues were coming to surface and were being questioned. The most repeated sentence was probably 'we have to change this'.

But there were not many solutions yet. It seemed that people were becoming better at spotting defects, problems, impediments, and improvement areas, but still were not able to craft viable plans to cope with them. Some managers had approached Michael in private conversations to share how preoccupied they were when the teams were asking them for changes *and improvements – and they had no clue on how to provide these. Everyone seemed to look at their boss for solutions – even managers.*

Finally, it happened. The CEO called Michael for a one-on-one on the situation. In a very managerial way, the CEO started with some compliments on how happy everyone was around the Kaizen initiative, but then he moved to the real target of the meeting: they had to show results. If the situation kept same for too long, people would lose their impetus. They needed results, and they needed them sooner than later.

Michael's efforts to explain the long-term nature of Kaizen had no effect on the CEO – again, he wanted results, and he wanted them now.

Michael walked away from the CEO's office silent and meditative. What was the problem? People seemed to be in the correct Kaizen state of mind, but still they lacked some spark, some magic.

That night, Michael couldn't sleep. He wandered randomly across the lines of his favorite Kaizen books in search of an answer. As usual, all made sense, but it was difficult to translate all those principles and values to specifications or to concrete tools. Then, the answer struck him: his teams were having the same problem. They had been growing principles, attitudes, and values. But now they lacked process. They lacked tools. They lacked practices.

Now Michael couldn't sleep, but because of the excitement. It was time to introduce a practical approach to Kaizen, starting at the production line.

The CEO wanted results? The CEO would have them...

Starting with Purpose

One of the biggest mistakes when it comes to Process Kaizen—improving your process, that is—is to rush on a blitzkrieg of waste reduction, lead-time improvement, inventory reduction, and productivity improvement.

What if your customer does not care about that?

Modern companies have come to worship the idea of cost reduction and productivity improvement, just for the sake of them. But that's not the way Toyota created their now-famous production system. They started with the purpose of the company. They began by understanding value from the perspective of the customer. Our modern managers believe that they know their customers even better than their own customers know themselves. They love to talk about how Steve Jobs knew much better than the market what the market needed, and they usually quote Henry Ford:

> *If I had asked people what they wanted, they would have said faster horses.*
>
> – Henry Ford

Of course, in order to maintain their own paradigms, they fail to mention Steve Job's failures with companies like Next, who built a fancy, powerful computer *nobody wanted*. Or how Henry Ford did not actually *invent* cars: he was already playing on a proven market for them. His customers, in fact, wanted cheaper, more reliable, more durable, more easily repairable cars, and did not want to wait a year or two for them.

We will dive deeper into product concerns in the next chapter, but for the sake of process improvement, let's start by stressing the importance of understanding value from the customer's perspective before making any attempt to improve. Some companies engage their Kaizen with the purpose of cutting costs and earning more money. This is wrong. Of course, in the long run, if you maximize value and eliminate waste, both from the customer perspective, one of the side effects is that you will *actually* lower costs and earn more money—it is just not the main purpose!

Selling more, growing bigger... this kind of purpose leads to both bad corporate cultures and bad moves that could eventually kill your company. In the book *Gemba Walks*,[1] Jim Womack mentions how General Motors was able to improve during the 1920s when Alfred Sloan proposed a product portfolio that made more sense from the customer perspective, and how the 'grow bigger, sell more, make more profit' strategy of the 1980s and the 1990s eventually killed the company. He also mentions how, sadly, Toyota might be going that way right now.

Second to understanding your purpose, in the sense of how you provide value to your customers, your next step is to set an improvement goal. If you set a goal of reducing time to market, you might want to opt for a development process that introduces more costs and hence delivers a more expensive product, as well as the other way round. You might also want to create some balance between different goals—like cost and performance. Shooting for the

[1] Womack J (2011) *Gemba Walks*. Lean Enterprise Institute.

moon—bigger, better, cheaper, faster, fancier—might dramatically decrease your chances of success, so it is usually a good idea to focus your performance efforts on the most valuable aspects, the ones that deliver more value to your customers. If you don't set any clear goals, the improvement efforts might be clueless, and your people might be improving parts of the process that, from the customer's perspective, didn't actually need improvement that badly.

Once you understand value from the customer's perspective and have set proper improvement goals, your Process Kaizen (improving your process) should rely on a system that constantly brings issues up. Kaizen events will undoubtedly help to raise issues, but only from time to time. You must shorten those feedback cycles, with preference for shorter timescales. Plus, you should also encourage the development of feedback tools that work constantly. These feedback tools might be both push-based or pull-based. An example of what I consider pull-based devices is a suggestion box or an informal conversation with your people: you ask them (pull) about what's going on. On the other hand, push-based devices would be those where information is constantly being displayed so you can spot improvement areas—like all the information radiators and visual management tools.

Defining Your Process: The Use of Standards

Kanban, a well-known framework for process improvement, has a very short list of rules:

1. Start where you are. Keep doing things exactly as you've always been doing them.
2. Identify and visualize the Value Stream. Make all workloads visible.
3. Limit the work in progress (WIP) to the capacity of the system.
4. Make policies explicit.
5. Help the system flow and improve.

I usually make the joke that people love the first rule—keep doing things exactly the same—and that, in fact, people are usually very good at it. But when it comes to the other rules…

The interesting thing about the first rule is that Kaizen starts by knowing exactly what you are *really* doing. That's why Lean (and Kanban, as a Lean framework) encourages the definition and observance of standards. But western companies usually fall into misunderstanding when it comes to the use of standards.

First, there is often a conflict between what the standards say and what people are *actually* doing. This usually happens because the standard was created by an engineer, manager, or consultant who wrote what he or she believe to be the best way of doing things—and occasionally he or she is right. But then, the people on the production line are disconnected from the standard definition, and if they are told to follow the standard they usually don't understand why they should do so. Sometimes they will figure out different ways to perform their tasks that might be more convenient to them, but might also be counter to the improved performance of the whole line or some other goals of which the production workers are not aware.

The whole scheme is basically old-school Taylorism—managers and engineers thinking, workers wielding the screwdrivers. It's the same kind of company that talks about documenting best practices—which are usually 'past practices', as the company's situation is always changing, and what was good a year ago, for a given client, in a given market, with a certain technology and set of tools, might not be the best practice for the current situation.

Lean and Agile teach us that Taylorism might be a good way of mass-producing the same product over and over, but if your company is facing complex situations—the markets and the demand are changing, technology is evolving, new products are arising faster and faster—Taylorism will not give you the best results. You need to mobilize every ounce of talent in your company and form problem-solving teams.

Another well-known problem with the typical western-style implementation of standards is that they are written once—usually by external consultants, as explained before—and they are never reviewed again. Maybe people are following the standards—according to my experience, they usually aren't—but there is no assessment of this fact unless a periodical 'Quality' certification review takes place.

By the way, the idea that quality can be defined as 'conformance to standards' never ceases to amaze me—what if the standards defines a poor way of developing products and providing customer service?

Standards are meant to stabilize the process before we attempt any improvement. There is no way to improve if everyone is performing work in a different way, or if we tell them to work in some given manner and everyone just does something different. People must get used to working in a similar, standardized way. This does not mean perfect work definition by standards, especially in knowledge environments where we never create the same product twice. But introducing shared guidelines, procedures, or ways of working helps to stabilize the process before we attempt to improve it. As an example, Agile software teams are introducing common frameworks—as Scrum, Kanban, or XP—or shared tools and coding strategies—test-driven development, continuous integration, code standards, and frameworks.

Standards are meant to change—frequently. A change in the standard means improvement, or at least that the team is trying to improve. If standards never change, that means that we are doing things the same way over and over—how can we expect different results then?

A bad system will beat a good person every time.
– W. Edwards Deming, *Total Quality Management*

Bad process does indeed produce 'bad people'. Over the years, scores of managers have asked me what to do with some disgruntled, disengaged, or demotivated employee. I often ask them if this employee was like that when they hired him, and the answer is always 'nope'. Still, these managers only focus on this 'bad person', as if it was both his fault and the only thing to fix in the system, instead of seeing it as a symptom of something way deeper in the roots of the corporate culture or the process. Ultimately, they ignore this person or they fire him. For me, every single time someone gets fired from the company, it should trigger a serious 'stop the line' to see where we failed and how to fix that. Maybe it was a bad hire—fix that. Maybe there are so many demotivational patterns present—fix that. Probably we failed to see the symptoms early and act accordingly—fix that. Maybe we did not support, encourage, train, or give different goals to the fired person—fix that!

Back to standards, someone could argue that, if they change so quickly, what's the point of writing them down and making them visible? The idea here is to make sure that people are behaving in a given way before we try to improve. Imagine that we have four people working: two of them are following the standard and two of them are not—and you don't know it. You introduce a new kind of material in the production process. The new material helps the two people following the standard to work faster, but it makes the people who are not following the standard slower. Overall, the team speed remains same. How would you know the effect of this new material on the team's work? How would you know if it was a good idea or not?

Standards, hence, are a baseline for improvement. A commonly agreed state is defined, and everyone remains in that state—honoring the standard—while looking for new ways of improving it. Teams must self-cultivate the needed discipline through peer review and even peer pressure, while managers and facilitators must help the team to improve and call their attention when there is nonconformity to the standard.

Nonconformity to the standard may only mean two things: an impediment or an improvement. If the team is not following the standard because they can't, they are being pressed to bypass the standard, or they just don't feel like doing things right, that's an impediment. On the other hand, if the team does not follow the standard because they found a better way of doing things, that's and improvement to the standard. The standard should be updated visibly, so other teams can benefit from the improvement idea.

To improve the chances that other teams learn from any team's improvement, standards should be made visible. Another reason for constantly visible standards is that people outside the team can check if they are following the standards. Of course, in a Kaizen culture everyone in the company should be empowered enough to call to other people's attention if they are not following their standard.

There are company standards, and then there are team standards. There should not be any problem as long as these sets are not contradictory. For instance, a company might ask every product to be delivered with user documentation—that is a company standard and every team should care about that—but a given team might be using some special technology, developing a specific product, or might even have a special cross-functional configuration and they can develop their own standards for that. Kanban boards, as a way of making policies and standards explicit, are essentially team-specific and therefore should not be standardized and normalized for all teams—this could be a huge mistake.

Standards should be seen more as ground rules or the team's way of working instead of as company processes, rules, and procedures. The idea that the company is enforcing standards leads to Tayloristic environments. On the other hand, if the company defines the standards, workers may feel that the company should be improving them—bye, bye ownership!

Agile/Lean companies must create shared definitions and improvement of standards—what some people call comanagement of the production line. A good way of defining processes and visualizing them, as proposed at the beginning of this chapter, is the use of Kanban or Scrum boards. Short descriptions of Kanban and Scrum are found in my first book, *Agile Management*, but full descriptions of Lean and Kanban are far beyond the scope of this book.

Kanban boards can be enhanced with noniteration-related information that can also serve to visualize the team's process—their standard. It can include team structure, duties, periodic meetings, WIP limits, what to do in the case of urgent tasks, service level agreements, team capacity, what to do in the case of dependencies, code standards to be observed, rules for interacting with the knowledge systems and team work tools, and the definition of 'done'.

Besides helping everyone work in a similar way, making current best practices emerge, and creating structure improvement around a common process, standards are very useful to train new employees. In order to get the most out of this training, a complete 'improvement backlog' can be maintained describing things that were done in the past and why they were changed.

Go and See

I reiterate the need to have visible standards because visual management is an important practice in Lean implementations. Visual charts, boards, standards, and reports make concrete the values of transparency, trust, ownership, empowerment, collaboration, communication, courage, and probably some more.

No matter how much you insist on the benefits of visual boards, there will be always someone asking if there is no electronic tool they could use to substitute the visual tools—from spreadsheets to really expensive and complex state-of-the-art software or ad hoc developments. These persons will rapidly find reasons why you can't do visual management, it 'won't be efficient', 'there will be needs not covered', or 'it would just better if...'

Lean experts have repeatedly stressed the importance of relying on manual, 'analog' tools—i.e., pencil and paper—instead of digital forms of reporting. This has been repeated for value stream maps, Kanban boards, A3 problem-solving forms, Root Cause Analysis exercises[2]... *All* Agile experts agree[3] that physical team boards are much better to any alternative—and

[2] Several of these tools will be explained later in this chapter.

[3] And if you know enough Agile experts, you'd know how amazing this is.

those alternatives should only be considered as an add-on to the physical board, never a real substitute. There are several reasons for this:

- **Real visibility**: an electronic tool is only visible when you access it. Most of the time, people will only access it when they need some information, so there is no real 'Radiation' of information—automatically exposing people's actions it even if they don't want them to be. You could, of course, invest thousands of dollars in big flat screens—if you don't do something similar, you don't really have visual management, and there are still the rest of the reasons in this list.
- **Peer pressure**: in the absence of real visibility, there will be no peer pressure on the team. You only know the real power of making your commitments public when you have worked updating a burndown chart or any other visual progress update tool on a daily basis. If your iteration status is hiding in some dark corner of an obscure tool, this effect is not likely to take place.
- **Adaptability**: it is very difficult and time-consuming—if it's possible at all—to adapt these tools to the reality of every team. Some teams will need board space for maintenance, some others might need different lanes for service level agreements. Forcing all of them to a common 'standard' in order to use an electronic tool is really looking at the world upside down: the tool should support the standards, and not the other way around!
- **Real Collaboration**: something I usually notice when companies are using electronic tools is that, in team meetings, there is one person using the keyboard while the rest of the team looks at the screen. This artifact creates a cultural message that you can guess on your own, and no, it's not about collaboration. On the other hand, I also see that, during iterations, team members will open the tool, update *their* task information, and then close the tool again—without taking a look at the other members' work. Again, you could spend thousands of dollars on tactile, electronic tools that let your teams move electronic sticky notes around, but when it comes to creating new charts, lists, posting information, or just writing things on-the-go, nothing beats the simplicity of markers and sticky notes.
- **Sense of ownership**: no electronic tool is fully 'owned' by a team in the same sense a board is. Some of the best tools out there let the team customize some sort of 'team space' or 'team wiki'—those are worth a try, but they will never be close to the real deal.

- **Team identity**: the lack of adaptability and ownership doesn't help the team create their own identity. Maybe they will grow their identity by other means, but an electronic tool is not likely to contribute in the same way that a physical team space does.
- **Cognitive matters**: studies have shown that the engagement of the cognitive mind is much higher when performing hands-on tasks like drawing or handwriting, and is lower when using a keyboard, listening to a speaker, browsing through a report, or staring at a screen.[4]
- **Using electronic tools looks a lot like old-style reporting**: At least, using these tools looks like old-style reporting a lot more than does updating the team's board. No Agile team I've met[5] has ever been motivated and proud of their electronic tool, nor have they had any fun around it. The contrary is almost always true when it comes to physical boards.

Of course, some cases will actually require electronic tools. A couple of usual situations that call for electronic tools are distributed teams that need cross-site coordination or big companies where heavy reporting and consolidation of data is needed. In those cases, some manager will be eager to throw some tools onto the team. But before you rush in, you should first question the situation and consider it as an impediment—a serious one. Do you really need distributed teams or is it 'just the way things are around here'? Is there real value—customer value—in the heavy reporting and consolidation of data or, again, is it 'the way you do things'? In fact, in many cases were I've consulted, the organization was mandating the use of a tool for team and project data extraction, but the reality showed that the use of the tool by every team was so different that there was no way of extracting consolidated data—and no one was, in fact, using that information in a meaningful way!

So again, think twice if you are all positive and clear about the use of electronic tools. Even if you find yourself in one of the few cases were the company really, really. . .*really*. . . needs electronic tools, you should use them as a complement to the physical boards—which means that the teams must update information twice, once on their local boards and then on the

[4] Several of those studies can be consulted in Indiana University's white paper *Handwriting in the 21st Century? Research Shows Why Handwriting Belongs in Today's Classroom* (2012, Saperstein Associates).

[5] By the hundreds, in case you wonder. ☺

synchronized electronic tool. If you consider this situation inefficient, let me give you a tip: *the problem is not the team's board*!

Another related aspect that Lean experts have stressed over time is not to make report-based data-driven decisions on your own from your fancy manager's office. Instead, they mandate that you spend as much time as possible at the place were the work is being performed—the *Gemba*—and to make decisions collectively based on what you see, feel, and try there. Observing the Gemba was considered so important that Toyota's Lean Champion Taiichi Ohno would bring new graduates to the Gemba, draw a chalk circle on the ground and tell them to stay in the circle and take note of everything they saw for hours.[6]

The practice of *Genchi Gembutsu*—which translates to 'go and see' or even 'management by walking around'[7]—is intimately connected to the Gemba Kaizen principle. Kaizen takes place at the Gemba; in fact, instead of Genchi Gembutsu many companies prefer to talk about Gemba Walks. Kaizen that takes place in the CEO's office is most likely another form of corporate policies pushed top-down, which is, as I hope you understand by now, against all core values of a Kaizen culture.[8]

There's a story I find everywhere in Lean literature and reported cases, up to the point that I sometimes wonder if it is just a Lean urban legend.[9] Jim Womack, for instance, describes himself as the main character of this story in his book *Gemba Walks*, but the same behavior is described in several other works. The story always starts with a company that is having a lot of trouble that decides to hire a Lean expert or a group of them. The day comes were the experts are visiting the company and everything is ready. The management committee welcomes them and tries to show them to a nice, luxurious conference room where they have prepared a lot of presentations—coffee and doughnuts included—to help them understand their situation. But the story always goes the same way: the Lean Sensei refuses to go into a closed office and demands to go and see the factory floor.

[6] This practice became known as Ohno Circles.

[7] Although 'management by walking around' seems to be a similar but unrelated practice with origins in some American companies in the 1970s, it was later popularized by some American management consultants.

[8] If I wanted to make this more radical, I'd ask what's the point of having a separate CEO office in an Agile/Lean culture? Luckily for you and me, I don't want to make it more radical. ☺

[9] I actually have friends in the car industry who swear it happened to them.

It doesn't matter how hard the managers try, the experts insist that they want to see the factory floor first. When they are finally walked through the plant, the experts point at everything, saying 'That is waste, that is waste, that is also waste'. At the end of the walk, they tell the mangers 'your problem is that you are discussing a lot of data, but you don't know how to *see*'.

Problems and impediments are everywhere—except in spreadsheets. Numbers can always be massaged to look nice—in Spain, we even have a saying: 'paper supports everything'. If you want to grow a Kaizen culture, Kaizen agents, champions, and managers must make it a real priority to go-and-see: spend time at the production line. Sit with your teams. Talk to people performing tasks. Go with your sales people and talk to customers. In five words: *get out of your cave!*

Mapping the Value Stream

If you already understand Kaizen as a cultural matter, and you have also approached Team Kaizen—growing the appropriate goals, values, and behaviors in your people—and you have also managed to schedule some periodic, structured time to study how to improve (Kaizen events and Retrospectives), my next advice would probably be to start with a value stream mapping exercise as soon as possible.

Several books have described how to perform a value stream map, the most prominent probably being Rother and Shook's *Learning To See*[10] and, in the Agile ecosystem, the *Lean Software Development* book series by the Poppendiecks.[11] The goal of a value stream map is to map the set of activities that lead from a customer order to a delivered product and analyze ways to optimize that sequence—from concept to cash, from problem to solution.

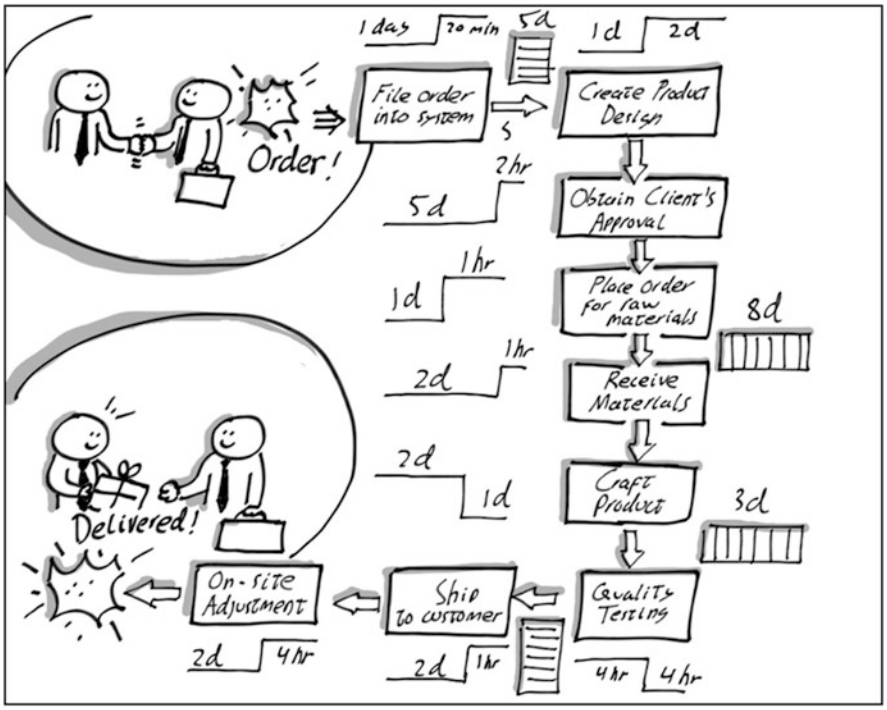
The usual advice is to start at the delivery of your finished product—or even later, when payment is received—and start mapping backwards until you reach the moment where the order was placed—or even sooner, when the client realized that he had a problem. You get to decide when the clock starts ticking and when to stop it, but the closer to the customer the better.

[10] Rother M, Shook J (1999) *Learning to See: Value Stream Mapping to Add Value and Eliminate MUDA*. Lean Enterprise Institute.

[11] Poppendieck M, Poppendieck T (2006) *Implementing Lean Software Development*. Addison-Wesley Professional.

Keep in mind this is a whole-system exercise: there is no sense in optimizing parts of the system, which only creates suboptimization.

For every hand-off, meaning every time the product or the service is passed from one person, production cell, team, or department to another, a box is added to the value stream map. The box represents a value-adding activity. Usually, above or below the box, there are two numbers representing the amount of time we were actually adding value to that specific product or service, and another one that represents the amount of time we were doing something else—for example, working on other products, waiting for resources, or solving problems.



It is usual that the time it takes to perform a given task might vary, sometimes by a little bit, sometimes by way too much. The numbers we use in a value stream map might represent the average or, for example, the norm. Variability, from the Lean perspective, is a source of waste and should be reduced as much as possible, so measuring the standard deviation might also be important in order to understand what is preventing us from achieving a more efficient state.

Between boxes, there might be queues. A queue means that when we hand off a piece of work to the next production box or ‘cell’ in the value stream, and that cell is busy doing something else, the piece of work we are following has to wait in a queue until the production cell is ready to handle it. If more pieces are coming, the queue will grow and the queuing time will be longer.

When analyzing the value stream, Lean expert Jim Womack describes several useful questions to ask.[12] These questions try to define key aspects of the value stream, including:

- What is the ‘customer purpose’ of the business process or, in other words, what’s the value we provide to the customer with it?
- Who is responsible for the business process or value stream as a whole?
- What’s the current *real* capacity, throughput, and workload of the system?
- Which are the current bottlenecks or ‘pace makers’ of the value stream?
- What’s the flow efficiency of the value stream (value-adding time divided by total time).
- How do we integrate suppliers in the value stream, and how do materials reach the different production stages?
- How does information flow up and down the stream?
- How are employees trained, motivated, and supported; how empowered are they and how is ownership assured?

For knowledge companies, ‘materials’ can be substituted for information—for example, how is information reaching the workers who are creating a product—and the considerations around pace maker processes might be complicated. Anyway, as we will see next, analyzing the current bottleneck process—and making similar considerations with the pace maker process—is also important for any kind of company.

There are also a couple of very important factors in order to understand and improve the flow of the value stream:

- **Cycle time** is the total time we need in order to perform one given unit of work.

[12] Womack J (2011) *Gemba Walks*. Lean Enterprise Institute.

– **Lead time** is the average time it takes for one unit of work to be processed, including the time waiting before we actually start working on it.
– **Flow efficiency** is the ratio of value-adding time over total time spent.

The first time people perform value stream mapping exercises, they usually find that most of their time working is, in their opinion, 'value-adding time'. Meetings are value adding. Fixing bugs is value adding. Answering e-mails is value adding, and so on.

Waste is anything that depletes resources of time, effort, space or money without adding customer value.
– Mary Poppendieck

This is not only about what the client is paying for. Ultimately, if the company has reached break-even and there is no more money coming from investors, the client is paying for everything, from the office to the Christmas party, from electricity to office supplies.

Lean companies defined three main sources or classes of waste: *muda*, the usual term for waste; *mura*, or inefficiencies because of production and process variability; and *muri*: overburden, over-process or over-commitment. Lean Software Development introduced a mapping of these terms to knowledge environments, thus listing several sources of waste in knowledge companies: handoffs, wait time, lack of information, process overhead, context-switching, wrong priorities, queues, bottlenecks, over-commitments, and rework.

My personal rule to determine if something is actually a value-adding activity is to ask 'if we were able to do twice as much of this activity—would we like that?'. If the answer is 'no', that means we should label that element or activity as 'Lean waste'. Lean waste is everything we should eliminate or, if it cannot be eliminated, reduced to a minimum. For example:

– Do we want to have twice as many managers?
– Do we want to have twice as many reports?
– Do we want to find twice as many bugs?
– Do we want to answer twice as many e-mails?
– Do we want to have twice as many meetings?

The answer to all of those is probably ‘no, we want to have as few as possible’—but not fewer than that. Once we have as few as possible, we should find other sources of waste—or figure out how could we live with even *fewer*.

Everything should be made as simple as possible – but not simpler!
– Albert Einstein

For instance, let’s say there is a value stream box labeled as ‘crafting’. We know that the average time it takes for an average product to be crafted is around 2 weeks. People who are new to value stream mapping will sometimes consider those 2 weeks to be value-adding time. But in those 2 weeks, we might be crafting five different products at the same time. We might also spend 10 % of our time in project meetings, another 30 % of our time fixing defects and taking support calls, 5 % of our time doing research, development and innovation, 10 % of our time just slacking a bit in order to prevent our brains from melting inside our skulls; and then there’s an additional 10 % of time we need to answer e-mails, attend other meetings, interview job applicants, write reports, file documents, and so on. . .

If the math is right, that leaves us with 35 % value-adding time, which we then must divide amongst the five products being delivered, for a grand total of 6 % value added time per product, or roughly 4.8 h on an 80 h iteration—hooray! That’s closer to the usual performance you should expect before a deep Lean–Agile Kaizen initiative has been worked on for some time. In fact, numbers between 8 and 20 % of overall ‘flow efficiency’ are not unusual.

In the example from a couple of pages ago, the lead time is around 37 days—16 of which the product is just waiting in queues—while all the value-adding activities add up to, roughly, 5.5 days. This is a flow efficiency ratio of around $5.5/37 \approx 15$ %—we can probably do better than that!

Visualizing the Value Stream

Learn how to see. Realize that everything connects to everything else.
– Leonardo Da Vinci

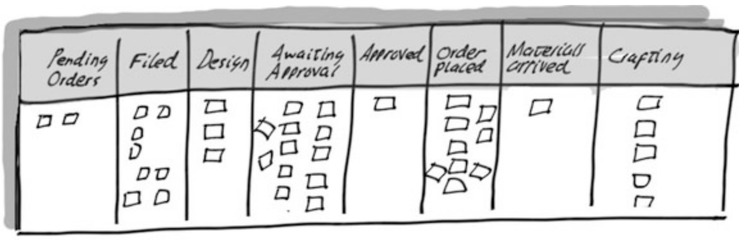
The ideal state of a Lean company is the 'one piece flow', meaning that one piece of work should be able to ideally cross the whole value stream nonstop—for instance, in the previous example, this means that in a perfect state we should be able to deliver one product in 5.5 days. This is the main goal behind a Kanban implementation. Of course, as we have shown, the more products someone is working at the same time, the less time he is actually dedicating to each product.

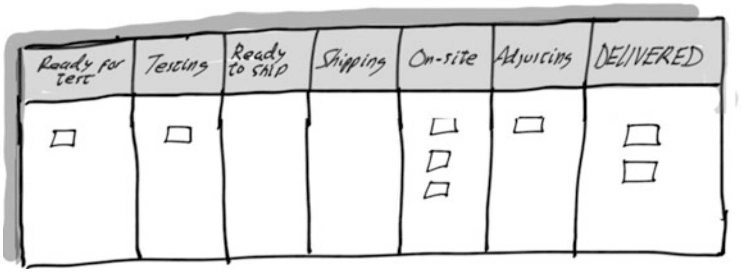
Queuing theory, systems theory, Lean theory, thousands of Lean/Kanban implementations worldwide, and plain old common sense show that, in order to be able to perform work faster, we should be working on *fewer* things, not more of them. Lots of work everywhere might give us a false sensation of productivity—'*man, look just how busy we are; if this is not success at it's best I don't know what is*'. But being very busy and delivering valuable products as frequently as possible are two very different things. We are in business, not in *busy-ness*!

Here's where a lot of people will quote the old saying—'nine pregnant women don't give birth to a baby in 1 month'. We can't argue with that. Probably, it would be inefficient, maybe impossible with the current technology, to have nine people working on the same product at the same time. But the point is that the usual, easy solution, which is to have every one of them working on a different product or feature *because it's easier for them*, causes inefficiency on a value stream scale.

Making the value stream visible is a huge step toward Process Kaizen. Not only should the value stream be pictured and published—with all measured data—the amount of work being performed should also be made visible, so we can spot improvement areas.[13] If you draw your value stream and then conduct a census of all work being performed in every stage, making it visible, what you have is your very basic whole-system Kanban board.

[13] That's consultant talk for 'problems'. ☺





As soon as you can visualize your value stream, you will start to realize what the most urgent problems are—like, in our example, you will see that filing orders is taking too much time, there is a lot of work waiting for approval, there are also a lot of materials orders waiting, we might be crafting too many products, there is not much crafted product to be tested, and so on. . .

Different product families or services can have different value streams: each of them must be mapped independently through the same process of walking and inspecting all activities from the time the order was placed until the product is successfully delivered and accepted by the customer. In some cases, especially if you are looking for a just-in-time level of optimization, where goods and materials are supplied exactly when needed, the value stream might also include your supplier's value stream. In addition, the Kanban board might need to be extended and tweaked to include suppliers' live information—this could be one of the cases where you might need electronic tools, but only for the sake of synchronizing the supplier and customer physical Kanban boards.

Backlogs Are Waste

Once of the most compelling statements in Lean ecosystems is 'inventory is waste'. A bunch of unfinished goods and materials, needed only because of our inability to make the process more efficient, is like a hidden bank account—a lot of investment, no returns. It also introduces costs and waste in the form of inventory state, inventory management, and inventory movement.

In every single implementation, non-Lean workers may complain that ‘it is not possible’ to run without inventories, a paradigm created by years of working in a wasteful manner. Remember that the paradigms of perfection for Lean companies are ‘one piece flow’ and ‘just in time’ production—materials are supplied just in time to be processed, no inventories are formed, and pieces never stop moving through the production line. Even if absolute zero-inventory perfection might never be attainable, Lean companies can experience a 90 % reduction in their inventory needs—sometimes even more. Small inventories of pieces, unfinished goods, and materials might be needed in every production cell, but as soon as one of those cells starts to grow too much, this is understood as a need for improvement.

These ‘growing queues’ are easily found on a Kanban board. As they start to form, they will be evident in a visual system, and then we will be able to identify if there’s a bottleneck, some other kind of inefficiency, waste, variability source, or maybe if it is just a temporary or accidental situation. Additionally, we can see if there is any chance that it might happen again.

When it comes to knowledge-based organizations, the absence of physical materials can make it difficult to trace what is really happening. If you look at a car repair workshop, you can see if they are idle or if there is a lot of work to be done just by noticing how many vehicles are parked in it. But as you look to the Gemba in a knowledge-based company—which usually takes the form of an office—every day seems similar to the day before, and there is no immediate way to locate queues or inventories.

Again, Kanban boards introduce the visualization of work in knowledge companies and allow Kaizen agents to detect inefficiencies. Inventories take then the form of product backlogs—information, orders, requests, and features. And in the same way that inventories are considered waste in Lean environments, Agile environments will point at backlogs and label them as waste. Big inventories might be a sign that there is a lot of business to be done, a lot of orders, and a lot of clients interested in our services—and that is true. But this also means you are looking at your backlog with vanity instead of with the eyes of a true Kaizen agent.

A big backlog means several bad things, the first one being an increase in lead time—the lowest-priority items will take longer to be delivered. Big backlogs are a sign of ‘busyness’, a wrong cultural artifact for a Kaizen culture. There is also more time invested in managing the backlog—estimating, prioritizing, planning, and road-mapping. It is harder to predict and commit to specific results and, as companies usually resist trimming the backlog, it will tend to grow and grow, sometimes even preventing higher-priority items to be processed because there are several other things 'we promised' and that 'must be done before'. This is the 'First In–First Out' backlog, which is a terrible implementation of an Agile framework.

The healthy state of any company is to have more work to do than capacity to do it. It would be nice to have *exactly* the amount of work we can process, and as we will see next, reducing variability is one of the Lean goals, but if we have less work to do than capacity to perform it, that is very bad news. If we consider this statement, then you will realize that either we establish some backlog funnel mechanism (we build this, and we don't build that), we trim and prune the backlog periodically, we add more capacity indefinitely (which might also be a terrible idea), or the backlog will inevitably grow and grow until it is absolutely unmanageable.

A trick I learned from Mary Poppendieck herself is to ask people about the size of their backlog and their average production rate. The worst case I ever traced was a company that had over 300 features on their backlog and was delivering around 50 or 60 features a year—which of course meant that they had feature requests for the next five-plus years! When confronted with this fact, they couldn't see anything wrong in it. We had a long way to go with this company. Sometimes companies argue that they know that there are several things in the backlog they will never be able to deliver, but they still want to trace them as ideas. Mary Poppendieck suggests that, in those cases, you create a 'Never' backlog and put all the ideas you might never be able to work at in it, thus keeping your actual backlog real and clean.

You should also consider the psychological effects of a huge backlog. Goals have a beneficial effect on the team as long as they represent some challenge but are still attainable on a reasonable time scale. If they consider the goal to be a severe case of over-commitment, it will affect their morale and engagement.

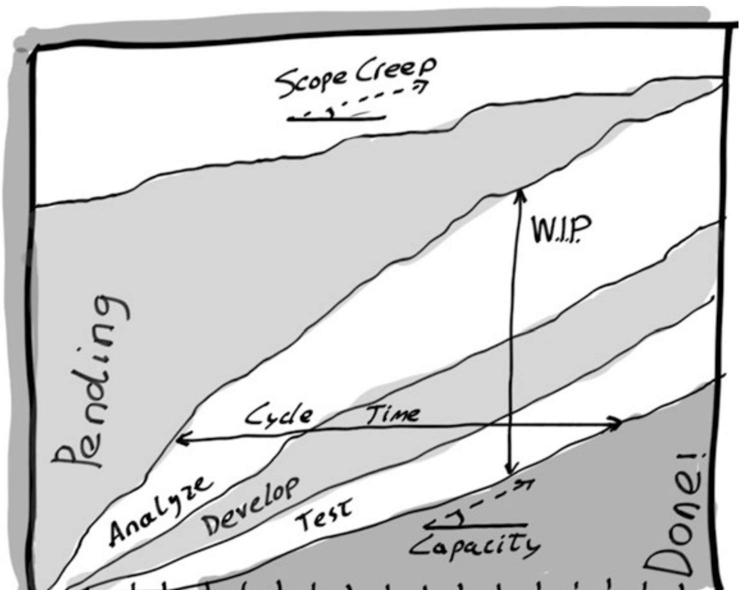
One of the problems with backlogs and Kanban boards is that they only represent the current state—it is difficult to maintain some kind of historical trend. Burndown charts[14] were designed for that, and they are an awesome Kaizen tool. In my experience, more and more Scrum teams have stopped

[14] Explaining burndown charts and basic Agile tools is beyond the scope of this book, but if you want to learn more about them I recommend that you read the *Scrum Guide* (http://www.scrum.org/Scrum-Guides) and/or the *Scrum Primer* (http://www.scrumprimer.org). A good Scrum vs. Kanban guide by Henrik Kniberg and Mattias Skarin can also be found at http://www.crisp.se/file-uploads/Kanban-vs-Scrum.pdf

making burndown charts. The usual stated reason is that they don't need them or they don't find any real value on them, but when I ask them to explain historical trends, iteration behaviors, or even when I ask them to tell me the last iteration's story, they usually fail to provide clear information. So these teams are not improving their process. Instead, they are dropping a tool because they don't understand how to use it.

The excuse that it takes effort and time to maintain a burndown chart does not hold water—burndown charts are pretty much immediate to create. Sometimes, the hidden reason is that the team does not like the pressure that a burndown puts on them, because they are actually able to see that they are not going to make their predicted pace or meet the whole expected scope for this iteration. But if you were a customer of this team, when would you like to know that things are not going to happen as predicted? As soon as possible, when you might still react, or as late as possible, when there is no chance to change anything or manage expectations?

Another tool that some more-advanced teams use is the cumulative flow diagram. It provides a good view of how backlogs and work-in-progress have evolved through the project, and it is another good tool to consider when designing your process Kaizen.



A cumulative flow diagram shows, for every day in a project or for a given time frame, the different amounts of work in every value stream stage. It is relatively easy to build a cumulative flow diagram: for every time unit (days, for instance), you just note down how many pending work units to you have, how many are in the different progress states, and how many are already done and delivered.

On a vertical perspective, from 'Pending' to 'Done', the height of the cumulative flow charts shows you the amount of work being processed in the organization at a given time—the work in progress, or WIP. The lower the WIP, the closer you are to a perfect 'one piece flow' situation. On the other hand, the bigger the WIP line, the more clogged are the arteries of your company. A WIP line that is getting wider over time will probably mean that there is a bottleneck in the system.

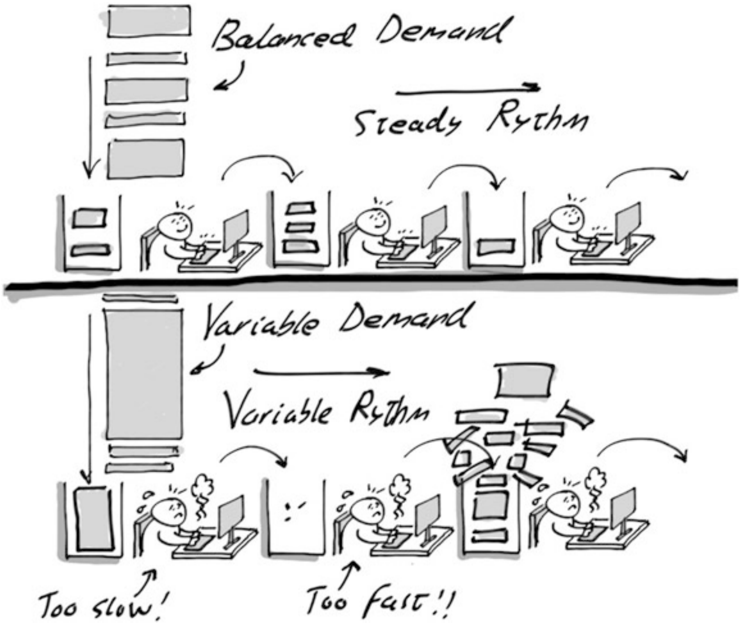
If we look at the horizontal perspective, again from Pending to Done dates, we will get an idea of the cycle time of the system we are observing. Another interesting trend is the 'scope creep'—how the Pending line grows over time. The slope of the Done line shows you the system capacity. If this slope is smaller than the scope creep, then you will never be able to finish the product.

Variation Is Waste

The perfect Lean state—never attainable—is to achieve a perfectly predictable, non-variable production rhythm. Even if this is not possible, the Kaizen way is to get as close as possible to that perfect state, then to go one step further.

Even if the production rhythm is constant, fluctuations in demand also introduce waste. When too many orders arrive, your lead time will increase, which is waste from the customer's perspective, and if you elevate your production capacity and then orders drop, you will have too many spare resources—hence, introducing waste in other form. Lean companies designed several strategies to cope with changing demand, like, for example, demand leveling—increase demand in low-demand periods, and decrease demand in high-demand periods, both by tweaking the sales pipeline or by using other approaches—or production leveling, that is, by making your production capacity more flexible.

Variability can introduce small time-bound queues. If you define your Agile process to handle seven standard-sized requirements per iteration and you suddenly receive 20, a queue will form. Sometimes there is no way to predict this variability, but in other situations people just get used to it and never wonder if there might be a way to reduce such variability.



Lean companies developed *Heijunka* devices to level workload and reduce variability. The Heijunka devices—also known as Heijunka boxes—worked as small queues between production cells that could absorb small local variances in work performance without disturbing the overall production rate. If the variance is too high and it overflows a Heijunka device, it triggers a 'bottleneck alarm'.

In mature Agile companies producing software, savvy product owners—the ones responsible for the product backlog—design work items, also called stories, on a reduced set of sizes—like small, medium and large. Every size might also have a small, predicted variability: a small item might be anything between half a day and 2 days; a medium item might be anything between 2 and 4 days. This is not perfect standardization of work packages, but it helps to reduce the variability. Kanban queues are also a way to implement Heijunka in Agile environments.

In another example of variation management in Agile companies, some service companies that I coached had big issues with unpredictable demand variability. To manage this they created a side-product and dedicated all the spare resources to the product when demand was low. In the long term, the side-product turned into a profitable multinational side-business—how cool is that?

One kind of wasteful variation that is very usual in knowledge-management environments is context-switching. This happens when people are told to switch tasks constantly in response to rapidly changing priorities from customers, reprioritization from managerial levels, or, in some cases, due to the inability of knowledge workers to manage their own time. Studies have shown that the more things your brain is working on, the lower the overall productivity. There is a loss of brain capacity when you switch from one task to another, no matter how simple the tasks.

Gerald Weinberg[15] showed the loss due to context-switching when people work on several simultaneous projects: it starts with 20 % capacity loss if you switch between two products, and increases to 75 % when you work simultaneously on five different projects! This is a huge waste from any perspective, but the terrible thing is you might not know! This number does not show up anywhere. People are working 8 h a day, and you don't know up-front if those were 100 % productive hours or whether variance, context-switching, and other waste sources are turning most of those hours into garbage!

A way of managing or, at least, analyzing the quality of the time we invest working is by categorizing demand and, later on, reviewing it. For instance, a team might create different Kanban cards or backlog items for features, support requests, small changes, rework, bug correction, and unplanned work. At the end of every iteration, this information might be reviewed to spot sources of variance or context-switching.

Another example of pernicious variability in knowledge-based environments can be observed when teams specialize too much. Suppose we have team A, who are experts in technology X, and team B, whose expertise is technology Y. If a huge strategic project comes and it needs technology X, not being able to increase capacity in this project because of teams' variability introduces an opportunity cost to the company. Hence, Agile companies favor the creation of feature-teams, which can also be seen as another way of standardization.

[15] Weinberg G M (1991) *Quality Software Management: Systems Thinking*. Dorset House.

We Identified a Bottleneck: Now What?

Once you are able to visualize your value stream, there are several ways you can improve it. You might find ways to make it shorter, to parallelize streams, to focus your effort on the less-efficient stage of the stream or, most usually, to detect where queues are growing larger, which normally means that you discovered a bottleneck in the value stream.

In his business novel *The Goal*,[16] Eliyahu Goldratt proposed what is now known as the Theory of Constraints (TOC). He basically described how the whole system chain will run at the speed of its slowest link, and established three sequential steps in order to eliminate bottlenecks:

- **Exploit**: first, make sure that the bottleneck link is running at its full capacity. Eliminate all low-priority tasks that can be done in other parts of the system. Make sure that the materials or requests coming to this part of the process have good quality and are ready to be worked immediately. Spot any interruptions, variability sources, or inefficiencies that might be making this process run slower.
- **Subordinate**: only once you are sure that the bottleneck process is running smoothly and efficiently, make the whole system run at the same pace as this process—it will be your pace maker. This will make some other processes run more slowly, and you will have some spare capacity—use it to work at the bottleneck. See if there are any resources you can reallocate in order to increase the bottleneck's capacity.
- **Elevate**: if and only if you are sure that the process is really efficient, that the whole system is running at its pace and that there is no spare capacity anywhere, then and only then may you add more resources to the bottleneck in order to increase the overall system performance and capacity.

Managers are really eager to add more resources anytime they find themselves lagging behind: it's the easy response. But if you think twice, you will realize that adding more resources to an inefficient process is just making it more inefficient. It is a sort of 'corporate cancer' that will grow and grow, eventually clogging all the arteries of your company and killing it. Of course, the process of finding bottlenecks and exploiting them is much more difficult, but in the long run is the right strategy for Kaizen—continuous improvement. Adding more resources is seldom improvement.

[16] Goldratt E (1986) *The Goal: A Process of Ongoing Improvement*. North River Press.

Subordinating is another thing modern managers have a lot of trouble with. When you implement Kanban and ask a team to limit their WIP, there will inevitably come a time when they have some kind of bottleneck or block, and they will not be able to open any more tasks if they don't want to break the WIP limit, so they will basically sit down with their arms crossed and do nothing. Too many managers think this is nonproductive and will prefer them to 'just work on something else'—hence increasing the work inventory—but of course the whole point of the WIP system, as a way to subordinate, is to be able to *stop the production line* when we spot a bottleneck or some form of inefficiency.

I work with real companies that have real customers. I know the emotional challenge of telling your people to stop, think, and improve when a deadline is near. I know that your conscious mind is telling you that it is the right thing to do, follow the J-curve and invest in long-term rewards instead of instant gratification, but your gut is screaming 'Just not now! Some other day maybe!'

There is no easy way to cope with this, other than training and disciplining the Kaizen mind as we discussed in previous chapters. The different Kaizen agents are also very helpful in order to maintain long-term focus, but in order to be free from short-term emotional pressure, they have to be absolutely detached from short-term outcomes. This is the reason I always recommend that Kaizen agents, ScrumMasters, and Agile coaches, for instance, should not be a part of the Agile team—in the sense that they should not be directly contributing to delivering products at the end of the iteration, nor should they be responsible for project deadlines.

Symptoms and Causes

One of the most crucial steps in waste and bottleneck removal is to be able to identify the root causes of the problems. Many times, the symptoms are pretty evident. We are late. The customer is angry. Our costs are high. So the company decides to craft a plan for improvement that consists of telling people that we have to go faster, delight our customers, and reduce costs. Of course, this seldom succeeds, because we are addressing the symptoms, but not the causes. 'We are late' is something that most people won't know how to fix. 'We have to many items on our desks', on the other hand, is rather direct to fix—it is an addressable issue.

There are several reasons why people are bad at detecting root causes and addressable issues. The first might be that our eyes have not been trained to see all the defects, waste, and inefficiencies in our environments—we lack the principles and tools. But also people very seldom think of what they can improve by themselves. Their usual thought process is ‘someone out there should be doing something’ or ‘it’s not my fault’.

I recently boarded a low-cost flight and could spot several disgruntled passengers complaining about a lot of things. One lady was very upset by the high temperature—I was wearing a t-shirt and was comfortable, while she was wearing a woolen jersey and carrying two jackets folded over her arm. Maybe there was something *she* could do about it instead of blaming the weather or asking to cool down a whole airport? Another lady was forced to leave her luggage for cargo hold storage because she was the last to board, there was no more space in the luggage compartments and the case was too big to store below the seats. She was complaining that the airline had lost her luggage before—but of course, she did not care to be at the boarding queue soon enough to board in the first positions, nor did she pay for priority boarding or manage to carry smaller hand luggage—so it was *not her fault, but someone else’s.*

When you think everything is someone else’s fault, you will suffer a lot. When you learn that everything springs only from yourself, you will learn both peace and joy.

– The 14th Dalai Lama

Besides addressing the sense of ownership, as we discussed in previous chapters, one of my preferred approaches to understanding problem causes is Toyota’s ‘five whys’ strategy: They usually advise their people to ask why at least five times to get closer to the root causes of their problems.

Toyota’s Chairman Fujio Cho says lean leaders do three things: Go see, ask why, show respect.

– Jhon Shook, Lean Enterprise Institute Chairman and CEO,
Foreword to J. Womack’s *Gemba Walks*

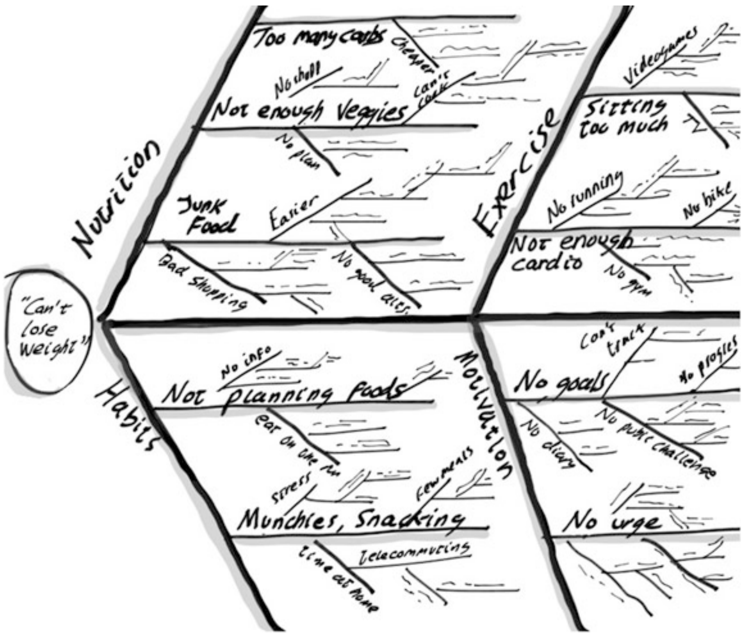
An example of the Five Whys in action would be:

– Why are there scratches in the car’s frame? Because the paint is faulty.
– Why is the paint faulty? Because there was dust in the painting booth.
– Why there was dust in the painting booth? Because the filters are dirty.
– Why were the filters dirty? Because they weren’t changed soon enough.

– Why weren't the filters changed soon enough? Because the standard says to change them every 5 days, and it was just 4 days since the last change—*actionable item: change the standard.*

Except for very simple problems, the Five Whys approach will probably expand into different directions. In complex environments, there is not just one single root cause, and hence there is no 'Silver Bullet' that will fix our problems. Most likely, there will be several causes working together to create a complex problem, and you might need to work them all at different levels in order to reduce or eliminate the problem.

A good way of keeping track of the different thinking directions created by the Five Whys is to use Fishbone Diagrams. Created by Kaoru Ishikawa as problem-solving cause–effect analysis tools, Fishbone Diagrams start with a core question or problem and then divide the causes into different domains to be explored. Sometimes these domains are predefined, and sometimes they are defined ad hoc for the problem to be solved.



For every cause domain, some symptoms might be identified and drawn as branches. In every branch, we will question why and list all the possible answers as sub-branches. We go on until we find root causes that are actionable.

There are some critical voices about the Five Whys approach. Sometimes 'why' can be perceived as an accusation, like when a parent asks a child 'why didn't you tidy up your room?'—and of course, the parent is not actually looking for an answer, he or she is just telling the child to go tidy up their room. The why approach might also mistakenly create the idea that there is one and only one reason why things go wrong. There also a chance that the why sequence might not lead to any constructive dialog, like in 'Why do you want to get healthier? To live a better life. Why do you want to live a better life? Errrr…'

While making sure that the Five Whys is never perceived as a blame game and keeping in mind that the Five Whys should lead to several answers, sometimes changing the why approach to a 'how' approach might also help you create constructive improvement dialogs: 'HOW do you want to get healthier? Eat better, exercise more, be happier…'

A root cause analyzing technique worth mentioning is the A3 problem-solving process. The idea behind A3 thinking is to try to fit all your facts, analysis, findings, and improvement plan into an A3 page (equivalent to an 11″ by 17″ tabloid-sized piece of paper). There are literally hundreds of templates for A3 problem solving—just use your favorite search engine and you will see for yourself—but most of them follow a sequence similar to:

– Identify the problem to be solved. Why is this a problem?
– Background situation: conditions, facts, environment, impact, dependencies, stakeholders involved
– Define a goal in the form of a desired state, success criteria
– Analyze root causes and key path to desired state
– Determine countermeasures: improvement plan
– Follow up and confirmation of results

The power behind A3 problem solving is not the template or the format, but instead it makes it easy and affordable to capture knowledge, to think constantly about the problem, and to engage others briefly and effectively.

Reviewing Your Agile Process

How do we apply Process Kaizen to our Agile process?

If you are already using a Kanban framework, you might already have visualized the value stream and the WIP, established some limits and you may be struggling with some bottlenecks—congratulations! Unfortunately, a vast majority of so-called Kanban teams will get stuck after a few iterations of their boards and then get comfortable with the state of their process. There is just a minority of teams who are using low WIP limits, even fewer have explicit policies, and only a chosen minority consider the whole value stream in their improvement efforts.

When it comes to Scrum or XP teams, similar considerations can be made—especially since Scrum and XP do not stress the importance of a whole-system approach, or they only imply it. Scrum and XP describe 'delivery of software' in an Agile sense—that is, delivery to the final customer in a concept-to-cash cycle of one iteration. But many Scrum and XP teams are just stuck in the middle of a giant waterfall. For them, their client is just the next process in the huge pipeline. Hence, even if the Scrum team is performing Scrum 'by the book', the overall situation is far from a Lean production state.

This same consideration can be made to local process improvements—as we've seen, they are a path to whole system suboptimization. But system-wide optimizations will, of course, be implemented via local actions and processes. It's just that improvement of those local processes must always observe this wider goal.

When using some Agile framework, the two things I've observed that contribute the most to continuous improvement is to develop and sustain the appropriate Agile culture while making sure that the teams using the framework don't just know the framework but also understand the reasons behind the processes, roles, practices, tools, and artifacts described. A team asked to limit their WIP will find countless ways of doing it in ways that do not help the system improve. A team asked to meet daily will perform the same project management reporting they've been doing for the last decades, only standing up.

Frameworks are not meant to be followed blindly, but that doesn't mean that you should start changing all the parts that feel funny in minute one of your Agile journey. The Shu-Ha-Ri model for learning and improving has been repeatedly used in the Agile community since it's origin:

– **Shu**: follow the rule. Do Scrum/Kanban/XP as they are described
– **Ha**: extend the rule. Adapt the framework to your specific needs
– **Ri**: dissolve the rule. Be Agile. No longer care if whatever you are doing is Scrum, Kanban, Scrumban, or something else.

One problem is that people jump to the Ri stage too fast—they start mangling, tweaking, pruning, and cherry-picking some framework without really understanding it. Consider martial arts: some schools will award you with a black belt in 2 or 3 years' time—some even on a weekend course! The most amazing martial artists I've ever seen train people for 10 years or more before awarding them a black belt; and then they remind them that the black belt is probably just the beginning of the Ha—adapting the martial art to their own.

So, one way of improving your current process is trying to better understand the underlying reason for the framework elements, then seeing if your process is providing those benefits or is failing to deliver. Even more: try to spot things you removed from the framework because they did not fit your situation or you did not feel capable of doing them—that might absolutely be an improvement area!

Some people say that the frameworks are 'best practice' and that 'all best practice is past practice'. I myself defended the idea that best practice might be past practice if the environment changed since the last time we used it. There are also some people who say that frameworks are 'one size' and 'one size does not fit all'. My personal perspective is that, if riding a motorbike, it is a good practice to wear a helmet, and I don't care if that is past practice or one size—you wear a helmet, *period*.\

Also, if you are using just one framework, you should review the reasons you are *not* using some others. People into XP are usually very concerned with engineering practices and not so much with whole-business value streams. Scrum teams are usually focused on rates, while Kanban teams are more concerned on lead time—cycle time—and WIP limits. Combining different frameworks may give you new perspectives on what and how to improve.

Anyway, please remember that Agility should not be defined by conforming to the process, but for your ability to be Agile. You should be reviewing your Agile process with the purpose of being Agile, which means you should start by asking yourself a short set of questions:

- Are you delivering useful, working software frequently and with short lead times?
- Are you collaborating with your customers, delighting them, and incorporating short feedback loops to adjust and evolve your product through gradual changes and increased functionality?

– Are people working on teams, collaborating and interacting frequently, self-organizing, engaged, and making themselves responsible and accountable for the development process and its improvement?
  If in doubt on any of those answers—*start with that*!

Summary

Lean and Agile companies start with purpose and understanding of value from customers' perspectives. Only through a deep, shared understanding of value can we design meaningful improvement strategies and identify the adequate sources of waste. Kaizen agents must learn how to see waste, although waste depends first on your definition of customer value. A shared understanding of waste sources and impacts must be fostered by Agile agents.

The next step to improving your process is actually defining your current process through a set of standards. Those standards are meant to stabilize the process, but should be changed on a regular basis. Remember: no change, no Kaizen. Processes should be simple, visible, and owned by the production/development teams performing those processes. In order to understand and improve the process, Agile agents, leaders, and managers should go to the Gemba, the place where value is being added, and learn how to see and identify value and waste on the production line.

Visual management should be enforced in the Kaizen company. The use of information radiators enhances the results of Gemba walks and helps to build a culture of transparency, ownership, empowerment, and collaboration. A whole-system value stream analysis is a good way to continue your process improvement, as it will tell you about the efficiency of your process and let you spot your bottlenecks.

As soon as you visualize work in progress using your value stream visualization, you will also realize the backlogs, queues, and inventories being formed over your whole process—every one of those is a chance to improve. The Theory of Constraints is a good way to approach bottleneck removal, although for backlogs and variability you might need different approaches as discussed.

Kaizen companies must make sure there is a defined process for pointing out issues that need to be improved and bringing up problems, defects, and impediments. For any problem you are able to identify, a root cause analysis process should be run to make sure that we address causes and not

symptoms. Designing a goal, realizing the gap between the desired state and the current situation, defining improvement plans, and then making sure there is a follow-up process are key to success. The Five Whys and the Ishikawa cause–effect or Fishbone Diagrams, as well as A3 problem-solving processes are good tools to engage in these activities.

Finally, when it comes to the improvement of your Agile process, make sure you understand the underlying reasons for every role, practice, process, or artifact. Being able to perform your favorite framework by the book is a good start if you truly understand the goal of the framework and not only the description of its elements. Combining different frameworks and assessing your overall systemwide Agility are also good ways of understanding where you need to improve.

Things to Try

– Make sure everyone in your company understands your purpose in terms of customer value. Before you rush to communicate it, interview your customers and check with them to see if your value proposition is, in fact, the one thing they really appreciate about your product or service. Use your value proposition/customer purpose to define your main improvement goals. Use these as strategic goals for your Kaizen initiative. Make sure you review those goals and their improvement frequently, with preference for a shorter suitable time scale.

– If your teams are still not using team boards/Kanban boards, this is another chance to reconsider their use, even if you are also using some sort of electronic tool. If you already have team boards/Kanban boards, congratulations, but make sure they also include explicit policies as a first way of establishing and enforcing the use of standards.

– Make sure your team has some kind of standards they honor. Standards can include coding standards for software teams, rules to use tools and repositories, ways of working/ground rules, and framework rules. Ask your teams to spot any impediments that might be related to the fact that everyone works in a different way, then ask them to propose some way a standard could help to minimize that impediment. Ask them to figure out a way to tell some other team how to perform exactly what they are doing, minimizing the training and knowledge-transfer time.

– Review how often do you visit the production line. Make sure you understand and frequently check all team boards to spot impediments, bottlenecks, or improvement areas. If you cannot spot the current improvement goal, ask teams to teach you how to see it or else re-engineer their boards. If there is any way you can move closer to the production line, check with the teams to see if that might be a good idea—or just a worthy experiment.

– Conduct periodic value stream analysis exercises. Make them a business priority. For every product or service family in your company, make sure there is a designated champion for it—be careful not to call him or her the ‘owner’, or everybody will think *he or she* is the one responsible for value stream improvement. On every value stream mapping exercise you celebrate—one every quarter might not be a bad place to start—analyze for the biggest impediment or bottleneck and set an improvement goal. Communicate the improvement goal to the teams and make sure they have enough support to conduct improvement plans that help to achieve the overall goal.

– Have some people from outside your company, maybe even from some other market sector, review your value stream. If there is anything they find awkward, don’t dismiss their surprise assuming they don’t understand your situation: sometimes, the ‘beginner’s mind’—*Shoshin*, in martial arts—is needed in order to consider possibilities different from those found in the expert’s mind.

– Encourage your teams to maintain visible information regarding their process performance: team velocity, cycle time, average size of queues, and work in progress. There is plenty of information they can provide using simple visual tools. Just make sure no superfluous information is asked for and that your company does not run into ‘analysis paralysis’: focus on a few meaningful and strategic metrics that are relevant to your current improvement goals.

– Kanban boards, burndown charts, and cumulative flow diagrams are usually valuable for any kind of business concerned with productivity, capacity, and time to market. For more basic team boards, just writing on the Kanban cards/sticky notes the different dates when there was some change to the state of the work item (entered backlog, work started, waiting for approval, etc.) will give you the ability to build histograms and to analyze the average lead time/cycle time or to detect relevant bottlenecks and queues.

– Introduce A3 reports and A3 problem-solving to your teams. Encourage them to use this kind of document to keep track of their improvement efforts and to have a more efficient way of reporting on Kaizen events.

– If you are able to spot some tricky root cause, don't spoil the fun: run a root cause analysis workshop and see if your teams are also able to spot the root cause. If they aren't, help them and guide them to see the root cause, so they train their vision and are better able to detect root causes in the future.

– Take a look at my ScrumBan presentations and videos (found through http://www.proyectalis.com/en/AngelMedinilla). The ScrumBan approach I have used with several customers worldwide has proven to be a good way of better understanding uncertainty and variance in knowledge-based environments by categorizing demand and introducing service level agreements/quality of service.

– Review the Scrum and Kanban resources proposed in this chapter. Learn more about XP. Consider introducing new practices from different frameworks, maybe one by one, with a priority order established according to your strategic improvement goals. For instance, if you have to improve quality, you might want to start with engineering practices, whereas if you want to reduce time to market you should start by introducing flow-related practices.

6 Product Kaizen

Improving the Stuff You Make

Months passed and the metrics were impressive. Production time was getting visibly shorter, and the company was able to get more work done. Morale skyrocketed as the most important deadlines were consistently met one after another – something that was unprecedented in the company's history. The CEO even allocated some spare budget to throw a company 'Kaizen celebration' party, and then people just went crazy.

Then one day, the stone cold news came in: there was spare capacity. We couldn't sell enough products to keep everyone busy. At the beginning it wasn't a big deal. There was less pressure on the teams for the first time in their lives, and they could dedicate some more time to learning and process improvement. But soon it started feeling uncomfortable. The company was doing fine, they were able to sell as much as always: it was just that they felt they could be doing much more.

Some people proposed creating a new product line and opening new business opportunities, and they were given some time to work on that, but their efforts were clueless. They just proposed more and more ideas, but none of them seemed worth investing in. The new product initiative was finally canceled and everyone went back to the usual daily tasks, which kept feeling strange.

Michael asked the teams if there was anything they could do to improve the product, but everyone was positive about it: the product was just fine the way it was right now. They had cut down costs, added a lot of features, the pricing was right when compared to their competitors, and they were able to produce it much faster.

Á. Medinilla, *Agile Kaizen*, DOI 10.1007/978-3-642-54991-5_6,

Being faster and having a better product, Michael couldn't understand why they were not able to sell more products and keep the production line busy enough. You know, it was nice not to be crazy busy, but again they just felt like they could be doing much more.

He decided to talk to the sales team, which hadn't been very engaged in the Kaizen program. They just had to go out and sell their fantastic product, they didn't know about production processes and team work, there was no point in including them in the Kaizen initiatives.

When he appeared in the sales team offices, everyone lifted their faces from their computers, some of them were even still on the phone, and stared at him with an amazed look. Michael was petrified for a couple of seconds, but then took a deep breath and started walking confidently to the sales manager's room.

George, the sales manager, just stared at him with the same amazed look. Before Michael could say anything, George greeted him: 'Well, it seems hell finally froze over, and the engineering boys decided to pay us a visit, uhm?'.

Michael didn't understand: 'What do you mean? I mean... don't you talk to engineers all the time?'. George adopted a sad face: 'Michael, you don't understand. We send orders, we ask for deadlines, but when we try to tell your people about the customer needs and how we need to change the product, they just dismiss us because they are the savvy engineers and we are just sales grunts'.

Michael just couldn't believe it: 'What? That can't possibly be true... In fact, I came here to see if there was anything you guys could do to sell more product... And my fellow engineers just told me there is no way they can improve the product!'

George laughed sarcastically: 'Mike, that's because they are engineers and they haven't talked to a real customer for decades. The product is fine for them, but our customers are not like them! We are losing customers to our competitors because they are able to change their product very quickly and provide new features for the new problems that our customers experience. We, on the other hand, have been producing the same thing for two years – and please, don't get me wrong, I love our product. But you guys are so focused on producing the same thing faster and cheaper, and that is not the whole story. Sometimes I feel like you guys are investing a lot of effort in *fixing something that is not broken... It's like you want to create a perfect thing nobody wants!'*

Michael was still confused: 'George... I think you might be a little bit negative on this... We added tons of new features to the product and introduced the latest technologies into it so...' George interrupted him: 'You see? You are exactly like them. You think that you know our customers and what's better for them. But you know what's the most usual complaint we receive from them? Our product is difficult to understand and use! But your people still think the user interface is fine and they keep making it faster and adding more features, which of course contributes to making it even more confusing and difficult to use. For heavens sake, half of the features of our product are never used by our customers! It is slower than most of the products in the market, it crashes, the maintenance costs are high, it's more unreliable... And then you have the guts to come here and tell me we should be selling more products!'

Michael was just terrified. He realized that he had made a huge mistake by leaving the sales people out of the Kaizen initiative. He had improved the production teams, the production process... But he actually never cared about the product itself.

That was going to change right there, in that precise instant. But how?

Process Is Not the Whole Story

Try to think of the worst possible product you've ever seen—maybe not in terms of quality, but just some product nobody in his right mind would like to own. In my training courses I use a couple of examples: dining trays that fit into the steering wheel of your car, watermelon transporters, and TV-equipped coffins.

The fact is: you don't know if the teams creating those products are the most Agile teams ever. They might be doing the whole set of Agile practices, have amazing coaches, and improve their process continuously. Still, they are being told to create a product nobody wants, and they are congratulated when they deliver it.

Some might argue that 'this is not Agile' because there is no true customer representative, just one person who *thinks* he knows what the market needs, and, if the *visionary* fails, that's not a problem for Agile product development: we were just developing the wrong product in a very Agile way. Some other people might also argue that you didn't care to understand value from the customer perspective—but of course, your customer, the one paying for the development, might be this visionary.

My answer, in any case, is *who cares?* If Agile or Lean are not to be blamed for stupid products, fine, no problem from my side. But we still have the problem of developing the wrong product in a very Agile or Lean way, so maybe we have to look somewhere else to include a product perspective in the Kaizen culture—I mean, if you care about the survival of your company, of course.

Consider another improvement area: customer support. Agile does not tell you how to improve that. Even worse: customer support is considered waste under a Lean perspective—it should be reduced to the minimum through good quality products and good product documentation or training. In fact, customer support requirements coming to an Agile team are usually considered context-switching and a threat to team focus. Still, even Toyota must run customer support lines, and customer support is one of the most valuable factors clients assess when choosing between different suppliers—so there's actually *some* value in providing good customer support.

Think about Zappos, the American online shoe retailer that achieved $1 billion in gross merchandise sales in 2008 and was acquired in 2009 by Amazon in an all-stock deal worth about $1.2 billion. For Zappos, customer support and customer care are their core product and key competitive advantage. Not shoes, not sales: customer support. Their customer service consistently ranks as one of the best in the USA, has been compared to those of Jaguar or Ritz Carlton, and is considered superior to that provided by BMW, Cadillac, or Apple.

Is Agile the source of Zappos customer service? Probably not. In fact, if you run a value stream analysis over Zappos' customer service stream it might not make any sense—they are not even measuring call times in order to be faster and handle more calls! In fact, they encourage support agents to take their time—the longest recorded call took ten full hours!

The product perspective is not only about the product itself: it also includes ways to reach your clients and how to manage your value proposition. You might have the best product on earth, but if your clients don't know about it your company will probably fail. If you are providing some web service but people don't know that your web site exists, that does not mean that you failed to understand value or how to improve your product—you forgot to improve your *marketing*. As we will see in this chapter, market funnel improvement should be also a subject of Kaizen.

The point I'm trying to make here is that process improvement and team improvement are not the whole story: you have to care about your product, your customer service, and your marketing strategy too. Additionally, when it comes to product or service improvement, it is not all about efficiency, time, and metrics.

Starting with a Vision

Once again, everything begins with the customer's perspective. Successful products are able to solve a customer's problem and provide him with some value that other alternatives or workarounds—including just not solving the problem and getting used to it—do not provide. Some entrepreneurship books and experts tell you to find a problem that your customer has and try to solve it. But of course, in order to do so, you need to determine who your clients are. And if you don't know what your product is, how can you determine who your clients are? You probably closed a vicious circle.

For me, it all starts with a vision. The vision must be a compelling statement that declares what our company is going to be about. It does not describe our product or our business model, but it sets the stage for all subsequent efforts in product management—including customer development, product development, and product improvement.

As I explained in my first book, the problem is that most 'mission statements' or 'strategic vision' documents are full of buzzwords, consultant talk, and corporate gibberish. The vision should be a small statement, and be clear and memorable. A five-to-ten word, verb-target-outcome format is a good way to start. Consider Google's memorable vision, 'organize the world's information and make it universally accessible and useful'. Here, the action is to *organize*, the target is the *world's information*, and the outcome is that it will be *universally accessible and useful*. The same format has been successfully used by many world-class organizations. Some examples include 'find and discover anything you might want to buy online' (Amazon) or my personal favorite, by John F. Kennedy: 'Put a man on the moon and bring him back alive before the end of the decade' (NASA).

Another problem with many visions is that they are not about the customer at all. Consider Ford's mission statement: 'An exciting viable Ford delivering profitable growth for all'—how is this relevant to their clients? Or for another example, ExxonMobil's mission statement: 'ExxonMobil Corporation is committed to being the *world's premier petroleum and* *petrochemical company*. To that end, we must continuously achieve *superior financial and operating results* while adhering to the highest standards of business conduct'. As you see, it's not about their customers, but about them becoming greater, bigger, and more profitable.

If you push these kinds of vision into your corporate culture, your people are not very likely to care about value from a customer's perspective—unless they consider their stakeholders as their main customers, which seems to be the case. As Dan Pink put it in his famous Ted Talk,[1] bad things happen when the company profit gets unmoored from a noble purpose, both in terms of questionable ethics and crappy products and services.

There can be several visions of the company—by product, by brand, by areas—and they can change over time. But too many vision messages might get confusing and even create contradictions, thus damaging tremendously your corporate culture and your product management efforts.

The Problem with the 'Visionary'

In his must-read book, *The Mythical Man Month*,[2] Fred Brooks described how IBM identified a 'surgical team' pattern on a few very successful projects. In those teams, there was a 'master surgeon' programmer who was much better than the rest, and he had all the design, architecture, and the core of the application in his head. The rest of the team just helped him with minor stuff and were absolutely bound to him, just as nurses are bound to the surgeon in a surgical team.

Even if the results of these kinds of teams were good, the model was soon abandoned for many reasons:

– Surgical teams were unreliable and not robust. If the master surgeon had to go or was not available, the whole team was pretty much useless.
– Surgical teams couldn't scale. As soon as the master surgeon got inundated with customer requests, there was no easy way to accept more work.

[1] http://www.ted.com/talks/dan_pink_on_motivation.html

[2] Brooks FP (1975) *The Mythical Man-Month: Essays on Software Engineering*. Addison-Wesley.

- Surgical teams damaged motivation. People were not learning because the master surgeons were withholding critical information in order to preserve their status. They were also not able to self-organize, as the master surgeon told everyone what to do and how to do it. The assistants did not feel any ownership of the project.
- Attracting, hiring, and keeping the master surgeons was hard.
- Surgical teams lacked creativity and innovation, as there was always just one perspective—that of the master surgeon.

Therefore, this model was abandoned over time, and the cross-functional team took its place. Cross-functional teams are successful, and they are also more scalable, reliable, and maintainable. They also provide increased learning, higher motivation, and more creativity and innovation.

Ironically, when it comes to product management, many Agile frameworks and so-called experts identify a 'product master surgeon' and describe him or her as someone that has a 'product vision' in their head and whose duty is to transcribe it to some minor product-mindless drones who will build whatever he or she tells them.

Just to be clear, for 'product management' I understand, at least:

- Be able to successfully identify our market, our customers, their problems, and their needs. Define value from the customer's perspective.
- Be able to propose valuable, usable, and feasible solutions to our customer's problems.
- Be able to provide a viable business model for our solution.
- Be able to guide product development in order to provide an optimal solution.
- Be able to provide constant feedback and customer collaboration.
- Be able to maximize our solution's market success.

As you see, it's not only about Agile 'product ownership', which is mostly about describing a product, guiding the development process, and interacting with the customer—Agile product management also includes generating a viable and validated business model, making sure we are working on the optimal product, and maximizing market success.

For me, the same considerations made for the surgical team might also be made for the product master surgeon. If you truly have a visionary, you might do well just following his vision. But as we mentioned in past chapters, even Steve Jobs, the *über*-visionary, had some very bad moves in product management that maybe could have been tackled by a product management team that had several different perspectives.

As Jeff Patton puts it, 'The product owner role is a stupid idea',[3] and a team of people with different backgrounds performing all the product management tasks might make more sense. It does not mean you should ditch your product owners, customer representatives, client proxies, or whatever title you are using; but it might be more efficient to make them lead a cross-functional self-organizing team dedicated to product management to be sure you don't run into all the pitfalls of a surgical team.

Product Space Versus Problem Space

One key characteristic of good vision statements, besides including some customer-value statement, is to provide information on the problem we are trying to solve while not committing to any given solution. This is what I call 'problem space' thinking instead of 'solution space' thinking. When John F. Kennedy urged NASA to put a man on the moon and bring him back alive before the end of the decade, he was not prescribing any means of doing so. He just described the problem to be solved and left the solution-related decisions to the product management team.

There is a very popular urban legend about the Americans designing a million dollar ball-point pen that could write in zero-gravity. Meanwhile, the Russians used a pencil in space. While the true story is not exactly like that,[4] the point of the urban legend is to show how you can be terribly wrong if you slip from problem space (write in zero gravity) to solution space (*pen* that can write in zero gravity) and just care about the product, the product, the product.

Thinking just about the product can cause terrible mistakes. In Spanish we have a saying: 'if you were born a hammer, all you see is nails'. Even

[3] Session at Agile 2012 conference—no slides or video available.

[4] Curtin C (2006) Fact or Fiction?: NASA Spent Millions to Develop a Pen that Would Write in Space, whereas the Soviet Cosmonauts Used a Pencil. *Scientific American*, Dec. 20th 2006 (http://www.scientificamerican.com/article.cfm?id=fact-or-fiction-nasa-spen).

when confronted with a customer who has bolts, you will try to use a hammer instead of a screwdriver because you are thinking in product space, not problem space. Harvard Marketing professor Theodore Levitt was famous for his saying, 'sell the hole, not the drill'[5]—another example of thinking in problem space instead of solution space. Levitt also describes how the American railroad industry lost their clients because all their value proposition was made around trains (the product) instead of people transportation (the problem).[6]

Most frameworks and production standards based on just 'processing requirements' can lead to huge waste and unneeded products. If there is no customer assessment and support during the problem analysis and solution design, the customer might just assume that whatever he is asking for will solve his problem, and he might be wrong. On the other hand, if there is no validation of the customer–problem–solution assumptions before we rush into development mode, we might be building very efficiently and quickly something nobody wants.

The old saying, 'customers are the ones running our company', means that the company should focus on customer value; but your customers are not actually responsible for your product management process. In the best cases, as we will see next, customers will be *part* of the product management team. In those cases, just doing 'whatever the client says' should be considered as a team dysfunction.

In his 2011 book *Lean Startup*,[7] author Eric Ries combined several frameworks—most notably, Agile and Customer Development—to create a product approach that starts by defining all your assumptions and testing them before you go into product or company building mode. For 'assumptions', Lean Startup understands everything you *think* you know about your business model but that has not been validated yet through a customer experiment.

The core assumptions—those that, if proved wrong, would invalidate your whole business model—are those related to the customer–problem–solution statement:

[5] Raynor ME (2003) *The Innovator's Solution: Creating and Sustaining Successful Growth*. Harvard Business School Publishing Corporation. p. 99.

[6] Levitt T (1960) *Marketing Myopia*. Harvard Business Review.

[7] Ries E (2011) *The Lean Startup: How Today's Entrepreneurs Use Continuous Innovation to Create Radically Successful Businesses*. Crown Business.

- Who is your customer? Is there any market segmentation? Are there different kinds of customers? Is there any relevant information about the customers that can influence the way we solve their problems?
- What is the problem we are trying to solve? How do we know it is a problem? How are customers currently dealing with that problem? Are there any workarounds? Are there any other products or solutions our customers are using to deal with that problem? What are their key benefits and their main defects?
- What solution that we can provide that is better than any other solution that the customer is using right now? What is the value proposition and the key advantage? How will we know we are right?

Overall, anyone in the company should be able to answer the 'why are we building this' question about our products and services—and the answer 'because the client/the boss says so' just leads to mediocre and unwanted products, in addition to bad corporate cultures and low employee engagement. Remember that one of the most accepted definitions of quality in Agile/Lean environments is 'conformance to client expectations', so in order to build quality into the product you must perfectly understand those expectations.

Even more—you should manage those expectations and act as a value consultant for your client, because too many times your client will be in product space thinking—'I came to you because I want this product'—instead of problem space thinking—'I came to you because I have this *problem*'. If your client is asking for the wrong product and you succeed in meeting his expectations, you might define that as 'quality' or might want to claim that it's not your fault—I don't care: I see very little value and too much waste.

The Client Is Also Part of Your Team

Probably one of the most valuable and powerful, yet neglected principles included in the Agile manifesto is 'customer collaboration over contract negotiation'.

Making the client join your development team is not a new practice. IBM coined the term 'Joint Application Design' in the mid-1970s to describe the practice of bringing together users and developers to design or even develop products together. The idea is to be able to design better solutions through common understanding of the problems and challenges.

Still, when I visit Agile teams throughout the world I very seldom find true examples of fruitful collaboration between development teams and their customers. Most of the time there is a client proxy or customer representative in the form of a product owner, product manager, or—even worse—project manager. Not because project management is bad or wrong[8] but, to be honest, project managers tend to be more concerned about tasks, deadlines, and cost than customer satisfaction, market success, or understanding real client needs. You could argue that this is 'bad' project management and you would be right—let's leave it here with that common understanding.

Anyway, as I was explaining, there's very little customer collaboration in most Agile teams. What's even worse: the client is very often seen as the enemy. He interrupts the team, asks for changes, but then he's never available for clarification, validation, or impediment removal. In many cases, the person from the customer's organization who interacts with the development team is, in fact, *not* the customer, but someone *managing the project* from the customer's side. This introduces a lot of noise and invalid assumptions—or assumptions that are validated by the wrong person, which is the same as validating them ourselves. This situation is what I call 'developing products for a *dude*', in the sense that the *dude* does not really know what the problem is about, nor he is able to provide us with real insights on the customer–problem–solution assumption—he's just been commissioned to deliver a feature list on a budget and a deadline, and he wants to achieve this with the least possible pain and effort.

Some of the most important impediments to team–customer collaboration are:

– **Lack of trust/transparency**: the company fears that, if the customer is constantly collaborating with the team, he will find out bad quality issues, hidden defects or any other kind of bad practices. A sort of 'black box Agile' is created where the customer is not able to see what is happening inside the development process.
– **'Us versus them' syndrome**: this usually happens when the company or the team have been abused by clients in the past and, instead of developing practices and skills to constructively deal with these situations, they just prefer to kick the client out of their environment and minimize their interactions—which of course introduces huge risks into development

[8] I was a project manager in the telecommunications industry for 10 years, as a matter of fact.

itself, but make their lives more comfortable in the short term. Another version of this problem is the 'tyrannical customer', which again signifies the inability of the company or the team to deal with such situations.

- **No Agile alignment**: the customer is not concerned about Agility or might have been using a *waterfall-ish* approach long enough to consider that the right way of managing a development project is to produce a highly detailed up-front requirements document—the 'Big Design Up-Front', BDUF. After that, he believes that his part of the project is finished until the final milestones arrive and he appears again to check everything against the requirements document. This behavior might also be identified as a lack of customer involvement.
- **Bureaucracy and large customers**: this is a special case of no Agile alignment. The customer might actually want to collaborate, but his company demands that a bunch of paperwork, meetings, reports and authorizations take place even for the slightest decision. This might be defined as a lack of empowerment problem.
- **No real customer/lack of customer involvement**: the customer's project manager or representative is not really concerned or involved in the success of the project. The reasons for that include the client wanting to push all responsibility to the supplier, or the client's representative not being really affected by the project or product, so he or she sees it as a burden more than a challenge, an opportunity, or a part of his or her responsibility.
- **Lack of focus**: the customer's representative is really willing to support and collaborate with the development process, but is just too busy. The usual reason is that collaboration with the development team is not a real priority, which also means that the customer does not see real value or the competitive advantage of customer–team collaboration.
- **Lack of customer skills**: the customer wants to collaborate, but he lacks the required skills for teamwork. It might mean that there is not enough support, training, and facilitation for the development team—including the client.
- **Geographical dispersion**: This is a problem in the middle land between lack of focus and lack of skills. The client is not able to move to the supplier's office, or vice versa, and there is not enough experience in using remote collaboration tools and practices.
- **Too many representatives**: different persons from the customer's organization are providing different input and sometimes contradicting each

other—this is usually a process problem; an effective collaboration framework should be enforced before the project starts.

There are no easy solutions for these problems. Understanding what exactly is your case—if it is described by one or more of the above reasons, and which of them—might help you define an appropriate improvement plan. Some all-purpose actions that you should implement in order to enhance customer collaboration are:

– **Ensure Agility first**: if we try to make our client join the development team too quickly, even before we are able to implement the most rudimentary parts of any Agile framework with minimum of efficiency, the results might be bad and prevent our client from trying again.
– **If possible, make it gradual**: you might start with biweekly meetings to review the results and the plan. Later, you can start asking your customer to attend some Kaizen events or daily meetings. As the client starts to realize the value of those, he might be easier to persuade in order to obtain more collaboration from his side.
– **Establish a common collaboration framework**: a very common problem is that companies create a value proposition or sales pitch that is centered on technology, features, price, and deadlines, but they fail to include some information on how the project is to be performed, conducted, and managed. If we really understand Agile as a strategic part of our company, we should be ready to explain in our value proposition how working on a more Agile way will help both the client's organization and ours to maximize the return on investment in a win–win relationship. On the other hand, contracts, project charters, and project kick-offs should foster discussion on what is expected from the client's side in terms of dedication and responsibility. The collaboration framework should not only describe an Agile development process, but also include descriptions of shared responsibilities, the customer representative's role, how to manage changes, or how to deal with conflicts and reach consensus.
– **Make sure everyone understands this framework at the beginning of the project**: constantly communicate the project's ground rules or collaboration framework and, if needed, make sure there is training and support available for your client—it should be part of your value proposition.
– **Demand good representation**: one of the key points of the collaboration framework should be to have one ultimate empowered-enough

representative from the customer's side with both ability to provide access to any resources that might be needed in the development process and to solve any dispute between other customer representatives, ensuring that, if needed, there will be a single contact point for prioritization and clarification.

- **Don't overlook soft skills**: dealing with conflict with your customers might be tricky, specially considering that their perceived power is way greater than the team's. Train everyone in nonviolent communication, dealing with conflict and constructive argument before sending them to discuss with the client.
- **Follow Up**: As with everything Kaizen, daily follow-up of customer collaboration dramatically enhances your chances of success.

What if the customer rejects our collaboration terms? From my real-life experience I know—too well, in fact—that several times the customer will actually fail to collaborate. This will happen either openly—he will reject to attend meetings or provide feedback—or in a more subtle way—he will promise anything, but will fail to live up to expectations.

Despite the evidence, it is still hard for me to be indulgent with the situations where clients fail to collaborate with the development teams. I believe that the main reason is that we fail to explain and show on both the rational and the practical levels all the benefits that the client obtains if he is involved in the development process. But, of course, as a human being, the client is also subject to the hard-wiring of his brain, and he will prefer the immediate reward of not being required to work on the project to the delayed gratification of better results.

If we believe that the root cause for lack of customer involvement is the customer's mindset, the evangelizing and training efforts performed by our Kaizen agents will take a new dimension: it's no longer enough to convince our organizations, we have to push the message beyond its boundaries up to the customer—and suppliers!

One crucial aspect that should be enforced by your customer collaboration framework is to be able to share both project profits and losses. There are several 'Agile contract' approaches, but my favorites are those that include mechanisms to share risks and results. 'Fixed time, scope, and costs' contracts push all risk to the supplier, but on the other side 'time and materials' contracts will push all the risk to the customer. None of these frameworks foster trust and collaboration. A shared risk approach will divide profits if the project runs smoothly, or losses if there are any delays or unexpected problems in it, hence giving your customer an incentive to be involved in the development and good management of the project.

For instance, you could set a 'target scope and time' to deliver your product. If you are able to deliver the target scope in less time, say, 2 weeks earlier, you could give your customer one extra week of delivery, which might be worth a couple of extra features, and keep one 'free week' to yourself. The client pays the same, but obtains more. You earn the same, but work 1 week less. On the other hand, if your project is running 2 weeks late, you could commit yourself to work an extra week and your client would have to cut 1 week worth of scope out of the project backlog—thus sharing losses. This is just one example; there are plenty of other approaches.[9] Just find the one that improves your customer collaboration framework.

Remember that most if not all the considerations made at customer collaboration can apply to yourself when it comes to your relationship with suppliers. Make sure you set an example by collaborating with your suppliers and helping them to set a mutually agreed-upon collaboration framework that provides more value to both organizations. In the same way, your teams will often have to collaborate with internal departments, areas, teams, or value units in a very customer–supplier way. Although this should be changed over time to a more cross-functional and less silo-based approach, ensuring good internal collaboration is also a way to start before pushing your collaboration framework to your external customers and suppliers.

Finally, as we'll see next, early hands-on involvement of your customer in the development process is both a key to project success and a boost to customer collaboration over the whole project.

Knowing Your Client Is Knowing Your Product

How do we put all this together? If you already have a compelling vision as described above, working on your customer–problem–solution statement is a good place to continue.

Some companies argue that they already have established products with long-time customers, so they don't see the point in restating the customer–

[9] I have some ideas at http://www.slideshare.net/proyectalis/120521-agile-contracts-21, but you can probably find some more with an easy web search for 'Agile contracts'.

problem–solution proposition. Similarly, they do not consider them assumptions in the sense of hypotheses that need to be tested.

For those companies, I would ask them to review Kodak's failure case as described by former Kodak executive Vincent Barabba in the book *The Decision Loom: A Design for Interactive Decision-Making in Organizations*.[10] It describes how Kodak invented the first digital camera in 1975, but the managers dismissed this potential advantage because *it was not about film*. Researchers at Kodak continued to improve the technology—in 1986 they created the first megapixel camera—but still the company was confident in their customers, their problem, and the product they were providing. They went as far as to declare in 2007 that Kodak 'was not going to play grab ass anymore' with digital photography. The rest is history.

Markets change. Technology changes. Customers change. Problems change. Consider your customer–problem–solution to be something static or given, and prepare yourself for the ride—downwards.

In order to improve how your teams understand your customer-problem-solution statements, there are several activities and tools you can use:

– **Impact Mapping**[11] is a tool that is becoming quite popular within the Agile community. Designed by Gojko Adzic as a strategic planning technique, Impact Mapping starts with the 'why' of whatever the project, product, service, or initiative you are analyzing—in other words, it starts with vision and purpose. Once we are able to describe the purpose and make assumptions about our value proposition, it helps us determine who can help us achieve our goals (or stop us from doing so), what can they do to help/stop us, how they can actually help/stop us, and what we can do about it.

– Impact Mapping is partially based on the Swedish usability company InUse's **effect maps**,[12] which also start with business goals and work them into features and activities by first determining user groups/personas and their specific goals.

[10] Barabba V (2011) *The Decision Loom: A Design for Interactive Decision-Making in Organizations*. Triarchy Press.

[11] Adzic G (2012) *Impact Mapping: Making a Big Impact with Software Products and Projects*. Provoking Thoughts.

[12] http://www.slideshare.net/Jonas_inUse/effect-mapping-a-better-way-to-get-really-usable-results-out-of-it-projects

– Jeff Patton's **Story Mapping**[13] is also a great way to learn more about your customers, provided that they participate in the Story Mapping session. Through Story Maps, development teams can learn more about the proposed journey of their customers while using the product and can engage in conversations about value, priorities, and customer needs.

– **User journeys** and **user personas** are well-known techniques for the user experience (UX) community. Popularized by UX expert Alan Cooper,[14] user personas are fictional characters used to understand the archetypes, market segments, demographics, and particularities of our different customers—as opposed to the 'the customer' statement that assumes that all users or buyers of our product will behave the same way. Having your team build user personas and, later on, interviewing real customers that should match our persona assumptions is a great way of generating insights into how to better solve our customers' needs.

– **Empathy Maps**[15] are also popular in the Agile and Lean Startup communities. Described by Dave Gray and others in *Gamestorming*[16] and later included in Alex Osterwalder's *Business Model Generation*,[17] this technique looks for a deeper understanding of stakeholders in the business ecosystem by analyzing what they think, feel, say, and do.

– The **Inception Deck**[18] is being used by more and more teams as great way of project or product chartering before going into 'full development mode'. Introduced by Jonathan Rasmusson in *The Agile Samurai*,[19] it gathers a set of tools and techniques aimed at creating better awareness of the goals of the product, the purpose, who the customers are, what the product is and is not about, and so on.

As you see, all of these frameworks, tools, and techniques are basically trying to answer the customer–problem–solution statement in a way that enhances awareness and understanding, not just by the product owner/

[13] http://www.agileproductdesign.com/presentations/user_story_mapping/

[14] Cooper A (1999) *The Inmates are Running the Asylum: Why High Tech Products Drive Us Crazy and How to Restore the Sanity*. SAMS – Pearson Education.

[15] http://www.gogamestorm.com/?p=42

[16] Gray D, Brown S, Macanufo J (2010) *Gamestorming: A Playbook for Innovators, Rulebreakers, and Changemakers*. O'Reilly Media.

[17] Osterwalder A (2010) *Business Model Generation: A Handbook for Visionaries, Game Changers, and Challengers*. Wiley.

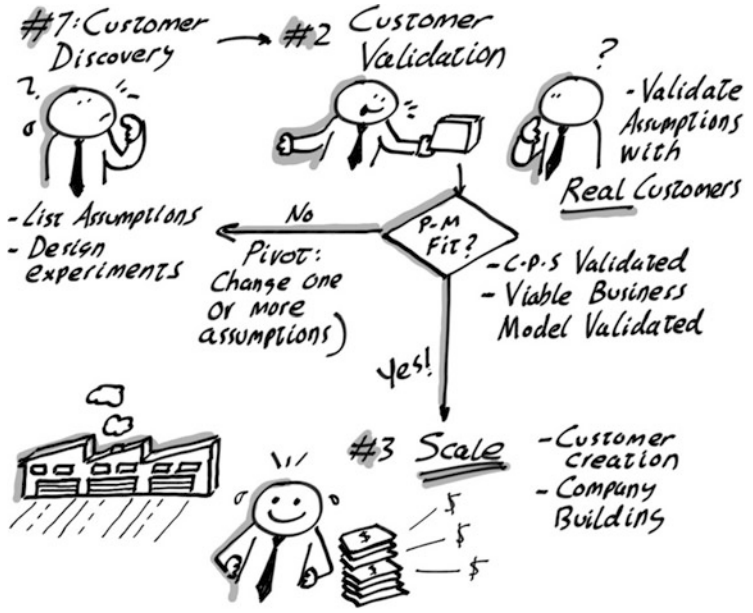
[18] http://agilewarrior.wordpress.com/2010/11/06/the-agile-inception-deck/

[19] Rasmusson J (2010) *The Agile Samurai: How Agile Masters Deliver Great Software*. Pragmatic Bookshelf.

manager/master, but from everyone in the company. No matter what your ‘cup of tea’ when it comes to techniques, just make sure these reflections are made, and that they are followed up and reviewed periodically.

Sustainable Business Models

As I mentioned before, the lean startup framework combines agile software development cycles with customer development cycles. Customer Development, as described by Steve Blank in *The Four Steps to Epiphany*,[20] consists of identifying your business or product assumptions and then validating those assumptions by conducting a series of experiments that may or may not take the form of a ‘Minimum Viable Product’. If assumptions are proven wrong, a ‘pivot’—changing one part of your business plan while maintaining the rest—must be performed until a valid customer–problem–solution has been validated. When all assumptions are proven right, you have a ‘Product–Market Fit’ and can focus on building and scaling a company around your now proven business concept. As remarked by both Ries and Blank, the main focus in the first stage is not to launch a product, but to obtain as much validated learning as possible.



[20] Blank S (2005) *The Four Steps to the Epiphany: Successful Strategies for Products that Win*. Cafepress.com.

The advice to stress is not to invest up-front on business plans if you haven't proved the assumptions correct. But even if you have proven that there is a customer who has a real problem and we have a validated product that solves that problem, we still have not proven that the customer is willing to pay for it.

Here is where the validation part of customer development gets tricky. Usually, entrepreneurs are so in love with their product that they will take any small polite compliment as a validation of their whole business model. But it is one thing if a potential customer says 'Oh, I like it' and a very different thing obtaining any currency from him—in the form of money, time, referrals, or any other kind of deliberate investment and effort from your customer's side in order to obtain your product or service.

Even when you have found some clients willing to offer some currency for your product or service, that is not a full product–market fit: you also need to prove that there is a sustainable business model for the product. A sustainable business model includes costs structures, revenue streams, sales channels, market funnel, customer segmentation, and demographics. In other words, a couple of handy customers do not justify a huge investment to create and grow a company: you have to be ready to prove that there is a business case behind the investment. As my good friend Alejandro Barrera likes to say, 'Your mother is not a market'.

A very popular framework for defining business models and their core assumptions was provided by Alexander Osterwalder and Yves Pigneur in their book *Business Model Generation*.[21] Their Business Model Generation Canvas has been adapted by others to create similar tools, most notably Ash Maurya's Lean Canvas. The idea behind all of these tools is always be able to define all your business model's core assumptions and use that assumption list to define experiments that will validate or invalidate them following a customer development cycle.

So, how do we use this framework to improve our product or service if we are not Startups or entrepreneurs?

Once again, it's a matter of understanding the 'why' behind the product: our vision, the customer, the problem, and how are we providing value

[21] Osterwalder A, Pigneur Y (2010) *Business Model Generation: A Handbook for Visionaries, Game Changers, and Challengers.* Wiley.

through our product or service. But, in the same sense that just focusing on profits or growth might eventually kill our company, neglecting the business side of our product or service can be equally deadly or even worse.

Being able to provide a viable business plan for a project should be an enforced practice in Agile and Lean companies. In previous chapters we defined backlog as waste and remarked on the need to have efficient project/workload funnels that are able to maximize the amount of value produced, given that there is a limited capacity and a potentially endless choice of projects or products we could promote. A business proposition in the form of a lightweight business model could help the company to establish effective prioritization frameworks.

A second reason to encourage the definition of business models for products, services, or projects is to maximize learning. Several times I have assisted in a project's kick-off where some business goals and a definition of success were crafted, but at a later time, when the project was ending, all conversations featured costs, deadlines, and charging the customer as soon as possible—the business goals faded away and nobody cared about them anymore. Did we actually meet our goals? Does the client agree? Nobody cares.

Success, I believe, is not only being able to deliver more features faster: it implies being able to satisfactorily solve a customer's need, and to do it in a way that creates value for everyone in the business ecosystem—stakeholders, employees, managers, client, suppliers, and community. Thus, being able to compare our actual results with the business assumptions, goals, or success criteria we defined at the beginning should be a mandatory practice. This practice is not just for Agile or Lean companies, of course, but it is especially important for them, because there is huge waste in repeating the same mistakes over and over.

Obtaining Client Feedback: MVPs and MVEs

As just described, everything in your product or service plan, from customer description to revenue statements, should be considered an assumption until proven right through a real-customer experiment. Sometimes these experiments take the form of customer interviews, focus groups, surveys, or some other way to obtain information from real customers. But from the Lean Startup perspective, this information might be misleading if there is no currency required from your test customers in the form of money, attention, time, or any other kind of resources that your test customer is willing to spend in order to know more about your product or even to obtain it.

To maximize the chances of obtaining customer currency and, hence, validated learning, Minimum Viable Products (MVP) are a key tool for Lean Startup practitioners. According to my own experience, this is one of the least understood practices in Lean Startup, and it might have something to do with the chosen name.

> *MVP, despite the name, is not about creating minimal products. If your goal is simply to scratch a clear itch or build something for a quick flip, you really don't need the MVP. In fact, MVP is quite annoying, because it imposes extra overhead. We have to manage to learn something from our first product iteration. In a lot of cases, this requires a lot of energy invested in talking to customers or metrics and analytics.*
>
> – Eric Ries[22]

In my humble opinion, 'Minimum Viable *Experiment*' (MVE) makes more sense. It better describes the idea of validating our assumptions and learning about our customer, his problem, the solution, and the viable business model that we can scale later.

There are several ways you can perform this kind of experiment. As said, a usual way is to just ask your customers and learn from their opinions. But it's easy to say 'Oh, yes, I would definitely buy such a product', and it is a very different thing is to sign a check. The closer to the real experience of buying or using the product, the better, but always keeping in mind that MVEs should be performed in very fast cycles—sometimes several times *a day*—and that they should be as inexpensive as possible.

Some very popular ways of performing MVPs/MVEs include:

– **Flintstoning**: the name, of course, makes reference to the Flintstones cartoon, where regular modern machine-based tasks were performed by animals instead. The idea is that you start delivering your service or product by hand, and later on, when a product–market fit has been achieved, you invest in automating the service delivery or product

[22] http://www.startuplessonslearned.com/2009/08/minimum-viable-product-guide.html

manufacturing. Some variations of flintstoning include the 'concierge MVP', where some kind of personal assistant performs the service for you before we are able to automate it; or the subcontracting/freelancing MVP, where we forward your requests (or part of them) to some external suppliers until we are able to perform them by ourselves.

– **A/B Testing**: this is an experiment where you present two versions of your product or service to two separate but market-similar customer groups. Then, you test their responses to it so you can see which of the two versions is more successful. Some similar approaches include ghetto testing, a small test performed on a small set of users; smoke tests, meaning fake advertising with no real product behind that we use to see just how many people are tempted by the product idea; or product batching, where different sets of products are issued with different conditions of price or contracting requirements to study customer acceptance and response.

– **Low-Fi Prototyping**: consists in the use of product mock-ups, usually in the form of hand-drawn sketches, wireframes, cardboard models, or similar concepts. These prototypes might sometimes be hi-fi: for instance, many mobile application developers start by crafting very detailed images of the application look-and-feel with no real functionality behind it just to get feedback from their customers.

– **Storyboarding**: a storyboard is created to show the customer how the product works or how he or she interacts with it. It might take the form of 'user journeys', describing the interaction of the customer with the product throughout the whole product cycle. Some variations over this kind of experiment include product brochures, slide-based presentations, flash videos, Story Maps, or some other kinds of simulation.

There are endless other ways of performing minimum experiments, including computer simulation, free samples, or mailing lists. My goal is not to create an exhaustive MVP/MVE guide, but to make you understand the key factors behind them: minimum, fast, inexpensive, and aimed at learning not launching.

Agile, in fact, has many times been closely related to the 'release early, release often' approach. Popularized by Eric S. Raymond in *The Cathedral and the Bazaar*,[23] and widely adopted, especially by the Open Source

[23] http://www.catb.org/~esr/writings/cathedral-bazaar/cathedral-bazaar/

community, this approach might absolutely be labeled as 'Agile' in the sense that it enforces rapid, continuous, and evolutionary software delivery. Keep in mind that, in order to perform Agile or Lean Startup frameworks, there is no need to 'release'. In the same sense, MVEs do not mean releasing your product to the full market—instead they are about just being able to show something to your client as soon as possible in order to maximize the amount of validated learning.

Again, this is not only useful for Startups launching new products. Before investing six team-months in your next feature set or client request, you might want to invest a week or two in validating your assumptions of how this effort is going to provide value and solve a problem! If you prove some of your assumptions wrong, you will have the chance to pivot—change the failing assumptions, then perform new learning experiments.

Traditional managers might argue that working on experiments, models, sketches, and mock-ups that may be discarded later on is a huge waste. Of course, if you were *absolutely* sure of what is to be built, you might as well go for a full waterfall development. But keep in mind that the whole point behind Agile is that we realize that changes will come over the development process—and we don't know which changes will come, or when. Hence, the sooner we are able to provide some vision of the *real* product or service that we will deliver, the sooner we can see if we are on the right track or, if not, the easier it will be to steer the project to a new course.

As for Agile development, 'walking skeleton' and evolutionary development—sometimes called 'iterative and incremental' development[24]—have always been implied, although not specifically mentioned in the Agile Manifesto. This can be seen as a way of producing an MVP—and in this case, 'Product' is an accurate definition as we are actually showing something to our client that will later on evolve to the full product. To improve your Agile product management process, always make sure that the design is aimed at producing an end-to-end MVP as soon as possible!

[24] I have used the term 'iterative and incremental' for a long time, yet I feel that phase-gate development can be called 'iterative and incremental' and is not Agile in the sense that nothing is ready until the last phase is built. Hence, I currently prefer to use the term 'evolutionary' in the sense that we start with a minimum end-to-end prototype and we evolve it, thus we are able to launch something at the moment the client calls it 'done enough', even if there are still features to be added.

Lean Analytics

A hot topic on the Lean Startup community is Lean analytics—up to the point that many people confuse both terms and think that Lean Startup *is all about using the right metrics*, which of course it is not.

The idea behind Lean analytics is to be able to set the right amount of metrics—no more, no less—that drive your product management investment in order to produce the maximum results. These metrics should be included in product definition as goals to be reached, and any development effort should be related to an attempt to influence one of our key metrics or key performance indicators (KPI).

In order to improve your product management process, defining the right metrics is absolutely a key success factor. Too many companies get lured by 'vanity metrics'—those that might make your company look good, but will not help you decide what to build next. For example, many web-based companies like to show their 'page hits' metrics because they tend to go up every month, which make them feel good, but they seem to assume that more visitors or more traffic means more profit, which is not always the case. It's like having a store and saying 'Oh, I'm so happy that so many people are able to see my window display'—but then nobody enters the shop or buys anything from it!

Improving your market funnel is one of the single more important improvements you might achieve in order to make your company more successful, but to improve your market funnel your metrics should hurt. To define the best possible metrics, your company should start by defining the market funnel—how customers reach your company and end up buying your product—and define your key funnel-related strategy right now: are we trying to get more leads? Do we want to improve sales? Are we interested in getting more repeat buyers?

There are several ways to define a Market Funnel, and the definition you use might depend on your specific product, market, or business model. As an example, I often use Dave McLure's AARRR funnel[25]:

– **Acquisition**: the client learns about our product. Metrics to influence acquisition might include customer's cost of acquisition or efficiency rates of different sales/advertising channels.

[25] http://www.slideshare.net/dmc500hats/startup-metrics-for-pirates-long-version

- **Activation**: the client performs some kind of activity regarding our product or service. Maybe he gives us his e-mail, creates an account, or he just browses through our store trying some products. Key metrics might include time spent testing our products, customer interest and preliminary evaluation of our product, bounce rate (customers who try our product and reject it), or percentage of customers that complete their account activation.
- **Retention**: the client comes back over time and keeps interacting with the product. You might measure visits over time, frequency, lifetime of our customers, or 'stickiness' of our product.
- **Referral**: our customer recommends our products to other customers. Usual metrics include the Net Promoter Score—how likely your customers are to recommend your product on a scale from 0 to 10—a virality index, or customer overall satisfaction.
- **Revenue**: your customer expends some money or some other kind of resource on your service or product. Some related metrics are the Average Revenue Per User (ARPU), the Lifetime Value (LTV), or the Monthly Recurrent Revenue (MRR).

Several activities can be performed in order to improve your market funnel, but investing a lot in revenue creation when your problem is that you don't acquire enough customers or they don't buy from you more than once or twice might be a terrible mistake. You have to find the biggest bottleneck in your market funnel and work there, in the same way we explained when we described process bottlenecks and the theory of constraints.

Of course, for some marketing experts this description might seem too simplistic—and it probably is. The point here is to make development teams and everyone involved in product management aware of the importance of this metric, the strategic goal of the company, and to create alignment on all development efforts.

As an example, many Agile/Lean Startup teams are starting to craft their user stories including metrics to measure success and goal achievement. Instead of the well-known user story template, 'As an X, I want Y, in order to Z', some teams are using a new template in the form:

We assume that customers of type X have problem Z that we can solve through Y; we will know that we are right when qualitative metric M goes (up/down) and/or quantitative metric N goes (up/down), which will contribute to key performance indicator K.

A wrong set of metrics or focusing too much on metrics and forgetting that they are used to pursue some business goal can be a huge impediment, up to the point of putting your whole company at risk. One of the main risks is that people will start pursuing a metric and run into 'moral hazards'—creating some damages or running into some risks that they don't care about because these risks fall into some other part of the system's responsibility. This also happens when we assume direct correlation between the metric and the results—like when you assume that any improvement in the metric is actually a sign of the desired result.

Suppose, for instance, that you are managing a call center and you assume that a lower call resolution time means that your customers are happier. You set a call resolution time goal of 'under five minutes average' and ask your team to pursue that metric—you might even want to throw some incentives for those who make the goal, just to make the situation worse. It could then happen that your teams start to hang calls at 4 min, 59 s in order to improve the metric, and the result would be lower customer satisfaction—just the opposite of what you actually wanted.

So my final advice when it comes to metrics and Lean Analytics is not to rely too much on metrics. Metrics are fine as learning tools, but they should not replace the actual goals we are aiming at.

Value-Driven Prioritization

The place where product meets process is the product backlog. Once the backlog is created, it can feed the development process and we can focus on being faster, smarter, and more efficient. As I have remarked during this whole chapter, just building 'whatever' won't give your company the best chances to succeed, no matter how quickly and efficiently you build it; you need to build the right thing. This 'right thing to build' is defined in terms of the most value delivered to the customer, and in order to be certain about what customer value is about, we should perform some kind of experiments to obtain as much validated learning as possible within our project's constraints.

Assuming we are able to define several features that constitute a viable product that makes a good product–market fit, just stuffing all the features in the backlog by alphabetical order would not make much sense. Still, many product managers see the feature set as a whole, so they don't care too much about prioritization—*every* item in the backlog must be delivered, so why should I care about the order in which the team decides to perform them?

Of course, if *everything* is a priority, then nothing is *really* a priority. This all-or-nothing approach sets tight limits to the Agility of both the team and the company. I could even argue whether that's Agility at all, as it does not maximize value from the customer perspective, nor does it give him the ability to have the most valuable parts of his product available early for his competitive advantage.

Our highest priority is to satisfy the customer through ***early*** *and* ***continuous delivery*** *of* ***valuable*** software.
– Principles Behind the Agile Manigesto

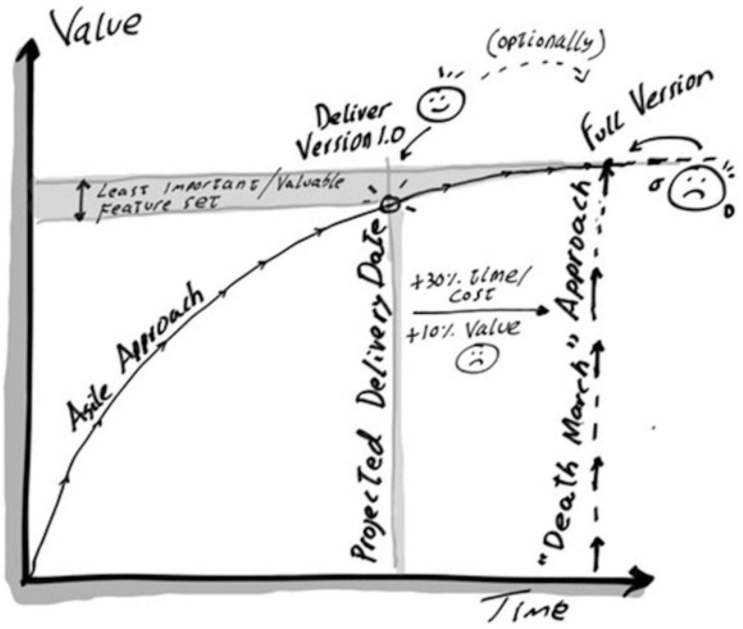
All-or-nothing backlogs lead to 'death march' projects: imagine you decide to cross a desert, and you estimate that you need 5 days to do so. You also need three liters of water a day to survive. So you get 15 l of water, you start marching, and on day 5 you are in the middle of nowhere with no water left. Now, you know that behind you there are 5 days of desert—you won't survive that. So your only chance is to go on walking and pray that the end of the desert is just behind the next dune.

The same happens when you set a comprehensive list of features with no way to be satisfied with some minimum set of them: you might run out of time and money and find yourself just 80 % done.[26]

Value-driven prioritization is at the heart of both Agile and Lean—and, by extension, Lean Startup. Agile/Lean teams should engage the backlog with a 'minimum marketable feature set' mindset—as opposed to the 'minimum viable product/experiment' discussed before. The idea is to be able to launch something to the market, no matter how shamefully simple. This dramatically reduces risk and gives you a chance—at least a chance—to cut the project short and launch earlier if there is more value on this. Of course, you can release a full version later, or just decide that the benefit of

[26] As a personal rule of thumb, when your project managers say that project is 80 % done, consider that you still have half-way to go.

that version is not worth the extra investment and engage in some other more profitable project.



Value-driven prioritization sounds easier than it actually is, but, no matter how difficult, it is always better to fail trying than to not use any real prioritization at all. On the other hand, as explained when describing the problem with the visionary, just relying on one person's opinion on what's more important or urgent might be a source of bad business decisions, even when that person is the customer representative. Team-based prioritization, for instance, can be subject to internal politics and influence games if there are no clear, explicit definitions of what should we consider a priority.

Agile companies should define and improve a prioritization framework to be used when including new items or modifying the product backlog in any form. There are several existing criteria that can be used to define such a framework, including risk versus benefit, opportunity/profit versus cost/effort, urgency versus importance, or user satisfaction with current solution versus problem importance. You can even move into user experience criteria, goal scoring, theme screening, Kano model, cost of delay, and more advanced frameworks, which are outside the scope of this book. The important thing is that you establish a common standard to determine priorities based on value, and then enforce the use of the framework when managing the backlog. If there are exceptions to be made to the framework, for example, if an urgent request comes from our key stakeholder, those exceptions should also be contemplated in the prioritization framework so people don't feel like the framework has been bypassed.

Other than maximizing value, your prioritization framework should provide a rational link between the development process, your product roadmap, and your company strategy toward improving market success.

Code Kaizen

Despite my background, I tried not to make this book too software-centric, but when it comes to product, quality, and Kaizen, I feel like I need to drop a couple of lines about Kaizen and software. If you are not in a software environment, I would still advise you to read the next section: I promise it will not melt your brain and you might actually get a couple of ideas on how to bring product improvement to the production line.

My desire to talk about software and Kaizen comes from the fact that there are several engineering practices that have emerged from the Agile ecosystem that contribute to creating better products, enhancing their quality, and also bringing an increased Kaizen mindset to development teams. Probably the most popular ones are some of the eXtreme Programming (XP) practices:

– **Pair programming**: people work in pairs: one person performs the task, and the other observes and discusses with the performer. This increases awareness, focus, and knowledge sharing.
– **Test-driven development**: before writing any code, we first write the acceptance criteria—how will we know that the code is working and that we have finished? Once we have the tests, we can start working on the actual product. This creates a 'testing scaffolding' that helps to debug the whole application, increases the programmers' understanding of what is needed, and also helps the programmers stay focused on required functionality, without adding any additional unwanted features that are not covered by tests.
– **Refactoring**: once the code is working, we spend some additional time 'tidying up' the code and making it smaller, simpler, more flexible, more robust, and more secure.

Good Agile teams are also known for the use of deliberate practice—not just repeating the same work over and over, but trying to make it better and learn every time—and for dealing with technical debt—revisiting legacy code or buggy parts of their product to improve it, even if it was accepted some time ago.

I once asked myself: is there any way we can move these practices out of software development? It seems so. The Wikispeed team created an eXtreme Manufacturing framework[27] for car prototypes manufacturing that includes all these practices. The result: they are able to produce a new prototype every 2–4 weeks, compared to the 4–7 years of the car industry average!

Even if you don't want to start with such an extreme implementation of XP practices, I've had plenty of success introducing pair working to writers, public speakers, trainers, or marketing professionals. I've also managed to use 'test-driven' product definitions with product management teams and have used a sort of refactoring process to company strategy at C-level workshops. My point: don't get distracted by the 'software' or 'car manufacturing' parts; take a look at Lean, Agile, or Lean Startup production-level practices and try to import them to your own environment.

Summary

Even with a perfect team and process, your company needs to find a good product–market fit to be successful. Achieving a product–market fit starts with the definition of your corporate vision—what is your company about, what is the higher purpose it pursues, and how does that provide value to your customers? The vision is not a matter of one person's view; it's usually a collaborative challenge that shapes the fabric of your company and its culture.

From the vision, you can start iterating your assumptions on who your clients are, what their problems are, and how you are going to solve it. Each iteration should consist of a definition of all assumptions, an experiment design, and conducting these experiments with real customers. The experiments provide validated learning about your assumptions. If proven wrong, you should change the failed assumptions and conduct another learning cycle.

Customer–Problem–Solution (C–P–S) might be your core assumption, but in order to achieve a product–market fit you should also validate your whole business model around the C–P–S statement. Customer collaboration

[27] http://wikispeed.org/2012/06/extreme-manufacturing-in-5-minutes/

is essential to define and validate your assumptions. This validation should be conducted before any development investment starts.

Validation through experiments might take the form of a Minimum Viable Experiment (MVE) or, in Lean Startup literature, Minimum Viable Product (MVP). The priority during MVE/MVP definition and development is not to be able to launch a product, but instead to obtain as much validated learning as possible in early product definition stages.

Product definition should also be a joint venture between you and your client. Collaboration frameworks should enforce customer involvement in the earliest stages of the design and development process. During this definition stage, key goals, performance indicators, and metrics should be identified, understood by all stakeholders, and built into the product definition.

Value-driven prioritization is key to delivering the most value as soon as possible. In order to avoid 'death march' situations, Agile projects and products should focus on delivering as much value as possible in the earliest stages of the project. A 'walking skeleton' or minimum viable feature set including all basic features to provide an end-to-end functionality—no matter how simple, ugly, or even shameful—might be the first focus of Agile development teams in order to validate the whole product scheme.

Things to Try

– Ask your teams to silently write what they think describes your company's vision—what's the company about. You might be surprised at the different results. Engage in debate with them about the different views and the canonical company-defined vision: where is the gap? How are we failing to communicate our vision? Is there something crucial we are not including in the vision? Or is there something we should eradicate from it?

– You might want to repeat the same exercise with some of your closest clients and suppliers for enhanced insights into your company's value and how people perceive it.

– Ask your product managers to create a product management team. They might not like it if they feel like their power is being questioned, so you might ask them to define a product management community of practice instead—and to be the community managers for it. The goal of the product management team (or community of practice) should be defined by the team, but you can borrow plenty of ideas for the different duties around product management from this chapter.

– Ask every team involved in project, product, or service delivery to state their core assumptions on who their customer is, what problem they are solving, and how are they solving it. Review those statements both with the teams and with their clients—or even better, ask your teams to perform experiments in order to validate those assumptions. Make sure these definitions stay updated through frequent follow-up. Use this definition as a team compass when discussing priorities, feature definitions, alternative solutions, etc.
– Assess your current customer collaboration level satisfaction both in your teams and with the customer. Ask your teams to develop a framework for customer collaboration and support them in the discussion of the framework with management and customers. Once the framework is approved, make it part of your value proposition and contract agreements.
– Start a customer evangelizing program. It can take many forms: informal breakfasts, post-work training sessions, hands-on workshops, presentations at customer facilities, or newsletters. The goal is to make your customer aware of your way of working and how Agile and Lean are helping others to obtain the maximum value out of their investment.
– Include a simple business case or business model definition as part of your project charters or—more important—new product definitions. Use the Business Model Generation Canvas or the Lean Canvas to inspire you, but make sure to customize them to your own needs.
– Engage your customer in early project/product definition through the use of the different techniques defined in this chapter and with the games/activities proposed in the second part of this book.
– Support your teams in understanding your company's market funnel and making suggestions on how to improve it. Define the key metrics and make sure that your teams, especially the product management team, include them as a part of their design.
– Define and communicate a prioritization framework. Search online for the different terms listed in this chapter's 'Value-Driven Prioritization' section (Kano model, cost of delay, goal scoring, theme screening) and learn more about them.
– As part of your prioritization efforts, ask your customer to define sets of features that might constitute a minimum, fair, and complete version of

your product. Story Maps—described in the games section of this book—are a very good way to address this kind of collaboration. Dot voting or 'buy me a feature' games, where the client has a limited amount of resources that he is forced to invest on features considering the trade-offs, are also good activities to manage customer expectations and generate understanding of the possibilities of the development effort.

7 Epilogue: Implementing the Framework for Agile Kaizen

I am pretty aware that actually implementing the whole improvement framework that I proposed in all the preceding chapters (events, team, process, and product) is probably its main weakness. When I designed it, I had to make a decision: base it on guiding principles and core values, then highlight some key practices and tools; or, on the other hand, try to create an 'algorithmic' framework: wash, rinse, repeat.

In my opinion, frameworks like Scrum for Agile product development or David Allen's Getting Things Done (GTD)[1] for time management are very successful and widely adopted in a large part because they provide a simple algorithmic guide to follow, especially in the beginning, during the *Shu*[2] stage.

Using martial arts terminology, these frameworks are in fact *kata*: basic positions and movements you repeat over and over to increase your skill and domain of the techniques before you are actually able to use them in real combat. Of course, no amount of *kata* really prepares you for your first combat. It would be like saying that lots of calligraphy will make you a successful novelist.

In the same sense, algorithmic frameworks are designed to improve your discipline and provide some guidelines. Over time, just following the algorithm will fall short when it comes to actually improving your environment; you need real-life combat—actually fighting your impediments and

[1] Allen D (2002) *Getting Things Done: The Art of Stress-Free Productivity*. Penguin Books.

[2] See Chap. 5 for the definition of Shu-Ha-Ri.

developing new, better ways of delivering value. Another metaphor I like to use is that nobody learns to swim in a book; you have to dive in, and the first time it probably won't be nice and you'll swallow a lot of water.

Algorithms are fine for some simple, repetitive situations. There's no one-two-three step guide for things like successful marriage, inner peace, empathy, happiness, or managing a business. In my humble opinion, growing the right mindset will serve you much better in those situations than following any kind of process or recipe. By the way, I would advise you to be careful with 'snake oil sellers' that promise any kind of easy way to deal with these complex situations.

The question remains: how to implement the framework? Where to start? As I review the chapters and try to figure out if there is a logical sequence, I once again conclude that, in Kaizen, everything is happening at the same time. As an example, if you hold the idea that production comes first, and inspection comes second, you are basically falling into Taylor's model. Lean urges you to build quality into the production line, in every part of the process. Quality is not limited to defect detection at the end of the line, where corrections are expensive and harmful: it enforces the idea that every production cell must pass perfect quality to the next cell.

I do understand that trying to manage roles, events, people, process, and product at the same time can be overwhelming. Nobody said it would be easy. In my Aikido practice, I learned that perfect technique comes from balance, breath, connection, timing, accuracy, and whole body movement—all at once. You need *decades* to embody those principles and make them a natural part of your practice—if you ever reach that state—and there is no guarantee you will. I'm fine with that. In fact, I enjoy it as long as I feel that every year I'm better than I was a year before.

My final advice is to focus on mindset and discipline, and be uncompromising with them. Even when the company knows that there is an impediment, a defect, a bottleneck, and nobody cares about it, even when you feel you are not able to do anything, just keep feeling uncomfortable with the fact. Keep pointing at it. Keep complaining, and keep asking people to debate about the uncomfortable facts. Keep asking if this is the best we can do, if we consider ourselves perfect, and if we are in fact perfect—*hint: you are not.*

I do believe that if companies grow a Kaizen culture, Kaizen will naturally follow. But of course, a Kaizen culture is a product of Kaizen itself, so this might be a ‘which came first, the chicken or the egg’ situation. Managers—Lean managers—are of course a key factor to Kaizen culture. Leadership, role modeling, teaching, supporting, coaching, and acknowledging the right behaviors is probably the most successful way to improve the culture.

Some people even state that culture emerges from middle management—front-line workers being dependent and subject to them, and top-management being sometimes too disconnected from real day-to-day company culture and operation. I personally can relate to the idea, but I also believe that everyone has his or her part on defining and changing a culture. Middle managers can be heavily influenced by top managers, and, in fact, top managers have the responsibility of hiring and promoting middle managers who represent, foster, and enhance the right culture. Front-line workers can also influence and inspire both their managers and their colleagues.

Once again, grow the right mindset, reward the right behaviors (not results). This book, hopefully, can serve as a handbook and a constant reference of activities, games, tools, practices, and, most important, values and principles. But at the end, Kaizen emerges from people who are paradoxically never happy with the way things are, but are also convinced that there is nothing impossible and we can always do better.

Part II

Retrospective Activities and Games

About the Retrospective Activities

Retrospectives that always use the same technique of analysis may become boring; therefore, introduce various techniques over time.
– Bas Vodde, Craig Larman, Pete Deemer, Gabrielle Benefied, *Scrum Primer*

Welcome! In this second part of the book you will find 27 different activities to introduce fun and reflection at your Retrospectives—that's more than one every 2 weeks for a full year of different and exciting Retrospectives. And if that is not enough, there are 12 bonus proposals in order to give you even more ideas on how to bring new insights to your Retrospectives.

Some of these have been proposed, used, or published by Agile coaches all over the world. During the research process, if I found one that I could attribute to someone without any doubt, I took it out of the list—I prefer the authors to take all the credit for them. That does not mean that I created all the activities that you will read in this part of the book; many of them are just popular tools and games that thousands of Agile teams are using around the world. I apologize if I was unable to trace some of them, and will try to give public credit to the authors if I find out in the future.

I'm the first person to find some Retrospective activities that I have been able to study childish and unprofessional. When you see a bunch of grown-ups playing with balls for an hour, drawing tress, hugging, singing, yelling, and when asked for a debrief of the exercise, all you get is 'Oh, we had a great time and learned to play as a team'. Well, let's just say I don't like it. Some of the activities I propose in this book may look playful and childish to some of you—I just ask you to keep an open mind. I've used them all, and with good results, before proposing them as part of my Retrospective toolkit. I may need some trust from you for this, at least the first time you use them. Keep in mind that this kind of activity gets better over time, when you are able to get more insights and clues out of the (necessary) debriefings at the end of the exercises.

Some of these activities might even sound strange; that's because they have been designed to help the team think 'outside the box'. When you study the creative process, you find that there are two mutually excluding mind modes—expansive, where you think laterally and freely, and contractive, where you go into analytical decomposition of ideas and focus on constraints, boundaries, rules, and what's possible. Again, some of the proposed activities are designed to break the regular thinking process and help the team get new insights through creativity and innovation.

Of course, I invite everyone to tweak, tune, and modify these activities to fit their needs. There is no right or wrong in thinking, and there are no 'Kaizen Police' who will fine you if you decide to run an exercise for a longer time or combine a couple of them. Innovation is usually achieved by recombination of existing ideas, so feel free to innovate.

Just a final word—tools are just tools, and games are just games. I did not write the first part of the book to have you jumping over it and running to the games! Games are not going to change your culture; they are just going to spice up your Retrospectives and can help you run Kaizen events or bring some Kaizen spirit to your facilitation toolkit. But remember that, above all, you are building a culture, and cultures are not built through games and tools.

8 Team Activities

Best and Worst

This is a storytelling exercise, aimed at defining the team identity through their shared values. According to my experience, just listing some values on the wall and dot-voting them produces artificial results. On the other hand, if you ask them to think about their best day at work (not necessarily in their current company) and their worst day at work (again, the same consideration), you can then engage in a productive discussion on what values you can identify in those stories.

The dynamic could be:

– Five minutes of personal introspection to think about stories. Optionally, tell them to craft a drawing or a sketch of the story, in which case I'd go for 10–20 min.
– Form groups of two. Each person tells his or her story to the other person. Then, they decide which of them is the best and which is the worse.
– You can now form groups of four and do the same.
– In a team of eight, with two story groups of four people each, we would have now two good stories and two bad ones. You can focus on those in order to gather values and describe them as things you want to enforce and things you want to eradicate from team's environment.
– In order to identify the values, you can prepare a set of questions, like, for example 'What are the characters involved? How would you describe their attitude in one word? Is there any fictional character you can relate to this story—like '*The Project Manager was behaving like Moby Dick's*

Captain Ahab'? What would the company look like if this kind of behavior happened more often?

– Keep in mind that there could be several different values and behaviors identified by a given story.
– For bad stories, instead of identifying unwanted behaviors, focus on turning them to positive—instead of saying 'the manager was tyrannical', we are searching for 'as a manager, you should seek your people's advice and make them participate in decisions' or 'you should foster collaboration and self organization'
– If you have more time, you can go for the rest of the stories.

It's a good idea to end the exercise crafting a Value Guide, with sentences in the form of 'This is what you do' (positive reinforcement) and avoiding the 'You don't do this' style (rules, negative, forbidding. . .). For a good example on how to craft a Value Guide, I recommend you check Netflix's presentation on corporate culture.[1]

WoW Manifesto

No, this is not about Blizzard's popular massive multiplayer online role-playing game, World of Warcraft (WoW). The 'Way of Working' (also WoW) is a set of ground rules designed to manage potential conflict in team's environment. Sometimes they are also referred to as team charters.

This is a good exercise to go deeper into team's identity. The idea is to start a WoW Manifesto that the team then hangs in a place that is visible to others, typically beside their Scrum/Kanban board or anywhere in their team workspace. During the exercise, the facilitator must engage the team in searching for rules, explicit or not, that they are already using when collaborating with each other.

A lack of rules, explicit or not, typically means that there's no real collaboration, as team collaboration needs some norms in order to avoid conflict. Maybe the team is just coordinated (you work on this, I'll work on that) or just cooperating (you work on this today, I will work on it tomorrow) instead of really collaborating (let's work together on this).

[1] http://www.slideshare.net/reed2001/culture-1798664

Some of the rules you could be searching for include:

- Time rules: schedules, arrival hours, punctuality, timelines
- Resources rules: use of printers, servers, budget, tools
- Dependencies rules: what to do when we need something from someone
- External interaction rules: how to manage external requests, interruptions, how to ask our customers for information
- Collaboration rules: how to divide the work, what strategies to enforce when pair-working, how to address conflict, how to reach consensus, telecommuting guidelines
- Process rules: mandatory processes to follow
- Information rules: documentation, knowledge sharing
- Environmental rules: office space, workspace, noise, even rules for air conditioning
- Cultural/behavioral rules: values, role models, ethical considerations, enforced behaviors

Just be careful not to turn this into a procedure handbook or some other form of corporate gibberish. I usually enforce the use of drawings, metaphors, and funny statements when describing the team's Way of Working. And of course, it must be visual and fit in a flip-chart size.

As discussed in the beginning of this book when we described Lean and Toyota Kaizen, the WoW should be maintained and updated just as any other team standard. Any solved conflict is an opportunity to create a new ground rule in order to solve it faster in the future.

The Listening Exercise

I must confess that this is one of my personal favorites.[2] It is an exercise designed to help the team communicate and listen. It's especially recommended for teams where meetings are turning violent, chaotic, too personal, or just noisy.

[2] I have to thank Certified Scrum Trainer Alan Cyment for showing me the first part of this exercise during Scrum Gathering Amsterdam 2010.

I usually run it in four rounds.

– Round one (2 min): people pair off. During the first minute, one person talks about the thing he or she likes the most in the world. During that time, the other person must do whatever he can think of in order to show that he *couldn't care less* about what the other person is talking about. Don't suggest strategies, but I usually watch people breaking eye contact, playing with their phones, looking at their watches, humming and puffing. . . When the minute passes, it's time to switch roles.
– Now engage in a conversation with them about how the exercise made them feel—frustrated, violent, and miserable. It is typical that some people cannot even finish 1 min talking—*about the thing they like the most in the world!* Make them think about that and hold that sensation to make them conscious of the terrible aggression of not listening.
– Round two (2–6 min): this is a similar exercise, but now the second person constantly interrupts, trying to make observations about themselves. The example I use is, person A says 'I love motorbikes and. . .' and person B immediately hijacks the conversation—'Oh, *me* too, *I* have two motorbikes that. . .' or 'Oh, *I* always say that motorbikes are a bad idea, because *I*. . .*I*. . .*Me*. . .*My*. . .'. Again, switch roles.
– Compare these sensations with the previous exercise. Some people like it more because now there *seems* to be a conversation. But of course, there is not. Person B does not really care about what person A is talking about.
– Round three (10–20 min): now form groups of three or four. One person explains a problem, and the other person tries to propose a solution. This is 'consultant/mentor' talk—person B considers himself smarter than person A and wants to show him available solutions for his problem. The third/fourth persons watch how the conversation takes place and, later on, comment to both persons A and B what they were able to identify, for example, when person B interrupts person A or when they feel like person A did not have the opportunity to go deeper into his problem. Ask everyone how they felt about this conversation mode.
– Round four (20–30 min): again, form groups of three to four people. The dynamic is similar—person A talks about a problem, and person B listens—but this time person B goes into 'coaching mode': there is not really a problem to solve, you are just curious about the situation and how person A feels about it, what the implications are, what has she already

tried, how close is she to a solution, what is the goal she is trying to achieve, what is helping her meet that goal, what impediments is she facing, and so on. Person B tries not to suggest or propose—'you could do X'—even with questions—'have you tried X?'. The observer later tells person B how well he performed in this role.

The team conclusions out of this exercise could be a great addition to the team WoW, in the form of guidelines, principles, and values to follow when listening to others.

Emotional Seismogram/Happiness Index

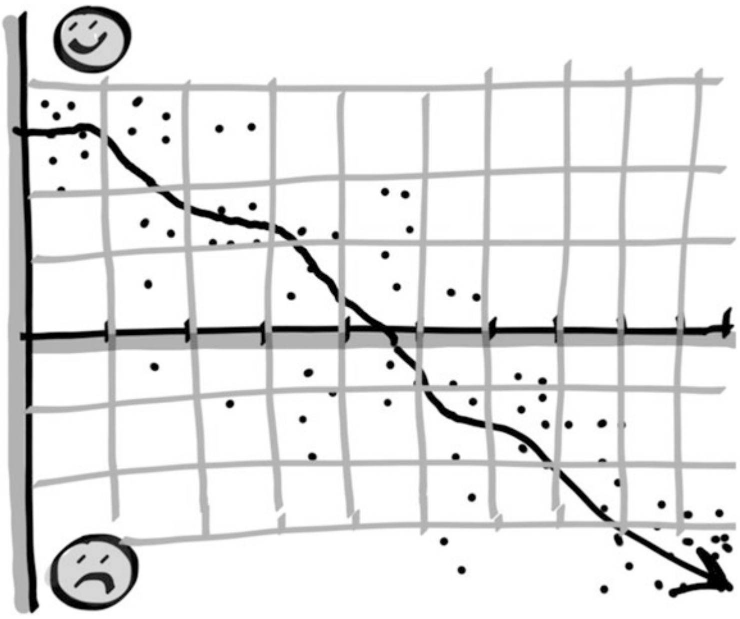
For some schools of thought out there, happiness is the Ultimate KPI (Key Performance Indicator). If you make everyone happy (stakeholders, managers, employees, customers, suppliers, community. . .) your company will be more reliable, and close to indestructible! Of course, you will be also be building a company that will attract more talent, better customers, and more investment.

Nevertheless, according to my personal experience, human resource departments fail to assess and manage corporate happiness in an efficient way. This exercise and the Motivation Radar—shown next—are ways to introduce the idea of corporate happiness and make visible how our behaviors affecting people's motivation and happiness.

The idea is to, during the iteration, build a tracking report of team happiness, mood or motivation, and later at the Retrospective discuss the changes and the noticeable events we are able to identify in the chart. Things that make the team happy or sad can then be identified and codified in team's value declaration and/or WoW.

Some usual ways of tracking happiness or mood are:

– Emotional seismogram: every day, each team member draws a dot on a chart declaring his current mood. An 'average mood' line is maintained by the team or their coach. Special events that affect team mood can be added to the chart in the form of sticky notes or hand-written notes.



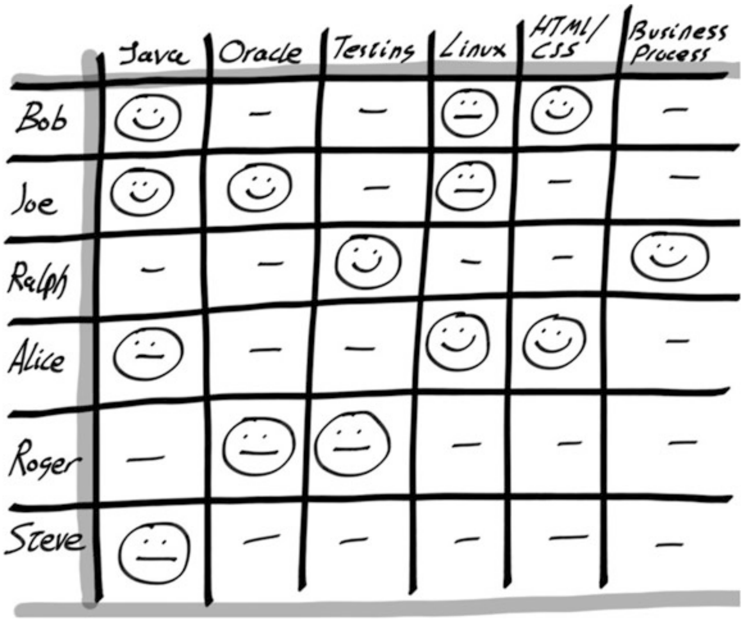
- Niko-Niko Calendar: similar to the seismogram, but in this case the display consists of a calendar with a smiley or frowny showing the team's mood every day. The team's average mood can be obtained every day asking at the daily meeting to give thumbs up, medium, or down according to personal moods.
- Happiness Index Chart: a spreadsheet where people update their current status, including columns to state what makes people happy, what makes people frustrated, or even suggestions on how to improve people's happiness.[3]

Hot and Not/Skills Market/Skills Matrix

I have seen several different ways to do a similar exercise (hence the title). Probably my favorite format is the Skills Matrix. The goal of this activity is to understand the team capabilities and design an improvement and learning strategy. In order to do so, first the team identifies all the skills needed to

[3] For some examples and further discussion, check Jeff Sutherland's blog post 'Happiness Metric – The Wave Of The Future' at http://scrum.jeffsutherland.com/2010/11/happiness-metric-wave-of-future.html

build an increment of a product. For a software team, this could include analysis, design, development, or testing activities—just think of your own value stream and the skills needed to implement it as a whole. Then everyone on the team scores himself or herself against those skills—good, fair, no skill. The results can be represented in a matrix:



	Java	Oracle	Testing	Linux	HTML/CSS	Business Process
Bob	🙂	–	–	😐	🙂	–
Joe	🙂	🙂	–	😐	–	–
Ralph	–	–	🙂	–	–	🙂
Alice	😐	–	–	🙂	🙂	–
Roger	–	😐	😐	–	–	–
Steve	😐	–	–	–	–	–

The first thing the team should look for is the ‘truck factor’—how many people from the team should be hit by a truck in order to severely affect the team’s ability to produce something. If the truck factor is ‘one’, that’s bad news—some knowledge sharing strategy should be enforced in order to distribute skills and make the team more robust and reliable.

In the example, we can see that Ralph is the only one who has knowledge of the business processes. If Ralph were missing, that would really affect the team. You can see that Ralph is a tester—maybe we could teach business processes to other testers, like Roger, and then Roger can teach some Oracle database management to Ralph.

The second thing would be to discuss learning strategy for the rest of the members. We can see that Roger and Steve are not ‘really good’ at anything—they are probably junior to others on the team or team newcomers. We should find ways to pair them with more experienced members. As Roger is already working with Ralph, maybe we could look for ways to pair Steve with other programmers like Bob or Joe—and so on.

Other ways of running similar exercises or fostering similar conversations are to play 'Hot and Not'—everyone creates a poster with two columns listing what are they good at and what they would like to improve, a good way of spotting hidden talents and interests—or running a Skills Market, where team members market themselves, creating brochures that describe their value proposition for the team.

Motivation Radar

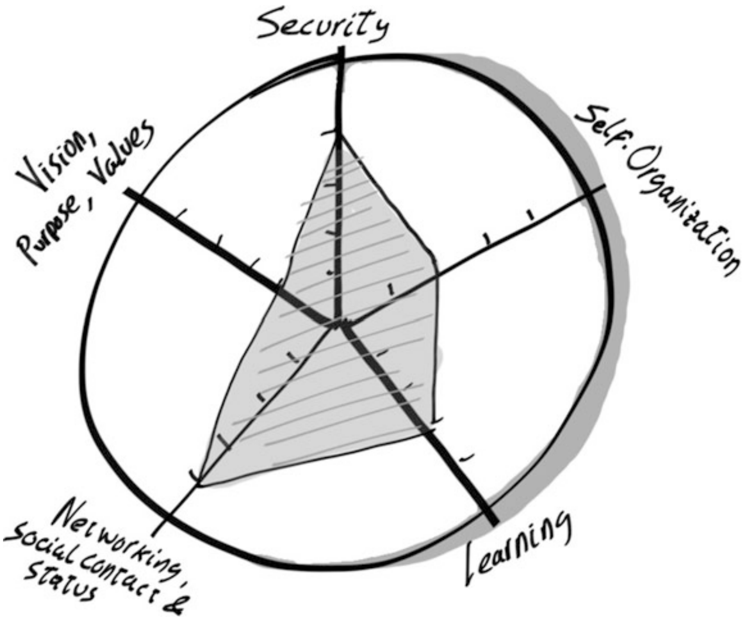
In this exercise, each team member rates his current motivation in the following five axes:

- **Security** or hygiene factors, including enough money to have a house, food, clothing, a family, health care, education, and some job stability.
- **Self-organization**, the need to have some decision-making power and some autonomy while pursuing the goals set for you, so you can decide how to perform your job and feel some independence.
- **Learning**, or the urge to grow your competence, get better at what you do, satisfy your curiosity, and fulfill your need to progress.
- **Vision**, the need to feel that you are contributing to a higher, honorable purpose that you can be proud of and that you behave according to noble values.
- **Networking**, or the need to feel connected and relate to other human beings who accept you; feeling that you belong to a community, and grow your status in it.

These axes are an excerpt of my knowledge-workers motivation framework, described in my previous book *Agile Management*, but there are several other schemes you can use. I would recommend that you take a look at Jurgen Appelo's CHAMPFROGS and his Moving Motivators game[4] as an alternative to this activity, or maybe as a complement.

Everyone will rate himself or herself on a scale of 1–5. Afterwards, the team can engage in conversations on why some motivation axes can be low, and how to improve team's motivation on those.

[4] http://www.noop.nl/2011/09/moving-motivators-free-exercise.html



Radar charts can be used for many other similar exercises by just changing the axis definitions. For example, the team can rate their code in terms of stability, security, quality, and performance; or they can score themselves against their self-defined values. You can also combine this game with the 'perfection game' (see Chap. 5).

Appreciation Exercise

In this exercise, everyone on the team gives one or more 'kudos' or appreciation notes to other people in the teams. The goal is both to close the Retrospective on a high mood and to identify good behavioral patterns that we want to enforce and encourage.

After all kudos have been issued, the team can group them in similar categories and try to spot the common theme behind each group. They can then use that information to update their WoW Manifesto or their team values.

Another way of running this exercise is asking everyone in the team to stand up in turns, then everyone in the team must make at least one compliment to the person standing up—everyone gets his turn to be complimented or receive kudos.

Confession

This is a trust-building exercise designed to make the team comfortable with the idea of vulnerability and transparency. In his book, *The Five Dysfunctions of a Team*,[5] Patrick Lencioni lists absence of trust as the most important team dysfunction, and states the need to be comfortable with the idea of vulnerability: if there is no trust and there is fear of vulnerability instead, the team does not engage in positive, constructive conflict and won't commit to action.

In this exercise, we ask everyone on the team to share a mistake they made during the last iteration or, if we want to make the scope broader, the biggest mistake or major catastrophe they experienced at work. Everyone reads his story to the team and, at the end of each story, the team applauds—this is important to show that we respect and value the courage needed to share our mistakes with the team.

It is important then that we engage in conversations about the reasons for such mistakes beyond 'bad luck' or 'clumsiness'. Maybe there was a lack of support, maybe a lack of training, or no standard to follow. When would we experience a similar failure again in the future? What would be the causes? What would a similar situation look like? How can we prevent this kind of failure? Is it just a matter of bad tools, or maybe there was not enough peer review?

Confession can be combined with the appreciation exercise to have a sort of 'personal pluses and deltas' review. As with pluses and deltas, the important thing is that we are able to spot root causes and design improvement plans with specific actions.

Pecha Kucha

Pecha Kucha, also known as the 20×20 format, is a presentation style where speakers are given a very short time frame, usually 5–10 min, to present a single idea. It is sometimes called 20×20 because sometimes the enforced format is 20 slides, 20 s each.

To run this exercise, team members (all of them, or maybe just some of them each time) are asked to prepare a presentation on some team issue.

[5] Lencioni P (2002) *The Five Dysfunctions of a Team: a Leadership Fable*. Jossey-Bass.

They can use slides, drawings, or just present bare-naked[6]—it's their choice. Some time should be scheduled after each Pecha Kucha to allow team discussion of the presented issues.

Issues can include some new technology, proposals for changing tools, repetitive team conflicts, or observed behaviors. Facilitators might let presenters choose their topic, or they can propose a specific issue they want to discuss with the team.

Bonus Games

– **Intervention**: a team member is selected, and the rest of the team decides an issue to be discussed with that specific person. This is a trust and vulnerability exercise, which is only suitable for somewhat mature teams. It is important that everyone has their turn being the 'intervened' person.

– **Team product brochure**: the team creates a brochure promoting themselves as if they were a product. They create a team title, slogan, value proposition, benefits from choosing this team instead of another, and so on.

– **If we were a**...: this is a visualization and creative thinking exercise that uses metaphors to make teams issues arise. Every team member describes the team, comparing it to a movie, game, book, country, or animal, for example, 'if we were an animal, we'd be a snail—because we are very slow and hide in our shells from responsibilities'; 'if we were a movie, we would be *Mission: Impossible*, because that's how it feels to meet the proposed deadline'.

– **Yes, and**...: this is a team consensus exercise. The team discusses an issue, but every phrase must start with 'yes, and...'. The team does not use negative sentences like 'No, I don't believe in that' or 'You are wrong'.

[6] Metaphorically speaking, of course.

9 Process Activities

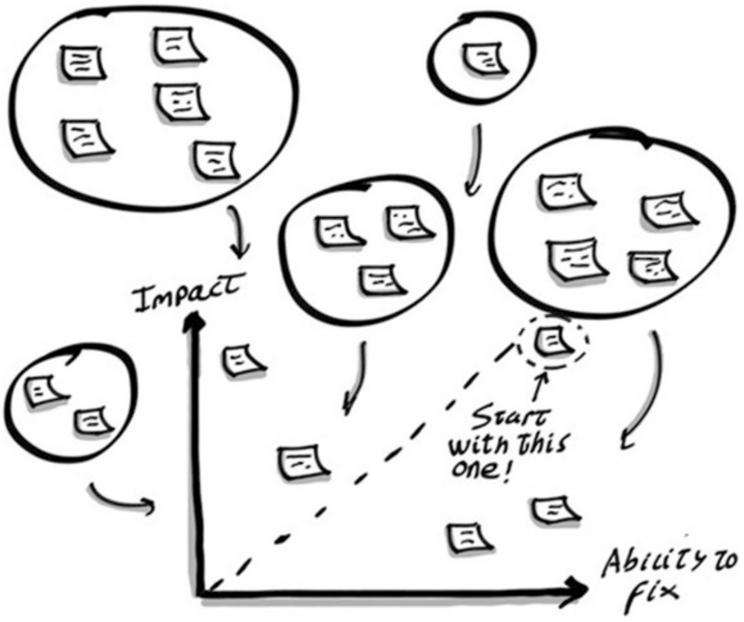
Overnight Miracle

This one is a classic in Agile teams. It is a visualization exercise that helps them define the desired state that we pursue through Kaizen.

As a set-up, you ask the team to imagine that an overnight miracle happened, and now all the company problems have been solved. But of course, they don't know that it happened. So the question is, how would they notice that the company had changed?

The team spends some time thinking, first on an individual basis and then begins a dialogue, and tries to list the main things they would notice if the company changed and all the problems were solved.

Usually, team will cluster all the similar ideas and will identify the common theme, the underlying problem, or impediment. Then, you can ask the team to dot-vote them in order to find those which are the most important—that is, have the most impact on the team—and which are the most affordable to fix or remove. You can combine this exercise with the Perfection Game and ask them how close to the ideal state we are right now, what do we need to advance, or what made us improve in the past.



Perfection Game: 'To Make It a Ten...'

You can run the Perfection Game in many facilitation situations. I sometimes use it at the end of a talk or a seminar to ask people for constructive advice on how to improve. The idea is that they must give it a 10, unless they are able to give some ideas on how to make it better—in that case, they can say 'I give it a 6; in order to make it a 10 you should...'

You will need to prepare a couple of things: first, you need to determine what are you evaluating—a specific process, a way of working, a given iteration, a product, some role in the company, and so on. Then, you'll need to define the different aspects that you want to evaluate. For instance, you could ask the team to rate their code in the categories of cleanness, usability, its value to the customer, security, robustness, scalability, flexibility, standardization, knowledge sharing, and collaboration. As another example, you could rate the last iteration along the following dimensions: success, efficiency, collaboration, motivation, learning, customer focus, or managerial support.

It is really up to you—the possibilities are endless. You could rate your environment, teamwork, tools, roles, artifacts, and meetings. After the rating, you will have several suggestions on how to improve them. Some of the suggestions will probably very vague, as in 'we have to make meetings shorter'—yes, but how do we do that? As a second part of the exercise, engage in conversations on how to make these suggestions actually happen.

Impossible Deadlines

This is one of the 'expansive thinking' exercises that I run—and it never ceases to amaze me. I was inspired by a story told about Steve Jobs.

One of the things that bothered Steve Jobs the most was the time that it took to boot when the Mac was first powered on. It could take a couple of minutes, or even more, to test memory, initialize the operating system, and load the Finder. One afternoon, Steve came up with an original way to motivate us to make it faster.

Larry Kenyon was the engineer working on the disk driver and file system. Steve came into his cubicle and started to exhort him. 'The Macintosh boots too slowly. You've got to make it faster!'

Larry started to explain about some of the places where he thought that he could improve things, but Steve wasn't interested. He continued, 'You know, I've been thinking about it. How many people are going to be using the Macintosh? A million? No, more than that. In a few years, I bet five million people will be booting up their Macintoshes at least once a day.'

Well, let's say you can shave 10 seconds off of the boot time. Multiply that by five million users and that's 50 million seconds, every single day. Over a year, that's probably dozens of lifetimes. So if you make it boot ten seconds faster, you've saved a dozen lives. That's really worth it, don't you think?

We were pretty motivated to make the software go as fast as we could anyway, so I'm not sure if this pitch had much effect, but we thought it was pretty humorous, and we did manage to shave more than ten seconds off the boot time over the next couple of months.

– Andy Hertzfeld, Folklore.org

As you see, the idea is to set a clearly impossible goal, like cutting cost or time by half, or producing double the amount of product. The question is, what would need to happen so we were able to do that? Usually, the first answer that comes to everyone's mind is 'we need more resources'—be careful with that one. You are trying to make a better process, not to throw more money into a crappy one.

If people run into an 'it is impossible' state of mind and don't seem to move out of there, you can tell some stories about how Toyota was able to cut their exchange of die time from 36 h to 9 min.[1] The risk in this exercise is that, if the goal is too small, they will go for evident improvements, but if it is too wild they will be blocked and might not be able to visualize a better state.

You might combine this kind of exercise with the perfection game or a value stream mapping exercise.

Visualization

Several different exercises might fall into this category. One of the most well known is the sailboat exercise, where the team must create a drawing with a sailboat representing themselves and then identify the ports (where do they come from, where are they heading now), winds (what is helping them), and anchors (what is holding them). A similar version is the speeding car exercise, where you draw a car that is heading to an abyss: a parachute is holding the car and keeping it from falling, and the team must identify what is holding them, what is pushing them to disaster, what is speeding the car, and what is helping it to brake.

The idea behind visualization is both using a metaphor to drive creative thinking, and empowering that metaphor with visualization and hands-on engagement instead of just talking or discussing. The benefits of visualization can be studied using the resources identified at the end of this book.

Some other metaphors that you could use include:

– Your product as a tree: foundations (roots), core features (trunk), feature families or focused epics (branches), and small low-priority features (leaves). The tree image can also serve as a problem-solving metaphor (root causes, causal mechanisms, problems, and defects) or as a culture metaphor (noble cause, values, artifacts, and behaviors).

[1] http://en.wikipedia.org/wiki/Single-Minute_Exchange_of_Die

– Your path to improvement as a mountain—the desired state (peak), the current situation (middle camp), where do we come from (base camp), and impediments (rocks and cliffs).
– A desert island can represent you in the market—what do you have to offer (coconuts), where are the clients (other islands), what's separating you from them (the ocean), how can you find them (map), how can you reach them (raft), and what risks are you running into (sharks).

Cars, boats, trees, mountains, and islands. . . Just be creative and keep it different—but focused!

News from the Future

This is another visualization exercise, although in this case the team uses more words than drawings. The idea is to write a 'news report' from the future, telling the story of the product or project the team is working out now. The news report should contain a flashy title, information on what happened, how it happened, and what the key events were.

Team can be split in pairs or groups of three, and then the different stories are presented and discussed. Some drawings can be included in order to keep the exercise creative and fun. The article should include excerpts from interviews (client, managers, team members) and should constitute a full story, with a beginning, plot, and an ending.

As a variation, two different articles might be worked out: one telling the story as a huge success, the other one telling the story as a terrible disaster—as in 'amazing company rules the market' and 'project crashes dramatically, making company bankrupt'. In each story, the team must identify the critical success and failure factors, then engage in conversations on how to manage those.

Pain Snake

In order to run this game, you should ask your team during the iteration to write a sticky note with every impediment, bad mood moment, interruption, waste source, defect, bug, or any other painful event that they experience in the course of their work, and then stick them to a wall, one after another, thus creating a 'pain snake'.

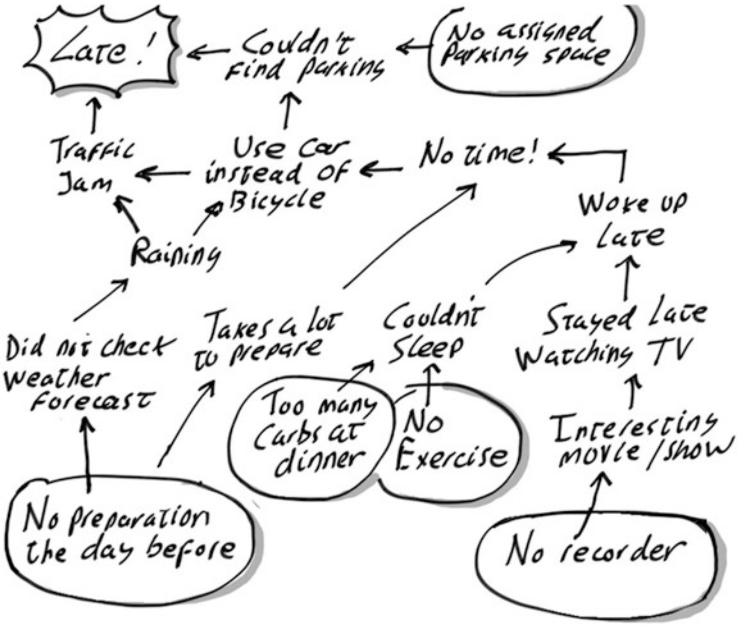
The goal of the pain snake is twofold: first, it will help you tell the story of the iteration at the Retrospective. But a second, more important effect is that it makes the team's pain visible, both to them and to others. You could forward pictures of the snake to the team managers so they realize how much the team is suffering from a hostile environment.

You could also ask the team to categorize the pain events and use different colors, so you can start quantifying the sources of problems and establish relative importance and priority.

A variation of this exercise is to use a cardboard mailbox and ask team members to put their insights in the box during the iteration. The problem here is lack of visibility, but it might be worth a try if we still haven't built a culture of transparency and trust on the team.

Causal Diagram

This is a root cause analysis exercise. The team selects a defect, problem, or mistake and tries to construct the whole causal sequence that led to it.



There might be several causes and possible sequences, and the team should brainstorm all of them in search of different events and stages in the problem sequence.

In the example, we would start with the observed problem (arriving late at work), then look at the most evident causes: traffic jams and no available parking space—which are, of course, not directly actionable. But if we keep tracing back, we will find some things we might do in order to reduce or solve this problem—prepare adequately the day before, record the late-night shows to sleep earlier, exercise, eat lighter dinners, or ask for a personal parking space in the company's parking lot.

Some more sophisticated formats of causal diagrams can include different symbols for events, causes, environmental factors, or actions. My advice anyway is to keep it as simple as possible.

Alternatively, the team can use Ishikawa/Fishbone Diagrams, the Five Whys and other root cause analysis tools, or maybe combine them in order to perform the analysis.

Connecting the Dots

The goal of this exercise is to review the team's past successes and improvement over time. As explained in the first part of the book, if the team does not conduct a similar reflection every once in a while, they might get frustrated to see that there are always new impediments to be solved.

You can combine this exercise with the visualization format and use a metaphor (a river, or a path) to explain the team's journey up to the present moment. The ups and downs of the journey should be highlighted, and the conditions that made them possible should be studied in order to see how could we enforce similar improvement events in the future or prevent failures and relapses.

Team improvements can include the introduction of new tools, training courses, successful deliveries, customer acknowledgements, changes in team composition, new members, changes in the development process, or growth in responsibilities.

5S

This activity is actually based on the Toyota-based tool of the same name for team environment improvement. 5S receives its name from the five activities involved:

– ***Seiri*/Sorting**: there is an old Lean saying—a place for every thing, and every thing in its place. The idea is to know where to find everything. Knowledge worker teams can do this by defining repositories, knowledge bases, or archiving rules. It might also apply to the physical space for tools, books, and documents. There should be a way to find everything in the minimum possible time. Seiri also means eliminating all the unnecessary tools, parts and items that are not needed on a regular basis—this could apply to software code, tools, documents. . .
– ***Seiton*/Set in order**: make sure that things are in a given order that enforces and supports value stream flow. For knowledge teams, it might include moving people closer to those with whom they interact more—hence, cross-functional teams. It also involves reducing movement of parts and waiting times, which might mean re-engineering some processes or making them more flexible.
– ***Seiso*/Systematic cleaning**: standardize the cleaning of all workspace and equipment, keeping them organized and tidy, and making sure they are ready for the next worker that uses them. Again, for a knowledge-based team it might mean the discipline to maintain information, follow information or software coding standards, keep information updated, use on-line tools following the agreed standards. . .
– ***Seiketsu*/Standardize**: ensure that there are standards so training is faster, processes are stable, and there is easier interchangeability. In some Agile environments, this leads to the creation of feature teams instead of functional/technology teams, so we can respond to different project demands in a more flexible way.
– ***Shitsuke*/Sustain**: ensure that the rules are followed, maintain discipline, inject energy, and prevent relapses.

You can ask the team to propose ways to improve their processes to enforce the 5S directives, or you could focus on one of them and explore ways to enhance it. A creative way to run this exercise is to ask your team to research how different companies are implementing 5S, then try to brainstorm ways of importing those ways to your own environment.

Bonus Games

– **Letter to the impediment**: similar to the 'News from the future', this is a storytelling exercise to foster creative thinking around impediments. The facilitator chooses a team impediment and asks every member to write a letter to it—‘Dear Impediment...’. The letter can include information on how the impediment is affecting them personally, how they feel, what would they expect in the future, and so on.

– **Feature trace**: choose one delivered feature, story, product, or piece of functionality and trace its full history ‘from concept to cash’. Analyze in search of waste, blockings, or problems. This is the equivalent of doing a value stream mapping on a small scale.
– **Yes We Can**: if the team has identified something they can’t solve on their own, ask them to list all the things that would be needed in order to be able to do so, for example, budget, decision power, tools, and collaboration from other departments.
– **If I were the owner**: ask team members to describe what they would change as the company’s new owners. Engage in further conversation, not only about ‘what’ to change but also ‘why’, ‘how’, and why this change hasn’t happened yet.

Product Activities 10

Kill a Feature Day

In the book *Rework*,[1] 37Signals founders Jason Fried and David Heinemeier describe their practice of killing a feature every once in a while in order to keep the product more simple and usable. Even if a small percentage of users miss the feature, the broad majority will have a better product—more features does not imply more quality, and of course the product is faster, easier to maintain, has fewer bugs, and it's easier to refactor and redesign.

On the other hand, Standish Group 'Chaos' reports usually show that 64 % of software features are never or seldom used[2]—waste! I don't know if this number is accurate, but I usually ask the people attending my seminars to share their feeling about it, and the vast majority feels like the number might be right—which is outrageous. In 'Kill a Feature Day' we don't need to actually kill a feature; usually, teams are not empowered enough to make those decisions. But as a product discussion exercise, evaluating what parts of the product just don't feel right can be a good source of insights on value from the perspective of the customer or enhance the understanding of the product from the team's perspective.

As a variation, instead of evaluating the current working features, we can run a backlog trimming exercise—what features in the backlog could be safely removed so we would still ship a great product? The discussion might also include aspects other than customer value—complexity, cost,

[1] Fried J, Heinemeir D (2010) *Rework*. Crown Business.

[2] Standish Group Press release, http://www.standishgroup.com/newsroom/chaos_2009.php

maintenance, risk, or usability. In order to run this activity, as with most of the Product Kaizen activities, the presence of the product owner or even the customer representatives is encouraged.

As a closure of this exercise, the team should craft a proposition to run experiments in order to determine if the identified features are really valuable for the customer or just something that should be removed. As an alternative, the team could propose how to replace this features with something they feel is missing in the product.

Low-Fi Prototype

There are several ways to run this exercise. The idea behind all of them is to make the team experiment and visualize the finished product, holding conversations and insights on the core features, value proposition, and competitive advantage.

One of the most popular low-fi prototype exercises is to create the packaging of the product. The 'product box' should highlight the best aspects of the product and give us an idea of what goal we are trying to achieve with it, and what problem we are trying to solve.

Another usual way of creating low-fi prototypes is to create a storyboard telling a user's journey since he realized that he had a problem until he solves that problem using the product. Storyboards are very useful as a base to identify user journeys and, later on, start creating 'Story Maps' that will become your product backlog.

If the strategy and the structure of the product are already clear, the team can move into more specific prototyping. Wireframes are a great way to explore product surface and visual distribution. Another variation of the exercise is to ask them to design something different from our current product or project and then ask them to discuss differences and resemblances between both designs.

Some rules behind a low-fi prototype are:

– They should be easily crafted by anyone without the need of special tools or props—usually just paper, markers, sticky notes, tape, scissors, and not much more.
– They should be created in a collaborative way.

– It should be easy to make changes to the prototype; hence, there is no need to discuss too much about design.
– They shouldn't go too deep into details (we are looking for ideas and structure, not for final 'look and feel').
– Design different prototypes to explore options and key features.
– They should be easily discarded—no huge investment made.

Storyboard
Agile Kaizen
Kaizen
Need to improve?
Agile Kaizen
The Book!
22/12 Activities Included!
- Real Teams!
- Better Products!
- Improved Process!
- Cooler retros!
Packaging
Agile Kaizen Web
Wireframe

Press Release

This one is similar to the 'News from the Future' process activity. The team writes a press release for the product with no help of the customer or the product owner—they will step in later and see if the team's idea of the product matches their expectations and discuss any mismatches. It is interesting to explore the origins of mismatches, especially if the team has been through a whole product inception or joint application development event and still has the wrong ideas of what they should be building or the main success indicators.

As variations of this exercise, you can combine it with low-fi prototyping and create TV or press advertisements for your product. Those should focus on the customer segment, problem observed, our solution, and how it compares to existing solutions.

Market Funnel

In this activity, the team reviews the market funnel for your product and discusses ways to improve it. I have used it especially with web-oriented products, but there is no reason why you couldn't run it with some other markets, technologies, or products.

Suppose, for instance, that you are running a clothing store. One possible funnel, based on Dave McLure's AARRR model, might be:

- Acquisition: people see your store's window or locate your shop.
- Activation (I): people decide to enter and take a look,
- Activation (II): people decide to try on some clothes,
- Revenue: people buy clothes.
- Referral: people tell others about your store.
- Retention: people come back to your store.

Once you are able to set a funnel for your product or service, there are several ways you can improve it. First, you can introduce new ways of obtaining relevant metrics: for instance, you could tell your clients to open an account or obtain a customer loyalty card so you can track retention. You can then devise ways of improving the funnel, like, for example, giving some discount to your customers if they bring in new customers in order to improve referrals. Some of these metrics and funnel optimization strategies might be implemented into the product itself.

Any attempt to improve the market funnel should be stated in the form of an experiment, including your assumptions of the problem, how you are going to solve it, and how you will know that you were right or wrong. This last part is better defined if you craft some metric and some improvement goal, for example, 'we believe that re-engineering our user experience would improve the conversion rate of our sign-up process; we will know we are right when the sign-up process conversion rate rises and user satisfaction about the conversion process rises too'.

Guess Who's Coming to Retrospect

Bringing your client to the Retrospective is both difficult and risky; however, by doing this you can gain insights on what he finds valuable, what can be labeled as waste from the client's perspective, and how we can improve our understanding of the customer's problem and needs.

The difficulties come from both the client side and the team side. Your client might be 'too busy' or not fully understand the goal of the activity—the facilitator must work with him or her prior to the Retrospective in order to ensure that the activity does not turn into an extended version of the iteration planning or review meetings. As for the team, they might be concerned that the customer will take this chance to ask for changes or to push the team's commitment. The facilitator must make sure that any changes they are able to identify are a new step closer to a better product, and that they should not discuss team commitment or deadlines, but more abstract concepts like definition of success, value, the nature of the client's problem, or user segments.

You can combine this activity along with several of the other games and activities in product, team, and even process Kaizen. For example, you could create low-fi prototypes with your client in order to better understand his thinking process and his values, or you could validate your assumptions on user persona profiles/empathy maps.

Variations of this activity include:

– Create a focus group with some users and silently watch them use the product, then ask them about their experiences and anything we saw that seemed contrary to our expectations.
– Design user personas of our different client segments, then run live, real interviews with client representatives to correct our assumptions on user profiles.
– Creating/reviewing Story Maps with your clients.

Benchmarking

You might need some advance preparation in order to perform this activity. The idea is to ask your team to compare your product or service with those of the competition. Many companies make the mistake of benchmarking and deciding on the benchmark criteria on their own, for example, 'our product is bigger'—but what if the customer does not care about size? Hence, your first task will be to define value from the customer perspective. The 'customer value radar chart' bonus activity can be combined with this one.

Once you have created the value definition, the next thing you might need is information about your competitors. It is especially interesting to find information about their value proposition—in terms of performance, quality, pricing, etc.—but their marketing or technology information might also be interesting: market share, sales channels, business model, and technologies used. This information might be available in internal reports, white papers, magazines, or even interviews with customers.

The teams use all this information to benchmark your product or service against those of the competition. Depending on your marketing strategy or your improvement goals, the team should analyze where you are lagging behind the competitors on a meaningful way—it doesn't matter if your product is uglier if you found that 'pretty' is not something your customers value in any way.

Toad for Breakfast

The name of this activity comes from an advice I once got from a very successful product manager. When I asked for his secret, he told me 'every morning I have a toad for breakfast'.

The idea is to spot the most buggy, ugly, messy, and uncomfortable part of your product and commit to doing something about it over the next iteration. It could be a complete re-write, a refactoring, a bug that is fixed, some grooming, or a redesign. It does not matter as long as you finish with something a little bit better at the end of your next iteration—then you can play the game again and choose some other part of your product and try to enhance it.

Show and Tell

This is a good inter-team activity. Every team has a predefined time to talk about what they have been creating in the last iteration, what have they learned from it, and how well the customer liked it. If you run this activity inside a single team, different team members could be able to tell about their work during the iteration or even go into the technical details, as we do in the next game (spin the bottle).

A variation of this activity is to give a small presentation on your project to the rest of the teams—or to other team members—once it has ended. A project post-mortem analysis can also be performed as a Kaizen event to learn more about how we provided value to our customer, how well the product met the customer expectations, and what waste sources we were able to identify.

Spin the Bottle

This is a peer-review game. We spin a bottle—or throw dice or use any other kind of random-picking—and the person who gets picked is the one reviewed. We then take a look at this person's work on the last iteration, for instance, projecting her code if she is a software developer or reviewing her writing if we are creating some kind of documentation.

The team looks both for improvement areas and for tricks they can learn. If there are quality standards defined and enforced by the team and/or the company, those should be observed while reviewing the team member's production.

Before playing this kind of game, be sure to introduce your teams to common guidelines to give feedback:

– Good feedback should be fact and information based.
– Good feedback should not be personally or emotionally addressed: start by admitting that you might be wrong and that you are just expressing your opinion with the best of intentions.
– Be always aware that any constructive feedback, no matter how good your intentions, will create some discomfort. Be nice and kind.
– Acknowledge the good intentions of the person receiving the feedback.
– Complement your feedback with some examples on how you could improve the situation.

Bonus Games

– **FedEx Day**: popularized by Australian company Atlassian, this activity has many different names and formats—hackathon, exploration days. The idea is to give the teams the freedom to build something and show it at the end of 24 h. They can work with whomever they want, and they can do whatever they want that is related to the work and purpose of the company. It might be a prototype idea for a new product or feature, a refactoring of an existing product, or a bug fix.

– **Customer Gemba Walk**: pay a visit to your customer—with your team. Walk your customer's value stream to see how your product fits in and how your customer uses it. A reverse-customer Gemba walk can also be performed: bring your customer to your value stream and see what he thinks about the way you deliver your products.
– **Story Mapping**: this very common exercise in Agile teams was introduced by Jeff Patton as a way to understand the different parts of your product, the customer's journey when using your product, how the different parts interact together, the different value they provide, how to design a minimal version of your product to ship as soon as possible, and how to evolve your product from there on. You can learn more about Story Mapping at http://www.agileproductdesign.com/presentations/user_story_mapping/.
– **Customer Value Radar Chart**: use a radar chart with some predefined axis representing value for your customer—price, performance, quality, functionality, usability, etc.—and ask your team to rate your product on every axis. Then play the perfection game—what would be needed in order to make it perfect?

Further Resources

Facilitation

Books

- Schwarz R (2002) *The Skilled Facilitator: A Comprehensive Resource for Consultants, Facilitators, Managers, Trainers, and Coaches*. Jossey-Bass
- Kaner S (2007) *Facilitator's Guide to Participatory Decision-Making*. Jossey-Bass
- Bowman SL (2008) *Training From the Back of the Room!: 65 Ways to Step Aside and Let Them Learn*. Pfeiffer
- Cooperrider D, Whitney DD (2005) *Appreciative Inquiry: A Positive Revolution in Change*. Berrett-Koehler Publishers
- Hammond SA (1998) *The Thin Book of Appreciative Inquiry*. Thin Book Pub. Co.
- Tabaka J (2006) *Collaboration Explained: Facilitation Skills for Software Project Leaders*. Addison-Wesley Professional
- Straker D (1997) *Rapid Problem Solving with Post-It Notes*. De Capo Press

Other Resources

- The University of Wisconsin's Office of Quality Improvement released a *Facilitator Toolkit*.[1] It includes several tools, activities, and frameworks for better facilitation of meetings, problem solving, and collaborating.
- You might be interested in taking a look at Lego Serious Play—http://www.seriousplay.com/. It uses Lego as a tool for building metaphors and enhancing creativity and innovation.

[1] http://oqi.wisc.edu/resourcelibrary/uploads/resources/Facilitator%20Tool%20Kit.pdf

Some Visual Facilitation Hints

– Dan Roam's series of books is a very good way to start learning more about visual facilitation. They include *The Back of The Napkin*, *Unfolding the Napkin*, and *Blah Blah Blah: What To Do When Words Don't Work*.
– David Sibbet's books, although more theoretical than technical, are also an interesting resource: *Visual Meetings*, *Visual Leaders*, and *Visual Teams*.
– Mind-mapping is also a very popular visual tool. The canonical reference is Tony Buzan's book, *The Mind-Map Book*. You might also want to have a look at his other books.
– *Sketching Using Experience*, by Bill Baxton, is a nice use of visual tools/ sketching applied to UX.
– Stay in touch with the Sketchnote army and find local sketchnoters at http://sketchnotearmy.com/.
– For something even more sophisticated, take a look at the urban sketchers community—http://www.urbansketchers.org/.
– Take a look at *The Sketchnote Handbook* by Mike Rohde at http://rohdesign.com/book/.

Retrospectives, Games, and Activities

Books

– Larsen D, Derby E (2006) *Agile Retrospectives*. Pragmatic Bookshelf
– Kua P (2012) *The Retrospective Handbook: A Guide for Agile Teams*. Leanpub
– Hohmann L (2006) *Innovation Games: Creating Breakthrough Products Through Collaborative Play*. Addison-Wesley Professional
– Gray D, Brown S, Macanufo J (2010) *Gamestorming: A Playbook for Innovators, Rulebreakers, and Changemakers*. O'Reilly Media

Some Other Resources

– There is an Agile Retrospective Wiki at agileRetrospectivewiki.org. It contains several formats, games, and activities for Retrospectives.
– There are several games and activities for innovation and learning available at the Tasty Cupcakes website, http://tastycupcakes.org/.
– There is a Random Retrospective Generator, also known as Retro-O-Mat tool, that is very interesting if you need some inspiration in order to make

your Retrospectives fresh again or just need some instant inspiration—http://plans-for-retrospectives.com.
- The Daily Retroflection twitter, by Ives Hanoulle and others, has been running for a while and provides powerful daily questions to think about with your team or company—https://twitter.com/Retroflection.
- The *Innovation Games* book has also a companion website at http://innovationgames.com/ . The same is true for *Gamestorming*—http://www.gogamestorm.com/
- The *Bootcamp Bootleg* guide is an awesome toolkit for design thinking and innovation/creativity workshops. It contains some of the Stanford Institute of Design foundation course basic teachings, and it can be downloaded at http://dschool.stanford.edu/wp-content/uploads/2011/03/BootcampBootleg2010v2SLIM.pdf
- If you can join a Play4Agile event, these are surely fun. Keep informed of upcoming events at http://www.play4agile.org/.

Process Kaizen

Books

- Imai M (2012) *Gemba Kaizen: A Common-sense Approach to a Continuous Improvement Strategy*. McGraw-Hill Professional
- Womack J (2011) *Gemba Walks*. Lean Enterprise Institute
- Womack J, Jones DT, Roos D (1991) *The Machine that Changed the World: the Story of Lean Production*. Productivity Press
- Womack J, Jones DT (1996) *Lean Thinking: Banish Waste and Create Wealth in Your Corporation*. Productivity Press
- Rother M, Shook J (1999) *Learning to See: Value Stream Mapping to Add Value and Eliminate MUDA*. Lean Enterprise Institute
- Liker J (2003) *The Toyota Way: 14 Management Principles from the World's Greatest Manufacturer*. McGraw-Hill
- Rother M (2009) *Toyota Kata: Managing People for Improvement, Adaptiveness and Superior Results*. McGraw-Hill
- Goldratt E (1986) *The Goal: A Process of Ongoing Improvement*. North River Press

Product Kaizen

Books

– Ries E (2011) *The Lean Startup: How Today's Entrepreneurs Use Continuous Innovation to Create Radically Successful Businesses*. Crown Business Innovators Dilemma
– Blank SG (2005) *The Four Steps to Epiphany: Successful Strategies for Products that Win*. Cafepress.com
– Osterwalder A, Pigneur Y (2010) *Business Model Generation: A Handbook for Visionaries, Game Changers, and Challengers*. Wiley
– Maurya A (2012) *Running Lean: Iterate from Plan A to a Plan That Works*. O'Reilly Media
– Adzic G (2012) *Impact Mapping: Making a Big Impact with Software Products and Projects*. Provoking Thoughts

Some Other Resources

– The Agile Inception Deck was designed as a series of questions to be asked before starting a project, but can also be used for product management—agilewarrior.wordpress.com/2010/11/06/the-agile-inception-deck/.
– The LeanStartupMachine Validation Board is a good way of managing and keeping track of your MVE/MVPs: https://www.leanstartupmachine.com/validationboard/.
– Roman Pichler has designed and is using some interesting boards for product management. You can take a look at his work at http://www.romanpichler.com/blog/agile-product-innovation/the-product-canvas/.

Index