ACE! 2014 Conference Proceedings

ACE! Conference

16-17 June 2014

Kraków, Poland

www.aceconf.com

ACE! 2014 Conference Proceedings

ACE! Conference on Lean and Agile Software Development

Contents

Liz Keogh: Keynote - Superheated Test Tubes and Anti-Bumping Granules

Can you hear me? Fantastic. That's a very generous introduction. Thank you, Paul. There's a lovely story about a Chinese painter who's asked to paint this beautiful picture by the emperor and turns to the emperor and says "It will take me a year." and the emperor says "Ok." And he comes back in a year's time to see this painter in his little village and he says to this Chinese painter "I've come for my painting." And the Chinese painter says "One moment." He gets his ink, he mixes up his ink, sets up his easel and paints in just a few minutes this beautiful, perfect painting. And the emperor looks at it and says "That's fantastic! But why did it take you a year? " And so the painter goes to his cupboard and opens the cupboard and out fall all the scrolls he's been practicing for in the previous year.

So, this talk did indeed get put together on Saturday night and I was still working on it five minutes ago but it feels that it has been in my head for quite a long time. It's been in my head for some months. So I've started…I had to do quite a lot of research for this, a lot of this stuff is actually reasonably new to me. So, what I want to introduce to you today is introduce you to a few bodies of knowledge that I think really important to know about or at least bear in mind when you go about your work. And it really is about change. I am getting used to the analogy of test tubes but this is really about why change is hard and when it gets easy. Ok? I know Gitte and Lilian are doing a talk called "Why change is hard" tomorrow. You'll probably get some other different ideas. They are both going to be right. Ok.

So, I have a little bit of a warning for you, guys. This talk contains some positive opinions about scaled Agile framework. Can I see the raised hands of who's never heard of scaled Agile framework? A few of you. Ok. So those of you who have never heard of scaled Agile framework, it comes with this really unique culture associated with it. And that is if you go on Twitter and say anything positive about scaled Agile framework, somebody will come and stamp on you. Right? So you have to the thing where you say scaled is bad, refer to it safe. Ok? And once you've said that safe is bad, it's ok to have a discussion about it. You have to say scaled is bad first or you'll get into trouble. I did. So what you have to do, every time you hear me say SAFe and it's spelled like that, we are not talking about things being safe, we are talking about Scaled Agile Framework, so when you see it with capital SAF like that, we are going to boo. Ok? So, I want you to practice. When I say SAFe.

Audience: BOOOOOO!

Liz: Fantastic! Great. Now, having done that ritual, we might be able to behave a little bit more like adults and stop treating it like villain. I'll be good. Ok.

So, what is a superheated test tube? So, a superheated test tube is what happens when you get something that's very, very smooth, very smooth glass and if you heat it up with a Bunsen burner and the liquid inside gets hotter and hotter and hotter until it starts to boil. But then an interesting thing happens. The liquid is so homogenized and the glass tube is so smooth no bubbles form. And it gets hotter and hotter and hotter and still no bubbles are forming until the first bubble forms. And then all the bubble form at once and it goes bang out the top of the tube. And it can happen if you put a glass pipette into it or a sterile or something like that. It's a really dangerous thing. Ok? That's what a superheated test tube did. I tried to find you some photos of actual test tubes for this talk but all the photos I found were full of focal shots. So, you'll have to excuse my bad art work.

All right. Who's familiar with the Cynefin framework? Who's not familiar with the Cynefin framework? Never heard of it? Right. There's a few of you. I am going to introduce it to you briefly. If anyone would

like a more lengthy ten minutes introduction of what I am covering here I will be very happy to give that to you at some point later in the conference. So come and grab in the brake. Cynefin is a way of thinking about different domains and different problems and how you approach situations within those domains. And they are not quadrates, there are actually more than four of them but I am only going to cover four in this talk. There's simple problems that children can solve, extremely predictable. You can go "Oh, it's one of those problems." You categorize the problem. So, your chain falls off your bicycle you go "Oh, Look, my chain fell off my bicycle!" And you put it on back again. And even children can solve them, right? There are complicated problems and they are frequently made up of different parts. And when you put the parts together you get a whole, and it's also predictable and it can be understood but only by people with the right expertise. So, you can think of a watch. A watch is made of parts and put it back together again. A watch maker can reassemble a watch and it works as predicted if it doesn't, it's not a very good watch. A mechanic can fix your car. And you being experts if I say to you "Use the registration", you probably know how to do that, right? It's complicated. But it doesn't have very many emergent discoveries associated with this.

Over this side of the Cynefin framework we have uncertainty. In the complex debate we get emergent outcomes. So, my favorite little story about that is a company called Ludycol who were making an online game called Never Ending. Multiple players on one game inviting lots of people to come and play it and they wanted to boost the number of people playing it. They thought "We should set up a site where people could share screen shots of our pretty game." And that would get more people playing it. So they put up this screen shot sharing site and they found people were not just sharing screen shots of the game, they were also sharing family photos and holiday snaps and interesting things they pictured and found and that became Flickr, started as a multiplayer game and you could never have predicted that, I mean you could see how it happened in retrospect but you couldn't possibly predict it. And then we have this loss situation of chaos, right? Chaos is accident in emergency, in your house burning down. Chaos is something that usually resolves itself very quickly and it doesn't always resolve itself in your favor. In complexity we have to try something out, we have to experiment. But in chaos sometimes things aren't safe to fail. I had a discussion with Ron Jefry on this on a blog post I was writing and he said 'Is it wise to try something that it's not safe to fail'. Well you know what? If your house is burning down you leave. And if you fail to leave, it's going to burn down with you in it. And that's not safe to fail but it doesn't stop you from even trying. Right? So it may not always resolve itself in your favor. My favorite example I used for software, for this, is night training, release something to production that was duplicating their trades. And by the time they managed to work out which server was coming from, and shut it down they lost 30 billion dollars. It's a lot of money. And you can bet some people where quite panicked at that point. So that's chaos. Guess which one the exploding test tube falls into? Right. Chaos. We don't like chaos. It's very high uncertainty. We don't like uncertainty anyway. It's our human nature to try and stop patterns that predict what's going to happen. And we feel uncomfortable when we can't. And we especially feel uncomfortable when we can see it might not resolve itself in our favor, right? So we really want to stop chaos from happening a lot of the time. And what we do is we apply safety. We minimize the things that could go wrong. This is what safety is.

I want to tell you a story about some people applying safety to something. I was on this project, really brilliant, highly innovative project, and these two junior devs went off for a couple of days to see if they could get fitness to work with what they were looking at. And they didn't really tell to anyone. They didn't think there was anything wrong with it. It was kind of related to the story they were meant to be working on. And there were so many people in that room that the standard got to a point where you only spoke if you had something to share with the team and they didn't think it was important and they didn't tell anyone in stand-up. And after a couple of days they said: 'Yeah we can't get fitness working.' And somebody went 'Is that what you've been doing for two days?' And got very upset about this and put a rule into place. We were not longer allowed to work on anything without talking to a BA first. Of course

BA's were in meetings all day, so it's really hard to get hold of them. Now I've been on that project for close on two years at that point. I kind of knew what I was doing, but even I was being told 'Pick up a piece of work '. I can't find the BE. I know I'll just fix these bugs. I understand these bugs, I talked with the testers about them already. What am I, caused? I am just going to go and fix these. And before I do it the architect was over my shoulder 'Did you talk to a BA before you did that work?" "No, look I can't get hold of the BA, let me just do something useful and I'll get feedback for them. Ok?" "No it's not good enough."

And overnight not only did the work slow down but every innovative thing that we'd ever seen happening on that project, every amazing ideas that people came up with that have made these project superb died because they only noticed the one thing that had gone wrong with a couple of people doing what they felt it was good and not the 50 things that have gone right previously. We are very, very good at spotting things that go wrong. Nietzsche says, I'm not going to read all these, he says 'Any explanation is better than none'. So whenever something goes wrong we tend to try and do a bit of root cause analysis. And if we find an explanation we stamp on that explanation and we think we solve it. Usually there are several factors, an environmental cause to everything that happens within a system. So there's usually several things that go wrong every time there is an accident. So, in a homogenized test tube the environment in which that liquid is boiling changes really, really hard. It's hard to get those bubbles forming. It's hard to make any changes the same way it was for me get any work done on that project right up until the point it explodes. Then it's changing. Ok? So you can think of Cynefin framework in terms of this pyramid. And when you are looking at the complicated and the simple domains, the predictable ones. You can see there is a very rigid hierarchy in those domains. So, in the simple domain everything is connected in this hierarchy where people tell you what to do. In the complicated domain you have to get consensus from everybody to get anything done. Sorry, somebody can grab my boo sign? Don't worry we will get to use it. Right. For complexity it really helps to connect the guys on the ground with workers, the people who actually know what the real problem is because they are the ones who experience détente the software developers usually caused it. Right? So we know what's happening. But in chaos everybody acts as an individual. In chaos we look after ourselves first. Do you remember being on a plane and the mask falling from the ceiling and they always say "Put your own on before you help someone else." Right? So, in chaos everybody is looking out for himself but it means we have got nothing to lose. And we can actually come up with really novel ideas working on our own.

So why is it that change become hard? Why don't we have this really imaginative society? Why is it really difficult to get people to adopt Agile in the first place? The reason is this J-curve of learning. What happens is you start off good at something, relatively good and then you learn a different way of doing it. Now I used to be a flute player, I haven't played for ages but when I was a kid I was a flute player and my first flute teacher got me some really bad habits, really bad. And that was fine for my grade 5 exam, I kind of got away with it in scrape 3. When I started getting up to grade 7, which is starting to approach professional level, suddenly my bad habits weren't letting me progress, they weren't letting me pass the exam. I found a different flute teacher who was prepared to help me undo those habits. And she took me right back to grade 2 and 3, stuff I'd be doing in junior school, right? She took me right to that level, to learn the right way of doing it, the better way of doing it. Now if you do that in an organization, and you're teaching an organization a new way of learning something, the organization has a certain tolerance to change. If you overwhelm that threshold, if you go beneath that threshold for change, they will reject the change. You know what? That Agile thing it's all very nice, now we can actually get some work done, and deliver some software because you know, the guys know what they were doing. And you said that would be something shipped within two weeks and you didn't ship anything. You know. Never heard that?

My friend, Mary Willeke, she has a thesis in adult learning and she's applying Agile techniques to university curriculum design of all things. It turns out they had a pattern that was a lot like Waterfall. And

she's been reducing the amount of time it takes to develop a curriculum from 9 months to three weeks. She says that people also have to change their identity in order to learn. So, I believe I'm a good flute player. I have to learn that I am not a good flute player, right? And that's difficult. It changes the way we interact with people. You can imagine flute playing is something I do for fun, when it's actually your work and you believe you are good at that work and now you have to learn a different way of doing it and a different way of interacting with the people around you. It's quite tricky.Corporations have identities, too. What we frequently see is when you get a little Agile team kicked off, the organization will act to wipe that team out. And they may not even realize they are doing it. They'll do it in different ways. For instance, your team might be asked ''Can you do a presentation on how amazing that project was?'' And you do the presentation and they are like ''That's really nice. We're glad it worked for you, but it's not how we work. We are not going to do it again. '' Or they'll take your team and say ''Ok, well. Obviously that was a very easy project. And you team was extremely successful because they're all obviously really great devs. And that is why it worked." So what we are going to do is we are going to take all the really great devs and put them all in different teams so they can pass their skills on to those teams. And of course it gets diluted and then it's gone. So this is how organizational antibodies act to stamp out change, to stamp out things that don't fit their identity. It's like having bone marrow transplant. Right? You get injected with somebody else's bone marrow cells. You have to keep taking anti rejection tablets or your body will reject it even though it's a good thing for you.

However, when we are in chaos, our tolerance for change grows. We are ready to embrace change, we are ready to act. We just want somebody, some direction. And command and control in this situation actually works doing top down big bang approaches to change. Revolution can be the right thing. You want to throw some constraints around it and give something, give a framework in which people can act. So Alan Carter has this thing about packers and mappers, people who collect information and people who set patterns, but he does talk about how to traumatize a person. I thought it was particularly relevant for this talk. So this is how he describes how to traumatize a person. You nuke them. Twice. You take apart their rigid, industrial, feudalistic society, it's very, very. This is Japan, right? It was very early industrial, just starting to lose their feudalism. You tell them an invader, America, is moving in and then you leave nothing to it. And in that situation a country like Japan is now ready for change. They want change. And they will take it even if it comes from the enemy. So into his works W. Edwards Deming with his statistical process control, he was actually asked to go over and help out with the census because they were trying to do a census and couldn't even make a phone call to Japan to get stuff sorted out. But the engineers took on his ideas and thought "This is fantastic! We really love this. We want to do this. "And his ideas are fool and flat in America but in Japan they were embraced. From that we get our understanding of Lean and the Toyota way be it startup, be it product development. This has all come from small things like this in the right place.

But evolution is painful. Have you seen a program of work where they fired the head of IT and got a new one in? And a quarter of the staff had been made redundant? That's not good. It's not what we want. It would be so nice if we could instead of having this exploding test tube have this gentle simmering bubbles of change instead. So how on earth do you get the gentle simmering bubbles of change? Well, I can tell you what it looks like. It looks like this. This is continuous improvement sometimes called Kaizen but we try to avoid the Japanese word. We have change, a tiny little change. And it is usually a change that comes from within the team so we are also giving them the habit of changing themselves which a really good thing and could be quite long lasting. And it's not going to spring the organizational threshold. It's little change. And then another little change, you know. And we keep doing this until we reach this new level of effectiveness. If this was possible, why isn't it happening more? Why isn't it happening in those companies in particular which are in chaos? Those rigid, bureaucratic organizations that simply can't deliver anymore and they are desperate to do something and they want change. They don't want little change, they want big change. They need it now.

Here is why it doesn't work. You make a little change. That was good. Improvement. But the next time you make a change is less effective and it's less effective. And you've probably experienced this as devs. It doesn't matter. How many hands up in the room? No devs? Not one developer? I know there's a few of you. Hands up if you're team leads, testers? One tester, I can see. Please don't shoot me. Hands up coaches, process consultants. I want to come around and find out what the rest of you do. Ok. So, it gets harder and harder and then we suddenly find we've reached this plateau where we can't improve anymore. And the point you've reached that plateau people start getting a bit demotivated as well because they really wanted to be better, they can see it could be better and they just can't make it happen. There's a gap. There's a gap and it's really hard to cross that gap. And it happened because we have got this local automation going on which managed to improve our team that there is something outside our team that's stopping us from changing and we haven't got any influence over it. So, I am going to show some examples of those gaps. This is probably the most famous: "Crossing the Chasm". This lovely curve is what you get when it's a small change. You've got a product and you're evolving it and getting out on the market slightly better and slightly better and slightly better and slightly better. And that's fine as long as it's not disruptive. But if you have a disruptive product that's going to change stuff there's a chasm between these early adopters and the early majority. And it's really hard to get across that chasm. What he says is "You have to build a momentum and build a momentum and build a momentum and build a momentum until it ticks. Sounds familiar.

This is Bob Marshall's "Rightshifting". He says when we look at organizations which are effective, we like to think of it as a bell curve. If you know. Some guys here mediumly effective and some people who are really good like Google, some people who aren't great. And he divided it into these four, these four types of organizations. Ad-hoc people who are just getting everything done. We are not looking at how we are getting things done. We're just kind of doing it. Analytic stuff, so tailorus thinking, scientific management. Synergistic, this is Agile, right? This is us. Kind of innovating a bit. And then you get chaotic. And he describes chaotic as ridding the knife edge where you're just about to collapse into chaos but you don't. So, companies like Valve. Valve have a lovely, lovely employee manual. And it's got instructions for moving your desk, unplug your computer, leave your desk. You're allowed to do whatever you want to at Valve. And you want to collapse into chaos but because of the structure of the community they have got in place, it never does. And they can use some magic as a result. But he also notes that there are these gaps, these transition zones between these organizations. When you start improving, your improvements start high but then they drop until you get through the zone and then they drop until you get through the zone and then they drop and eventually you reach your maxim of effectiveness. So, your improvements are most effective after you've just made the transition into the next zone. Sorry. Wrong button.

Richard Durnall talks about these patterns of Agile adoption. So, he says first of all the people break. People. I once went to an organization where there were these two coaches on their way out and I was on the way in to replace them and I said "How is it, guys?" and they said "Awful. It's just awful". I said "What's wrong." He said"They really don't like it. They don't like Agile at all. I had this one guy following me down the corridor going Agile's fragile, Agile's fragile, Agile's fragile behind my back." Like wow. That's from somebody who's a little bit scared, right? I found out who that guy was. By the time we had finished he actually came to the coach to coach this course. So, it took some time.

The tools break. Most of the open source projects we've got have been created, things like Hudson, things like open libraries in Java and .net that I am used to using. They've all been create because our current tool just didn't support the way we were working anymore. We virtualize stuff so we can get hold of test servers.

The governance process breaks. I remember when I was on one project. We did all the really risky, new things upfront first so that we could upfront that risk. All the things that we were unsure whether it was going to work or not we tried out just to see if the project was actually on a sound basis. And our project

manager was yelling at us "Why don't the estimates have any correlation with the amount of time you are spending on these things?" And we were like "Because we've never done them before and when we told you we knew how long they were going to take, we were lying." Right? "We were making it up because you told us you wanted a number." If anybody does that to you, by the way, there's a lovely site called estimategoat. com where you can rub the goat and it will give you a number.

The customer breaks. I've always had the best way to get your stake holders involved. It's the shit that something that's wrong. And then I look at it and realize that I actually need to engage with you if they want to get it right. But they are not familiar with spending that much time working with the dev teams. They are used to doing a bit at the beginning when they a get a document together with the BA's and then it gets handed to you and they never get to speak to you again. And now they have to sit in the room with you every day, frequently while they try to get their usual day job done as well.

Then the financial controls break. How on earth do you budget for a project you don't know what the scope is or you don't know what the schedule is? Ahhh!

And then the organizational structure breaks because if you are measuring individuals with KPI's and the testers being paid depending on how many bugs they find and then there's bonuses and gaming going on. That won't allow Agile to suffice and that breaks.

So what do you do if you are in this chaos? Dan North says if we ran chaos and everybody is trying to get stuff down and they're pushing against each other. What you want to do is getting all aligned. And he said " If that means top down teaching of all Scrum, you do it." Right? When they're all aligned, when they understand what the purpose is then you can get them to innovate. You can start allowing innovation to happen again. And it'll be happening. It might change the direction of the company but it's going to be roughly in the right direction. Let's talk about SAFe.

Audience: Boooooo!

Liz: Right. Say, when I heard about SAFe.

Audience: Boooooo!

Liz: I thought it was that kind of thing. We're throwing some constraints around the chaos to get everybody aligned. It looked a lot like Dan's pattern. What I've realized is it's not always sold that way. It is not suited for organizations which are already working. It really is heavy on the constraints. You do not want to apply it to organizations that are already shipping regularly. It's really suited for those big, bureaucratic, rigid organizations which are in chaos because they can't actually ship anything, but they don't know what to do anymore because obviously that rigid isn't working. So, they are looking for something to give them some guidance. And it's also sad, you know. They have got half a constant sense of danger. So we are explicitly choosing it for organizations that are tipping from that rigid complicativeness and simplicity into chaos. So, why you might want to do that? This is the MIT Sloan review that they did. They were looking at different companies and how well aligned their software was and how effective the development teams were. So how much value adding work was going on compared to non-value adding work. And obviously where we want to be is up there, right? Highly effective teams doing really valuable work. This is IT-enabled growth. Yey! That is the number of people who were in that space. So it's 7% of people in that space. It reduces your IT spending by 6%. That has to be a good thing. And you get 35% compound annual growth rate on your 3-year sales. I don't know what that means but it looks good, right? It's definitely good. Yes!

The place where we don't want to be is down here, not very aligned, not very effective. 74% of respondents, not really spending any extra money on maintenance but their sales are dropping year on year. So what do you want to do? Do you want to go down the route of being able to have really effective devs who are delivering the wrong thing? Or do you want to deliver the right thing and not worry too much about the

engineering practice? Tom Gilb who's talking tomorrow, I believe, has this thing called the matheo offer. Money or your life. You can choose one, but not the other, right? If you had to choose one, engineering practices or alignment, which would you choose? Well-oiled IT has 8% of people ahead, they're reducing their cost by 15%, and they're getting better sales because they are producing the same product they've always been doing but the quality of it is growing. But when you actually look at the other corner we get a bit of a surprise. 11% of people are hacking stuff out that should be really valuable to them but it's not very good quality, it's not maintainable. If you've heard the code of this liability this is the space in which they are talking about, right? You can't maintain it. And the devs are getting tired and it's getting slower and slower to produce anything. Your costs have gone up and your sales go down. I like to think that a little bit of that buys you a little bit of this, buy you a little bit of that, buys you a little bit of this. There might be some diagonal path to reach that. But if you absolutely have to make a choice, learn how to build things right before you try to build the right thing. Get those engineering practices in place. Make the cost of change low. You cannot experiment when the cost of change is high and expensive because your experiments won't be safe to fail. Let's talk about SAFe.

Audience: Booooooooooooo!

Liz: I can hear these guys do it. Come on guys! Booooo! There we go!

Audience: Booooooooooooo!

Liz: Right. It's important that I inoculate you against what you are going to discover on Twitter, right? Because this happens. SAFe stifles innovation. It is heavy on the constraints. It does ridiculous thing with story points and it basically says half a day for one person is one story point. And we are going to plan using that across multiple teams. And we going to make everything sized in those story points. That's how we are going to get people aligned. That's how we are going to get our dependency results. So it's really heavy on the constraints. But you know what? It has the risk of actually shipping something. This is for companies that cannot ship. This is not for all companies, this is not for companies that are working, this is for companies that cannot get anything out the door. It's better than Waterfall.

So, where's this heat coming from? Where is this actual flame in our organization that's causing this, this inability to actually get stuff done in the first place, that is causing the bubbles bust it's also causing the heat of explosion? You can imagine a team that are trying to get stuff done, right? They want to deliver. They want to deliver. They want to deliver. And they can't. They just can't get anything done. It's not because of internal stuff. They have influence over their own team, right? They can change locally. It's because of another team. They are either devs can't get a server of ops guys, the manger won't let them. It's friction between different groups of people that's causing this.

So what are anti-bumping granules? Well, in chemistry we put these jacket pieces of glass into the bottom of the test tube. And they act as nuclei for the bubbles to actually form. And then we get this gentle change, we get this bubbling. And we can see what's happening and choose what we are going to do about it before it explodes. So how would we get that given that the heat is coming from these two different groups of people? How would you actually get that bubbling to occur rather than suddenly being a big explosion? Well, if the big heat is coming from friction between two groups of people then what we want is a little bubbling. It's about individuals who have relationships between those two groups. They are our anti-bumping granules. Having just a little bit of connection between the devs and ops teams, one guy who's gone for a drink with the ops guys and knows what they are suffering from and says "Hey, look, the reason we can't get a server is because of this, is because we deploy some really shabby codes and we have to rewrite the scripts to reboot the server, you know." Somebody who actually knows what the problem is. And then you get your bubbles.

The innovation cycle talks about how we get to that point where we can start getting innovative stuff happening and getting that train. And if you remember the first mobile phones. Yeah? So the first

mobile phones were very differentiating and K. Sarah put facing the user and we learned quite a lot from that. They learned that people actually want to take photos and the Nokia spoiled it. And now it's commoditized. Everybody has got a camera on their phone, right? So, it's become a thing that we understand really well. This is what happens. Differentiators become spoiled and then commoditized. And off the back of that we can build more innovative stuff like Google glass, like little games which actually overlay your environment and things like that, right? So we can start building off the back of it. But when you are in a situation when you cannot innovate, cannot innovate, cannot innovate, what you want to do is a shallow dive into chaos. You want to cause people to act as individuals or very small groups. You want to split them apart, you want to get a wide variety of ideas from as many people as possible. This why we do silo work in our retrospectives if you've ever done it with the post-it notes rather than just shouting thing out. It's because then it harvest all of our ideas rather than causing us to come to consensus. And when we've got a wide variety of ideas from that, we generate much better experiment to try out. And then we've got our differentiator back again. So what does this look like, when you actually do it with large groups of people?

This guy Ronald Burt wrote about '' Structural Holes'' they stole almost, almost all of this from Jabe Bloom talk. He does a talk on social capital if you are interested in this I highly recommended going and look at it, it's really good. And he says, within these groups that are very tightly connected you have homogenized opinions. They all got the same ideas. And you can imagine that in a large organization where everybody has to get consensus from everybody else to get everything done you have a large connected group. You actually need holes between the groups, we need very loose connections between the groups, not really tight once, because the best ideas will came from these groups when these groups with different opinion meet. And you don't want them to meet for too long. So it's not just about making connections is also about the quality about those connections and the rarity of those connections.

This is something I like to do. When somebody invites me in to help them change, do something, you know get the guys preprogramming, whatever. I'm at the value streams and instantly I've got a lot of Russ's comments. Map the values streams. So what I say is : '' Ok, you know what the dev team writes, analyze and codes, test, whatever'' that's for local value stream, but where did the actual requirements come from, did they just come from the users. The users can tell the devs what they want and the devs just do it? Oh, no. There's a PMO office and they sort out the bugs into batches of stuff that needs to be done and the architects work with them to do that. OK, so there is this big project that it's been approved, there was the PMO office who decided that, just go ahead. Oh, no, before that there is like a board, and they do a feasibility study on which projects are feasible. OK, so where does the project come from? Oh, that comes from this company board, you know. And you actually find there's a massive value chain, I find one place on an Agile project. There were 6 months of this going on before the dev even saw a sniff of the actual work. And this wasn't high document analysis, this was just a very large organization. And then you go at the other end. Ok, when the devs are done and the teams done, can they cheat, can they just put it to the rush? Oh well no, because it's an architect rewiever, the security rewiever and the ops need to get hold of it and now decide when to deploy it. And it takes ages to get it to the users. What I ask people to do is. Can you put a dotted line around your sphere of influence and they can usually do that very easily what we've done it. We did it on whiteboards with actual realistic figures to remind them it's all about human beings. And I put the dot around it, and I then I say ''Ok, where is your biggest constraint in this value stream, where is something that is slowing you down most? And it is almost always outside the sphere of influence, because if it was inside the sphere of influence, they would have fixed it instead of hiring me. And I say ok, I can help you within your sphere of influence, because if I want to make a change and you think it is a good change you will help the devs to get it done, right, and you'll support me in that. But all that will happen you will bump up against that other constraint. You are going to bump up against the ops team because they already have one release every 2 months. And it doesn't matter how good you get you're going to hit that. Go and make friends with the guy in that team who has got the

constraint. Go talk to him, take him out for a pint. Let him know that if you are making changes that will make you more effective. And at some point it will start cascading on to him and find out what you can do about that together and if necessary escalade it, right? Go make a relationship, make good connection with that team. Find out if he's already got a connection, maybe there's a dev who knows an ops person. This is devops, this is how devops starts. Devops is just when a dev and an ops person get together and have a pint and a bit of a chat about what they can do to help each other.

Here's another model. On this ''Pareto Teams'' Jabe Bloom's talk again. I've stolen from that. What he does is they rotate 20% of team members around so they keep this core but the idea is always coming in fresh. Very loose connections. This is fortified, they have squads, squads are arranged into tribes, squads are each working on one thing and the tribes are working on related things. They have chapters which cross the tribes, but they are only ever done within the tribes. So things like the web devs will have a chapter, the testers will have a chapter. So although you have cross functional teams, you also have these communities and then they have guilds and the guilds cross the entire organization. And anyone who is interested, you know, people who want to become an Agile coach can join the Agile coaches guild. And these are supported by Spotify, this is really well supported.

Let's…. ''Convergence and Divergence '' this is a really interesting one, Simon Cromarty wrote a whole blog post on this. And he says what they do is they explicitly choose whether to put people in teams or pull them out. So he uses the example where they are putting one HR person in each division, so this HR becomes familiar with how actually work interacts with the people on the ground. So they are diverging a chart and the same time the ops who have been sitting with the dev teams for ages, now the devs have a much better understanding and a much better relationship so they are putting ops central again. So the guys can work together and share it, they are making that very explicit choices. And said standardization in a way –point, it's a journey and you keep doing this cycles. Let's talk about Safe

Audience: Boooooooooo!

Liz: I don't need to hold up that sign any more. On the surface it looks a lot like what Spotify are doing, right? We have this Agile relieved trains which are programmed of related work that need to happen. I really love the analogy, we are getting on a train. We are all getting on a train together and every so often we are going to stop the train, we are going to dump off some luggage we don't want any more, we are going to bring on some new cargo and we are going to deliver the cargo we've got to where it need to be and it's in the right place. It's a really great analogy, I love it. It does evolve putting absolutely everybody in a room together for two days. And as a dev with very limited patience, I can stand that idea, but apparently even within that room they are working in very individual groups. And Martin Burns who's in here and he's the person who taught me SAFe, he's over there. He was watching us all trying to do our planning in two, just two small teams together, and quietly watching us failing and giggling at us, right? The idea is you do your plan as individuals, but if you do find dependencies the other teams are in the room so you can talk to them. So that is that loose relationship and also remember that you've just shaken up an organization which have very strong role based communities anyway because they used to be a testers team, a dev team, a dev group, you know. So those relationships are already in place so I think that is a tacit community. I would love SAFe to have more explicitness about that, and then it would look lot more like Spotify and people would probable complain a lot less about Safe.

This is Chris Matts, another one of my favorite people. He has this thing called ''Distributed Consciousness''. So Chris used to go to conferences and learn stuff and then he found out that it was way too much to learn. So what he did was, he'd go: '' Oh Liz, which session are you going to? Oh, that's great, that's really good. And Mark which session are you going to? Oh, the same one as Liz. Look, Liz is going to that session why aren't you going to this other session and you two tell me what you thought. We meet back here and you tell me what you thought." So he is short cutting going to all the sessions by getting us to learn for him. But then he thought that even was too much. So all he does now is get people to go to

the different sessions, remember who's learning what and then when somebody needs that expertise he goes "Oh, yeah, I know a person you can talk to and connects them. And that's what we call information broker. So he's become an information broker but it has the effect of spreading his learning amongst multiple people so we say he's managed to distribute his consciouness amongst the community now, as a result of it. He doesn't actually do learning except through other people. It's not quite true.

My favorite, favorite example of Chris connecting people very quickly is when we were at Agile 2009. 44 floors of this skyscraper and the penthouse had a bar in it, so we met up in this penthouse bar and as I walked in Chris is there sitting in this big well of seats and he goes: ''Oh, Liz have a beer'' and hands me a beer "Thank you Chris" and immediately I could just dump my bags because I didn't have to queue at the bar, wishing I could join that group over there, you know, feeling like an idiot and then sidling to see which conversation I could join. He immediately brings me into the group, and it was great for new comers, who are new to that environment, because large conferences like Agile can be quite intimidating, I found it. Even though I've been to a lot of them, I still find them quite intimidating. So it's very welcoming and then the barman comes round and says "Chris can I get you a drink? And he's"Yea, beer for anyone who needs it and two for me" And he gets his two bottles of beer and somebody walks in and he goes ''Mark have a beer!'' So immediately that person comes in. And I said to him 'What happens if nobody comes in? and he goes ''What do you think?'' I like Chris.

So I promised as part of this talk I would give you an idea about what your responsibility, what your power as an individual is to make a difference. This is one of my favorite scientific books ''Permutation City''. It's about people who are downloading their brains into computers, but the computers aren't, you know, inner banks culture computers, they are really powerful computers but the human beings run slower in the computers than they do it in the real world and they are copying themselves, they not they not dumping their brains in so that they can make multiple copies if they want to. And there is one recluse and it's a story within the story, this one recluse is rumored to have decided he wanted to speed his copies up so he asked a computer to optimize them for him. And the computer gives him back his copy and says "This is optimized." And he says "It is still not fast enough. Make it as fast as it can possibly be.""What is the end result of this program that is me, how fast can you reach this result for me" And the computer comes back with zero. He has no impact on the universe whatsoever. That's what our power as individuals is. It's nil. We have no power as individuals whatsoever. We make no difference to the world whatsoever until we connect with someone else by talking to them or through the artifacts we leave for them. Think of how many times we focus on individuals still, we put our names on the story card. We look at what a team is doing, we talk about helping a team to change and we are going to have a pilot team over here. What would happen if instead of measuring what people were up to, and visualizing that, we visualize how relationships were going? Where relationship were, where there were gaps we needed to bridge. What would that look like that if we started visualizing that and started working on that. We have no power as individuals, parallel connection is everything. Thank you very much.

Tom Gilb: Keynote - What is Wrong with Software Architecture?

Good morning! If you want a copy of the slides there they are. When we finally get synchronized with Dropbox there will be about 14 papers on Agile at that site which isn't ready right yet. OK, so I'm going to talk about Agile methods but I'm going to talk about Agile methods that in fact our software engineering in the sense of using numbers and measurements and feedback. And I'm going to try and show that the best practices I recommend are proven by measure and quantified data. And in my dreams all talks at conferences offer the same evidence for whatever they are proposing. That is facts that really work and how they work and what they cost.

I, I love very busy slides. Some people don't. So if you don't I have a suggestion close your eyes, relax, meditate, maybe a little listen, but I have to have the busy slides because they are reality, they are the detail, you can study them at your leisure they are pretty downloadable. If you really don't like busy slides and you just want an easy ride I recommend my TED talk where they forced me to get rid of all my busy slides whether I liked it or not.

I was a programmer for about 20 years and decided there was some higher calling, something let's call it software engineering, requirements engineering, architecting, and I thought that was a challenge I would like to figure out how to do it and so the transition from being a programmer to being a, somebody who helps analyze and design that what programmers should do. Maybe some of you would like to make that transition after you programmed for about 20 years. How many people have programmed at least 20 years? Ok. Time left for the rest of you. Ok? This is an Agile so I'll bring out my Agile credibility. You can call me Grandpa, I have white hair and I was fighting for these ideas since the 70's actually when everybody thought I was absolutely crazy. And nice to know that the ideas that, iterations and feedback in particular are now widely adopted.

Here are some of your heroes, although we don't really need them. OK? But coming from specifically where they picked up their wonderful Agile ideas from. One of the things that worries me is that so many of you somehow had an education where in my view you haven't been taught enough IT and software history. So I am going to do a little bit of that at the same time. Those who.., if we don't learn from history the lessons we've learnt, the methods we've learnt, we will just reinvent the wheel, we will make the same mistakes, we will waste years of our lives. So part of my role is to be not only the grandpa but the historian. I remember I met the people who did it decades ago before some of you were born. So, just for fun I picked up this picture of me in an Indian guru costume and that's my son Kay actually getting married but I am symbolically I am the guru teaching my son and I hope all of you will be my symbolic sons and allow me to teach you what I know and I don't know enough, I hope you will teach me back what I don't know because there's this fast body of knowledge that I don't know.

Basic ideas of this talk. First managers are bad at, managers, booooo, right, are really bad at deciding your work environment. You should be the ones to decide it. IT architects

Audience: Booooo!

Tom: are really bad at designing what you should program, you should be in charge of that, too. And so what I'm, what this talk is all about is delegating power to the grass roots who are much better at making good decisions about both their working environment and what they should be programming. Yes!!! Yey!!!!!!

Audience: Yey!!!

Tom: Yey!!! Ok. So here is just, the talk I was originally going to hold, if you are missing that one about architecture, it's on a video and basically I am telling 300 architects in London that they are childish idiots and quite an embarrassment to their profession. And I think they agree. Ok? How are we going to do this? Basic idea is we give some super ordinate objectives from our stakeholders and users to the programmers and they figure out smart designs and measure whether they are getting them. That's for the design. For their own environment we give them information to the programmers about their bugs and problems, let them analyze root cause, let them figure out how to cure and let them implement measure when it's done. So it's quite simply clear delegation of power.

Here if nothing else, I thought the picture is funny, enjoy it. In case you didn't realize it, that's you. Right? And the rest of them have other talents and trades. I'm going to start off by giving you some case studies from Raytheon and IBM. The contrast here is from top down decision making management decides what your working environment is going to be and what I really would like is to get you in the loop deciding what your work environment should be and trying out your ideas and finding out whether your ideas are really what you want. Let's see, push button. There we go. In 1970 and 1980 two people at IBM Michael Fagan and Ron Radice they invented something called ''software inspection''. And the intent was primarily to collect data on the working environment of the thousands of programmers of IBM and use these statistics to improve the environment. Long story short it didn't work. I was there. I didn't have any wiser ideas myself. I didn't know why it didn't work. It sounded like a good idea, get statistics, show what was frequently happening and fix the problem. Along came Robert Mays and Carol Jones in 1980 from IBM Research Triangle Park in North Carolina and they had a very simple idea. Delegate the power to the programmers themselves to analyze their own environment. Don't let the managers do it. Don't make it a statistical accumulation. Make it, you know, our work, bugs that we created, why did we do it? Were we sleepy? Was that the reason? Maybe the process changes to get more sleep before we get to work. That kind of thing and that worked. It worked so well that Robert got a citation from the chairman of IBM and $100000 prize just to indicate this was not a minor accomplishment it was impacting the whole IBM.

Here's a detailed case study which I dearly love. You can easily access the details from the links there. We'll start off by, take a look at for 1000 programmers, take a look at what is called rework cost. Rework cost is the cost of fighting their own bugs and changing and retesting things. It is what normally is if you're not measuring and try to fix it at about 43%. That means out of 1000 programmers 430 of them were tending the wounded from friendly fire. They were fixing bugs they themselves had created. That's what I call a really stupid army. Unfortunately most of you are in that situation now but you don't know because you are not measuring it. Try measuring the rework of what you do. Were pretty sloppy. Anyway they learn from Phillip Crosby's work on "Quality is free" that high rework is not a necessity or an act of nature. You can gradually improve your process so as to reduce it. In this case they reduced it by factor of 10. And they reduced it by a factor of about 2 within the first year. So in the first year they were actually saving about 150 programmers for productive work and then they improved the process, sometimes they failed to improve the process well and they keep on improving the process systematically use a thing called defect prevention process, DPP. That's the invention of Mayze and Jones.

Here are some other side effects. The productivity of the programmers went up by a factor of 2.7. This is on 26 projects. This is partly getting rid of the waste cause by fixing their own bugs partly by smarter ideas that helped make them work more productive. Here's another one. They were actually quite good at running over any budget they had by only 40% but within 1 year, they managed to get to the stage they did not run over their budge ever for the next x years. Most of us, you know, run over budgets by about a factor of 3.14 every time. Ok? Even if we multiply the original estimated time with 3.14. We seem to do that. Here are some of the process improvements they made just to give you some sense of the detail. Interface problems, regression test repeatability, inconsistent system inspection process, inspection as eyeballing code or code review here. And bad requirements updates and more problems with requirements. These

are the real examples of the changes they make to their system to increase their productivity and reduce their lack of productivity. The bug density went down by a factor of 3 which is nice, not exactly 0 defects. Interestingly they discovered that when they invested time in training people to do this and allowing them to do it. Because I know your first excuse is ‘’This is very nice but we don’t have time.’’ Ok? We’re so busy fighting the fires we don’t have time to avoid the fires. But they, they figured out that they were getting 770% return on investment from investing in this, ok? Now how much is your bank giving you right now, is it 2%? 3%? Ok? They use this argument for their chief financial officer to get even more money to do even more and to do it better. So you may not be interested in finance but you may be interest in financing your adventures into better technology and I think putting the argument about how…, what the payoff is will loosen the first strings that’s what happened in this case.

This is just a basic map of Defect Prevention Process. This is also identical with capability maturity model level 5 as it started out. I’ll give you another address for where is this capability maturity model. Let me check something. How many people here said”I knew about the Defect Prevention Process before you began your talk, Tom.” Pretty good. About less than 10%. OK. How many of you have ever messed about with capability maturity model? Ok. Another 10% and almost a different group. Thank you. Ok.

So what’s going on here is that the 1000 programmers who later got merged with another 1000 they are analyzing their own bugs and specification defects as we see in requirements. They’re suggesting their own work environment changes such as the ones I just showed you a slide full of and they’re reducing their 43% rework by 10 times. By the way over an 8 year period. Management would like to get it done this year. Reality is cultures change slowly but they do change if you measure and you persist. And so, and you get big changes in the first year down from 43 to 27% but you really have to have patient long term management thinking to get the full benefits of what is possible. The point is power has been delegated to the programmers and this works far better than anything else known then or now. Ok?

Here’s one of my clients in London who looked at what Raytheon did and quite simply repeated it. They were just to a start up, just about to go out of business because they had so many bugs, nobody would buy them. They decided to copy the program and even had a 5-year time horizon reducing the bugs. They got so good at quality compared to their competitors that they took all the business going in their market from the competitors and won out. There’s a lovely case study “The age to change” by D. Colun showing how that worked with simply replicated what Raytheon had done.

Here’s another example. This is from what was now Boeing but was then McDonnell-Douglas. We found there was a tremendous problem. There were too many young people coming in, like 2000 in one year, new engineers and they frankly didn’t know what they were doing. They were creating errors and bugs all over the place. This is in aircraft engineering not in software. These principles apply to any form of engineering or planning. And this was quite simply delaying airplanes and threatening the company’s existence. We found that one simple drawing had 80 major defects per page. A major defect is violation of best practice or standard. We decided to measure this and set a standard. You had to have less than one major defect to get approved for handing out to anybody else. People started off at about 80 major defects and were told”Your work isn’t deliverable.” So they got motivated, decided they wanted to feed their kids and keep their job so they started learning the practices that the wise old man of the company had long since decreed. And they got pretty good but they have, their learning rate is about 50% reduction per cycle of trying to get good. But within about 5 iterations, this is one particular person called Gary but we have hundreds of engineers going through this curve. And within about 5 iterations they managed to learn the good practices well enough to practice them in practice and they got their work released. And that was happening within weeks across thousands of engineers. There’s case studies on that video if anyone is interested, some of the remarks of how powerful it was.

So, here’s a summary. It turns out that Mays and Jones’s method, defect prevention, prevention doesn’t mean we find a bug and fix it, it means the bug never happens. Ever. There’s nothing defined. That’s the

best condition of all. And it turns out that within about a year 50% of the bugs that normally happen will not happen. And as you will improve your process and put in a lot of changes, you'll move up towards NASA levels of order of magnitude.99% of all bugs that normally happen don't happen at all. The ones that do happen can be caught early by inspections. This is the advantage of catching them and it's ten times cheaper to repair because it's early days and what you don't catch early you'd catch by testing and what you don't catch by testing are gift to the user, of course. And this gives a perspective of these different technologies working together over time. One thing that I'd like to highlight here is some experiences from IBM in various places. On the very first line there in a 30-month period there were 2162 changes made to the working environment of the programmers suggested by them. In other words, there's not one big idea or which is what mangers tend to focus on. The big idea from the consultancy. We should go Agile, or go Lean or whatever it is. Get level 5. It's actually a whole lot of very small practical ideas about physical working environment, very largely noise, office space, when you can come in to work, support systems and ..So, the big idea is that the mini small ideas seem to add up to tremendous real improvement. And the big ideas from management tend to fall flat under feet because people resist them. People don't suggest ideas that they would resist. They suggest ideas they'd like to do, tools they'd like to have, working environments they'd like to have. So this is one of…management suggested change, there's no automatic resistance. So these dumb managers suggested stuff and it isn't relevant for us. If we suggested, we're very likely to suggest stuff we want and will accept and the whole problem of change resistance is dramatically changed.

So, here's a summary. Developers are better at managing their own work environment, than managers are. Directors should not design the work environment. The developers should evolve the environment through their practical every day deep personal insights about why have I created a bug or a bad requirement or whatever I've done. And they should take responsibility for their own situation.

So, we are going to turn our attention to a related subject which is delegating through the developers the actual detailed design. You are like delegating to them an architectural role. And the idea again is who decides design. Well the customers and users say I want this and I want that or often in fact although we may call it a requirement they're really designing the system and telling us what to program. And salesman representing them may do the same thing, and architects by definition try to figure out what design we should use in program. The new idea is put the programmer in a loop, a tight loop, maybe a weekly loop or something like that. You might like to call it a sprint and allow them to make the design decisions, to program them, to test them and measure them to see if they are any good. Leave that whole decision making to them. Refuse to accept anything resembling a design from these people on the basis they are not qualified, they don't have an overview, etc. They are just amateur designers. We've been allowing the amateur designers in for too long. This..You are capable of being professional designers. These people do need to give a signal. I want it to be more user friendly. But how it's going to be more users friendly is something we need to figure out, technically.

So, I'm going to give you a case study on that, this is my favorite client who's done everything better than I could possible teach him. In Norway we met as a gang of about 13 developers and 3 testers, here is even Peter the founder of the company and Trone, our test manager who is our hero and people like that. The interesting thing about this team is, 5 years later the same team is still there. You know why? They love their working environment, because it has empowered them. And they know they are going any place else isn't going to empower them so they just stay. Wouldn't you like to have a working environment you just didn't want to leave, it's so nice. The working environment not the perks, the working environment itself but that's what you are looking at it.

When we meet them they'd been 8 years in an international market with clients like this. They had 50% of the world market it which isn't bad for a little Norwegian company from Oslo. But they took an advice to dump the 1500 requirements in a queue which were largely designs by the amateur users, dump them

forever and they did and focus on some high level requirements set by the marketing director. We need to be more user friendly and things like that. Here is an example of one of them written in my planning language, planguage, which says ok you have subset for usability, productivity defined, time in minutes to set up a typical specified Market Research-report. The old release 65minutes, we'd like at least to get down to 35 minutes, this is every user every day anything and in our dreams we'd like to get to 25 minutes, that would be fantastic. This is the requirement coming in from the outside but the programmer stare that and say ''How are we going to design the system to get to 25 minutes?'' is the question. And they make a decision to try to do it, program test it and see how it goes. Ok? This is just saying that they completely left the idea of the detail features and burn down stacks and users stories and all this kind of stuff and they decided to focus on the what I call the top ten most critical objectives at all times. Tron weaved up a little tool we've seen far more advanced versions of it than this, but this tool is a snapshot of week 9 out of 12 weeks. After 12 weeks, a quarter of the year they released it to the world. But in the 9 intervening weeks, we call them value delivery cycles, something like a sprint, that sprints are not really focused on delivering value, they're focused on delivering code. But we focus on delivering value. So here is the requirement we looked at where these are requirements, this is the one we looked at, currently 65 minutes and every one is a sequence here. We're trying to move from here to there, we want to at least to get to the 35 minutes. We'd love to get to 25 minutes, so the and we have a 110 working hours with about 4 people working on this agenda for a quarter of the year. Ok? The team chose the beginning of step 9 to work on this one because nothing had been delivered and improved from the previous 8 weeks. That's a similar problem here and here, they've been working on the things that have been numbers. These numbers there, one hundred means they've met their goal, 200 means they've done twice as good as the goal, 50% means they're half way through the goal. So this gives the project team an agenda, basically the weakest links in the chain need to be worked on. This is protocol dynamic prioritization and the team is at liberty to prioritize whatever they want. Nobody is going to tell them what to do. Their only agenda is at the end of a quarter of a year to deliver this goal numbers within this timeframe.

So, that's the agenda, get to 25 minutes. They've decided to work on that that week. They have a stand up meeting and within a half an hour suggest about 12 different designs. One of them is market information recoding and the question we ask the very design is ''Ok. You think that's a great design. How many minutes of the 40 do we need to save do you think we'll save?'' And the estimate here is 20. And that was the best one. And they also felt it would take them their 4- working day cycle to do it so they didn't have time for anything better. 20 minutes is just of course just 50% of the way towards their task. Ok? Go, go. And, so they coded and implemented that and Microsoft usability labs kindly decided they would for free measure all the usability measures and Microsoft usability labs overnight Thursday to Friday said ''You've saved 48 minutes, which is 95% of the target. Congratulations the design was twice as good as you estimated." They then spent a weekend trying to get to at least a hundred and over the weekend one guy and not the whole team got to 20 minutes which 12,5% more than necessary. So this is sordid, no longer has priorities and weeks end they can do that or that or the third one. They've got three more weeks to play with. What they..by the way after 9 out of 12 weeks, 75% of the time they have achieved on the average 91,8% of all their value. In other words they are ahead of the curve.

This is the new burn down cycle. It's a value cycle. It's a value to cost cycle. We have coded cycle which may or may not give you the value of what is worth. We are changing focus to numeric values as understood by our users and customers primarily. So, that they also for fun compute priorities where red means you 'd better worry about it. Green means you're cool don't even think of using any effort on it. And yellow means your tolerable level you've got the minimum worst acceptable case but you're still not green you've not reached your goal level. So, this is such a logical system that priority can be computed directly by the spread sheet. These are the 25 goals they've set for the 16 people and each one of these is a 4 person team approximately working in parallel for a 12-week cycle. And you can see from the cumulative numbers here that all the teams are largely up there at about 90% of achievement within

75% of the time in other words well ahead of the curve delivering good value for money and time. This is the weekly cycle and I'm not going through in detail we develop this part in very large use of the Evo method at Hewlett-Packard. We have a case study publish by Hewlett-Packard on that. They innovated a little bit what the chief technical officer Peter was going to do and what Tron was going to do was QA manager and one would have to study that in time.

Of the 25 things they did set up to do on the first quarter, after by the way one day of training I should add, ok, not two days, you master these in one day apparently. These are some, these are the 5 of the 25 that have the both sensational change. For example the first one time for system to generate a survey these are gallop poll kind of things was 2 hours gotten down to 15 seconds. This is like a free release to the user. Imagine if you're driving to work and it takes an hour and your drive back takes an hour every day and the new I-phone apps which is freely downloaded says you will totally use 15 seconds in transport every day. Be my guest! That's what the user is forgiven of sensational results and they got about 25 good results. Ok.

A good sign of a method is when the people doing and loved it and feel empowered and understand it. So, here's Tron's comments on the fact that they really liked it. They love to stand around the corridors say ''We made this product great and therefore we are the essential element of this company''. Ok. Not the directors, not the managers, we are wise enough to make quite sure that the developers got all the credit. This is, using their own survey tool, this is how dead pleased their customers were when they're just about to get their second release. And this on the second quarter are some of the big numbers they achieved by turning to other things and keep on turning out quite sensational improvements by just shifting to things not yet done. So that makes the case for delegating the power to the programmers for finding the design. You will not see such numbers in any other context, I believe, please enlighten me if I'm wrong and you've got the numbers to prove it.

Now, the same team two years later did something, I think pretty interesting. They said this is really successful this idea of engineering towards multidimensional goals and empowering us to find things, works great. We have a problem, they have a problem before we've met them, not surprisingly 8 year old startup, hacking together code to get on the air and served the market. There was no grand plan of elegant architecture, so it'd be easy to maintain. Ok, the first thing we suggested is that they invested a little, one day a week in cleaning up the code, but that didn't work very well. So, they themselves figured out something much smarter to reduce their technical debt. And what they did was, they made a list of the technical debt aspects in another words attributes to the system that was made more or less easy to change and test and debug the system which are here. They set numerical goals and then they gave themselves one week every month called the green week where they worked on their own programming environment. Again there are teams in their own environment, ok? And they, it's exactly like the Raytheon in principle, but they are specifically given a week a month guaranteed every months or they will be nice to themselves, the programmers and developers. They decide what their agenda is they decide the design, they make it happen one way or the other and they measure whether they're in fact reducing their technical debt as described by a set of things like that and a set of rules like that. I thought that was bloody brilliant. I wish I thought of it myself. I didn't but at least I can tell you that my client figured that out. Devote time to instead of refactoring which only at best messes with the code, they're messing with the whole system including testing of everything they do to maintain the system. So they're engineering reduction of technical debt. And I don't think you do that by just letting a programmer loose on the source code. Ok. So, maybe I've said that well enough? Let's go and take a look at another historical great, something called the clean room method which was developed by Harlan Mills and IBM in the 70's. How many people say ''I know what the clean room method is?''

One, two, three over there with the speakers group, ha ha-ha. Ok, rest of you innocence, ok, Thomas. Let me introduce you to clean room, first we start with Harlan Mills, he's got an interesting challenge, he's

for me like Leonardo DaVinci, of software engineering, the genius. And he was on IBM Federal Systems Division with all that rocket scientist quite literally, space military rockets. And IBM's problem was that every time they won a bid they were the lowest bidder and every time they took that they lost money that is, they spent more money than the lowest bid. So they were just continually losing money on all of their contracts and they finally said ''this has got to stop'', either we find a way of earning money when we win the lowest bidder contract or we get out of the business. Harlan you have some time to figure out, this is the smarter way of treating the situation of, we just have to fix the income from the government and we don't want to cheat to get the rest of the money we'd like to have. So he actually worked for ten years and reported in IBM system journal number 4 from 1980 quite extensively, that's web available.

Now, there are two case studies here, one is the lamps project in navy helicopter system and the other is the NASA shuttle ground software. But, now you see what is clearly an incremental system. They are shipping the system in 45 incremental deliveries. That means 2%, that means once a month every month for 4 years. This is clearly what we today call Agile. And they're very certainly using numbers and engineering for qualities like availability. Very clearly as I will show you in a moment with Quinnan, designing and engineering on every cycle. But let's focus on the bottom line here. Very few late or overrun means budget overrun, financially overrun. In other words if you overrun you lose money, if you don't overrun you break even or earn money. In a very few such deliveries in that decade, and none at all in the past four years. They're building some of the most complex high quality systems on earth, space and military, to the very highest quality levels and yet they are achieving that, on time, under budget and they are making money. This is perfect project management for software. This is something, everybody should learn the recipe and repeat it. This is Agile at its best. It's Agile but it's engineering, too. These are engineers they understand the quantification of qualities and management to them very well. They are not just hacking code. Ok?

Quinnan, colleague of Mills is the sort of architect designer. The point he's making is, every cycle they analyze, how did this design work? Did it really give us the availability we thought? Did it cost too much? If negative feedback, find a smarter design. Engineers have a paradigm called design to cost, ok? So, they call this dynamic design to cost because they're doing it every cycle and they try to figure out what is that can reduce the cost and the timing and still give the high quality. If necessary they experiment, measure and get it right early. That's how they deliver on time, under budget at the highest cost levels. They learn to live within their budget, ok, there is no planning poker or any childish games like that here. They're told what their budget is, they're told what their deadline is, they walk in. They have to figure out technically how to get there and they have to do it evolutionarily in small steps. This is Agile as it should be, is what I'd call it. But anyway you can study Quinnan in detail on web available things. Ok?

Design is an iterative process not an upfront process. Again these are the programmers involve in the loop, you know, not the managers at IBM or the directors anything like it. They are estimating in the loop, not with planning poker to begin with, estimating based on knowledge, facts and measurements and then new hypothesis, new trail, see if we know what we're doing this time around.

Here's another client of ours, he e-mailed me in 2011 and he mentioned he intended a 3- day course at City Group in 2006, I didn't remember but I thought it was nice to have a new book review. He said there is a book review on my blog website but at the end of the book review he said ''By the way I didn't just read Tom's book. I've been living in it for several years." Ok? And then he told a story of what he'd done after he'd been in the course there. Basically he said our bank, you recognize City as being partly Polish related and I spent a week here in Poland teaching this gang the methods, I don't know how well that stuck but we did try. And basically the problem was they were tracking an amazing amount of stuff and they were very proud how good they were at tacking everything happening with software. But they were not tracking value delivered to stake holders. They were tracking functionality programmed and bugs and time, ok. So they are in a sense they're tracking the wrong things but they weren't even aware

of it. They had the illusion that they were controlling risk but they were just controlling programming not the risk of failure to delivery, to deliver what the bank actually require. So, instead of using our Evo method which is the Agile method that does it with engineering, the Agile method that quantifies values, an Agile method quite identical to clean room and quite identical to Lean startup in principle, ok, measurement multidimensional stuff in cycles. Because he has low profile, don't run if somebody uses this radical method you never heard of, just do it and if it works you will be forgiven, is another change tactic here. What he mentions here is that we talk very carefully, many of those things called requirements, that you are required to program are not really requirements they are largely design and the programmers should be figuring out the design, not the users. Users just stay at a higher level, I want to save time doing that task, they shouldn't be designing the screen and the graphically user in the face, all that stuff. So, basically he took that lesson of separating what users wanted, the achievements they wanted and let the design himself be part of his team. So interesting enough the requirements remained unchanged for the 14 months of the project. Everybody knows the requirements turn 40% of the year, ok, but this is design turn. Indeed. They had plenty of design turn cause they plugged in design that they thought will work, but the designs didn't work so they had to find some new ones. They were in the exactly same mode like in clean room. Ok . And a bottom line is successfully live, 800 users worldwide, undoubtedly some in Poland here, and big success by sponsoring stakeholders. Ok.

In fact the same, Richard Smith is going to present a small conference, private conference I'll hold in London next week, he's experienced with Japanese banks. I just got his slides last night, he is still doing them.

Is it hard to change? Sure, it's incredibly hard to change. Especially if managers try to buy top down command and control. It is, according to my evidence it's much easier to change if you delegate the change process to the people who know what's going on and have the power to make a difference and those are called developers. Wow, I've actually time left.

Just for fun, last night, I have a hobby, is whenever I do anything I like to find the 10 principles and summarize it. So, just for fun in the work environment we need to delegate to the doers, that's you, we need to measure the improvements, we need to let the troops, that's you, identify the common cause defects, the things that are wrong and create many bugs and problems, we need to let you suggest what is the root cause, like I didn't get enough sleep, and we need to let you try out, ok, come to work when you slept enough. Ok, whatever it is.

In the product development area the troops, as in confirmed, to choose the value goal to work on, in other words which one of these goals will work on, that's a choice. They need to estimate the power of their ideas, not throwing ideas let's throw an idea and say "Let's program and see what happens", but estimate in advance it will save 20 minutes. We need to let them decide which of many designs they choose to implement as a team internally and not let any manager or customer interfere with that decision making process. We need to let them measure the results this week now and total to date so they know where they are and getting towards the goal levels. And we need to credit them for the results and reword them for their success. My suggestion was always if they were done in 9 weeks all the goal levels shouldn't they have 3-week holiday, that's been turned down time and time again. The nearest we ever got to reward like that is that they had 3 weeks to do training of their choice at any conference in the world. That they thought they could get away with. They said we can't give them vacation. Everybody else in the company would be so jealous it wouldn't work. Ok, nice idea. Ok.

So, the revolution has been here for decades but maybe you didn't learn about it but I've tried to inform you the best I could, give you a lot of stuff to follow up so you know. Was it Karl Marks's ''workers of the world unite'' something like that. You guys remember that stuff, anyway. Programmers of the world unite. Finally if you really want to read a tough book with the deepest wisdom I've managed to collect in about 40 years I'll give you a free digital copy if you send me an e-mail. If you don't want to read anything

heavy I've tried to put some lighter reading, 14 papers on Agile including many things I've been talking about at this address although we are still struggling with the Internet trying to get it in there so it may be a while but sometime today that address will be valid, with the first person who finds stuff there check it out let me know. Copy of the slides are there and copy of my papers.

And a small miracle has happened. I'm done. Thank you.

Stephen Parry: Keynote - Three principles for designing adaptive, innovative and engaging workplaces

Stephen Parry: Thank you very much, I just want to say thank you very much for the invitation to come to your conference and speak. The audience and what this conference was about, mainly people, was very exciting. What was also exciting was the fact that it was in Cracow which is one of my favorite cities, as well. So, it's great, it's great being here. I've been here so many times. I have lots of friends. To come here it's been fabulous. So, I am here. I'm between you and getting home so I am going to try to make it a little bit entertaining if I can. It's a tough subject. I am trying to put a little bit of science into the whole theory of change.

Now, we've had a number of Lean changes for many years, for 20 years now and everybody has been talking about changing the mind set, changing the culture and all these things. But when I tried to look for some academic research on this it was very bitty. So, a number of years ago, in fact in 1999 when I was at Fujitsu, somebody from the London School of Economics phoned me up and said " We are doing this psychological research study on how organizations tick. Can you come along?" and I thought this is a good opportunity. Because we were going through a Lean change then, something that later became call center respond. What really surprised me about the research is that the academics and the work psychology people have been measuring cultures and work climates for a very long time and for the very first time I've got a glance of what a work climate for Lean looks like. And that set me on a ten- year journey with a number of universities to try and understand what it is when we talk about change and particularly Lean change and particularly adaptive change where we are engaging work force. What are the elements that really make a difference? And I am here to show you some of that today only briefly but to show there's a lot of foundation scientifically for some of the things I'm saying but also some of the things people have been saying up here today and yesterday. So it's of no surprise that some of these things.. we know but we don't really have evidence that we can prove it. I want to support that with the evidence now.

Ok. So, there are three principles to Lean or an Agile organization and there they are: it's people, people and people. Yes, people, people, people. It's people that make the difference, it's people that create value. And Lean and Agile are for the knowledge economy. And when you deploy the principles of Lean with the respect for people, giving people freedom to think and reflect and to experiment, wonderful stuff happens. Now we all know that but now we now have science evidence that tells us why. So, I am going to put some of the science in context with a case study. The case study comes with a slight warning in that actually it is one case study made up of three larger case studies by taking the best of those to show you an illustrative form. Ok? When you look at the case study it's not one, it's actually three that have been brought together to illustrate the point. Each one was very, very successful. But for learning purpose I've put the three together. If you want to follow me I am Leanvoices on Twitter, I have a blog on leanvoices.com. A lot of what I'll talk about today will be on leanvoices.com. Ok, so, let's get into this.

First of all I want to talk about changing perspectives. I will illustrate that in a moment. The thing I want to focus on is this thing called work climate. I'm not talking culture and I will explain why later. Work climate is something that we can all feel and we can recognize and I will bring that out. Respect for people and a plain free environment, this is often misunderstood. So I want to give you some of my view on what a blame free environment means, what a respect for people environment means, and it's not what you think or some of you think anyway. We need to move from what we call mass production thinking to Lean thinking. Mass production is quite clearly coming into very silo, we are hit in the numbers, we are

maximizing the output from people and we are focusing on the people and not on maximizing the output from the people as opposed to Lean thinking which is looking at how the whole thing works together to maximize the value to customers and we are free to experiment different ways of doing that. In the mass production environment you are not free to experiment. It's disciplined compliance in mass production compared to disciplined experimentation in Lean. A completely different way of looking at business and managing people and I will bring that out. We need to… I will demonstrate how we measure the work climate. These are busy slides, there's a busy slide warning on them quite simple because something as complex as a way an organization thinks it's very difficult to get down in a couple of slides. I've tried my best. Then into the case study itself. There's lots of references and resources and recommended reading at the back. I will say when I am talking about references and sources I have been absolutely, incredibly lucky throughout my career to met people that have helped me in my journey, mentored me, and that's been terrific. So, if you want to find out where that background is, please take a look on leanvoices. I have it fully documented there.

So, I just want you to watch this video carefully. This video comes from the 1970's and this is really about, this was put out by the Guardian newspaper. It's available on YouTube so you can use it in your transformation workshops. It illustrates the point that I am talking about. I will talk over this as we go through. So ,this guy has seen a car, he doesn't like the look on them so he is running away. Is it a gang land thing, who knows? From another perspective he is running down the street, he is going to hit that guy, steal his bag. But, let's take another look, the bigger picture. We see something completely different. That is the essence of changing the way organizations perceive the outside world and how they perceive what goes on in the inside world. In mass production, we each have our own single points of view and we do our busy , busy bang, bang stuff and we try to make more bang for the back in the hope that it will turn out OK. But we are all doing that independently. What Lean does, it allow everybody to get that bigger picture. In fact, without the bigger picture we cannot work. And some of the survey stuff that I will be showing you will be telling us how organizations see that bigger picture or don't. It's really important that we move to seeing the bigger picture. What we are trying to do is we are trying to liberate the willing controversy of our staff because it is only the willing controversy that creates breakthrough, cold knowledge, and insight. You can't command and control somebody to be brilliant. Ok? Somebody was saying this week "I make great code in the shower". I come up with my best ideas when I am relaxed, sometimes in the shower. You can't command and control those sorts of breakthroughs in thought. They come when they come. Musicians have a word for that, it's called the muse or in the groove. It's exactly that. So, in order to have that freedom your performance is a matter of choice, being able to do different things at the right time which is a matter of freedom with a power to do in this case what matters to our customers.

But there are things that get in the way. There's the red tape from the job role, the processes, the procedures or the methods that we are told to follow. The performance measures, the style of leadership, all of these things get in the way. In a mass production environment if we step out of the line there is a lot of blame. So, we are going through a transformation, we have to get these links, we have to break them down, and we have to do that in a systemic way or systematically. We cannot just go out to changing things because we don't know the cause and the fact and a lot of things. I would say that the biggest thing that would change your organization very, very quickly without thinking about all of these change processes is go and change your measurement system to end-to-end. And is about outcomes for your customer. Don't measure the quantities of things. It is how things perform together, end-to-end and you call everybody to that goal. You will have a massive change in perspective. And that's one of the places that I started. In fact, I don't start a transformation until we've measured that. What is the performance end-to-end for what matters to our customers. This is the first principle of Lean. But very few people actually do that, they do the other principles, the value stream map, their process map. They look at all the steps that don't create value, chop those out and make things go faster not realizing that what they were making before

didn't satisfy the customer . Now they are making it faster and what they've done is use Lean or Agile and created cheaper, neater, faster waste. And it's cheaper and you get this delusion, a self-deception, a seduction that works. We're still, we remain automated, we automate work that need not be done. We're still, we send it offshore to low cost labor economies so that they can process it cheaper, neater, faster waste. And the amount of waste that is in an organization is absolutely tremendous. But I am not going to focus on waste today. I am going to focus on the waste and human potential, not the waste in resources.

So, how do we break this down? Well, it's people, people, people. What I mean here is your clients or your direct customer or your clients' customer. That's one group of people. Engaging them and deeply understanding them on what they are trying to achieve, what is their purpose in business, ok? Deep understanding, this is not the same as the voice of the customer which you get as a sigma. In Lean the voice of the customer means he is in there with you day in day out on your shop floor working through the transformation. We don't need to write it down. He is there with us, we go into his environment. The one thing about having the customer in your environment as you're going through the transformation is this: it stops all the politics. It really does. Just try it. But you have to be brave enough to go open book and say "Look, we don't do everything perfect. There's lots of things that we need to do and get the right customer involved in doing that." So, we need to engage, deeply understand. And that's how front line operations, I mean in sales people, account teams, engineers, anybody that touches and understands the customer. We need to grab that information. And we share it, we learn from this. We share it end-to-end. We take this end-to-end perspective and the leadership then have to listen and adapt, not command and control. What is this data telling us, what should we be doing, what are the opportunities that are there? And then we should be directing our people to actually go after new sorts of value, to experiment new products and services at the same time we are delivering the other stuff. Most organizations can't do that because there is too much waste in the system. People say to me "We can't do this, Stephen, because we don't have the resources" and I point out "You do have the resources. You are just doing the wrong work." So spotting what the wrong work is really part of the improvement cycle. So improving, then we adapt, innovate and improve. This is using every point of contact with our business to systematically capture what's going on in the customer's world, making sense, measuring how we deliver to what matters to the customer and using that information to really drive innovation. Innovation comes from having low levels of waste. Innovation comes from having a work force that is capable of solving the problems at scale.

We heard earlier on in the week about A3. I love A3 but it isn't practiced as A3. It's a tool and we'll talk about tools in a minute. So, whatever your changed program is, somebody comes along and says" We're going to do Lean , Agile", and I am not going to say the other word. All of those things and somebody says "Ok, we'll base line our current performance. We think we are going to get to this transformation and objective up here. This is what we've heard, this is what we can do." So, then we put in the new tools and methods for improvement, we put them through a ship deep. You know what a ship deep is? Put a ship in and there's a crowd and out they come and they are supposed to be experts the other side. And then we take a look at, well, the new performance and everybody zip it, mystified. Well, he haven't had a steep change, it's just a little bit. So what has gone wrong? We followed all the books, we've read all the book. We are diligently using the tools and the methods. So, what is going wrong? And what I suggest is the gap. The performance gap is the way that we manage our people. Let me explain. Their perspective is wrong. It's within the silo, very often they used the tools and techniques to improve the silo and everybody knows you sub optimize the whole. But everybody is sub optimizing the whole. You've got subpotmiziation times ten, times twenty, however many functions you've got. It further fragments the business, not joins it up. It certainly doesn't lead to flow and it creates more waste.

So, the perspective from working in the silo to horizontal as Womack, Jim Womack of the Lean Enterprise Institute famously said "We need to move from vertical management to horizontal management". And that changes everything, changes our measurement system, our perception on the customer, our knowledge of the business and how we collaborate. Everything changes. So, if everything changes, we may have to

change the work design. We definitely have to change the measurement system and the behaviors that will come with it. Now, I am a great believer in that you get the behavior you're designing for or in some cases you fail to design for. Behavior is not a choice for most people. People are forced to behave in a particular way because of the climate, the measurement systems and the drive to certain targets. So, behavior is an outcome of your design and if it is not a good design, you have bad behavior.

Ok. All that above the yellow line I call the Lean work climate. If you don't get that right you will not bridge that gap and all those are related to how people work together, how managers manage, what they pay attention to. I will start illustrating that in a moment. But to do that we need to focus on four things: engaging and understanding, learning and sharing, leading and decision making, improving and changing. That's the learning cycle. Let's talk quickly before we get into the main body of this presentation, about this respect for people from first to last. Lean is not about cutting people, it's about growing people. And if you've grown those people that they can take out lots of waste it seems like a very wasteful thing to get rid of these smart people. You're investing in your people. Lean is an investment in your work force. I won't read through all of these. I think my favorite here is about as a manager or a coach which a lot of the Lean or Agile experts call themselves. Coach is helping them to see. There's a saying here "There is a world of difference between helping people to see and telling them they are blind." Very different. It takes time and conversation to grow. Lean grows people. A3 grows people. It's a talent management system at its best. It has nothing to do with processes. If you've got talented people with the right measurement systems, with the right incentives, everything improves, including your processes.

Ok. We try to create a blame free culture but I do like this quote and we've seen a lot of Lego slides this week. So, Lego is obviously in fashion. This is a quote from the CEO. We do blame for two things as it said: "Blame is not for failure, but it is for failing to help out when asked". Now I am sticking in my silos, I've got my target. Helping out is normal. The other thing is failing to ask for help when you know you are in trouble because you want to do it independently. Work in an organization is a team game and I know in that from the conversation in the opening speech I'll refer back to that. I am an introvert, too. There are lots of us. I like the science to do my talking for me. I like engaging in one to one rather than big events like this. But everybody needs collaboration. No man is an island. But even the introverts need to ask for help when they need it.

So, what does this engaging look like? I am going to take you gently through this now so we can characterize what engaging in a mass production organization looks like towards an Agile or a Lean organization. Basically engaging here says "Does the job design allow you to even talk to your customer? Or you're designed out?" The system designed you out of talking to your customer. Gone is the first principle of Lean. Very often is the product manager that does that and we hope it gets it right. But in Lean organizations we actually got more people so we have different perspective on that, a deeper understanding, so it's not lost in translation. And there's more involvement. To what extent can stuff modify the solutions in response to customer needs? Now, in development I think you can do a lot more of that but in other organizations and other industries you can't. The products and services are fixed. This next one gives you an idea. Is everything in your department forbidden, it's locked down? You can't do anything unless you ask permission. All right? Or is everything permitted, you can do all of these things . I trust you, you are trained at the right level but don't touch the soft compliance stuff over here because we'll be in legal trouble. But for the rest of the stuff, you see, it's a completely different mentality of management. You'd feel differently in that organization.

So, what does learning mean? Now, I've got this information about the customers. Do employees routinely share it within their team at the functions and even with the executive teams? How often do you see the executive teams? I spoke to one organization only last week and I found out that some people in that organization hadn't seen the chief executive in their building for three years because he was going to places. He was a great advocate of Lean yet he didn't go and see the people where Lean happens. So,

we changed that. But that came as a complaint from the staff. What is the management focus? Employ utilization, busy, busy, bang, bang, trying to get more time and utilization out, work intensification and work task intensification. Or is it creativity, building creativity in your staff? Focusing your staff on your customer outcomes. Does it do what is says on the ten: problem sharing, learning, sharing knowledge and collaboration, actively encouraging collaboration? Very often in the command and control environment people say "You are working for me, don't go and talk to the other guys." All right? We've got to meet our goals, don't waste your time elsewhere. But there is no time wasted in collaboration.

So, leading. What type of leading are we talking about? I believe leadership and I think we've heard it from a number of people, leadership is an activity, not a position. And Lean and Agile require people to lead at every level. But to lead at every level people need to know it's fairly risk free. This is the important thing. We reserve the right to be wrong when there's evidence. And that is a strength not a weakness. But unless you've got the prediction right, and you want to alter it you can't because you are not seen as you're not in control in a mass production environment. And that's last thing you want to do in a mass production environment is show that you don't have control when in fact the mass production has less control than a Lean environment as we will demonstrate.

So, improving. Can your staff improve the end –to- end process? Can you improve the products and services or you just sit there do your busy, busy, bang, bang and hope everybody else is doing the same busy, busy, bang, bang and it all joins up somewhere. But nobody is doing the integration. And then we put supervisors in to get a bigger bang for the back so we get busy, busy, bang, bang, bang, bang. And then we turn around and say we have no time for improvement. No, because we've gone for the utilization. We've taken people out as soon as we can because the utilization of staff maximizing the output of the staff instead of maximizing value to the customer is its philosophy. I'd rather have people sitting around doing other things as long as we are maximizing value for the customer. Because that's what pays the bills at the end of the day.

So, I am going to talk about a climate. I was comparing engaging, learning, leading, improving. I was comparing what a mass production in a Lean environment looks like. We know that, we can feel that. So, I am asking you to think about this thing called the work climate. It's not a culture. A work climate sits at the surface layer of an organization and these are sorts of things. Great work only happens when the management is trustworthy. And having a trust strategy is critical to your success. If the management are lying, changing the mangle all the time, not standing up for things, particularly driving knowledge work to excellence, if they are not standing up for that you will come back. Your commitment will no longer be willing. You'll hold back especially if there's going to be errors. You're going to hold back. You don't want the attention. So the climate now is gone cold and sometimes actually is not a cool environment is a red-hot environment because if I get out of sync with my management it becomes red –hot. You want to avoid that. So, this can be red-hot environment. So that reduces your performance. Or it could be a pretty cool environment. It's pretty relaxed. We're taking time for reflection, we are actually producing better quality work, we are getting deeper insights. All of these in knowledge work is about insight, creating better things. And the environment is that thing that forces us to behave in a particular way. If you've got that toxic environment, you won't behave in the right way. But in the cool environment, where it's safe, you will succeed.

So, this is a busy slide. These are the dimensions that we measure for engaging, learning, leading and improving. It is a scientific study, it's done online. The whole company gets involved. So we are measuring how the organization feels to the people, the managers, and the leaders. Is this a good climate? Now, a climate… The good thing about climate is it is easy to change. The bad thing about a climate is that it's easy to change. So, let me give you an example. You had a previous boss, Joe was really great. He gave all the time, the output was good. It was great coming to work. There was no challenge in getting out of bed. You worked overtime. You were putting more in. Then, there is a new boss. He says "I just want two more lines

of code.", all these volumetric numbers driving the efficiency. Cost at all cost. Suddenly the environment doesn't feel the same. The work climate has changed as quickly as that. That's how quick a change is. A new boss comes in and it can change it. But I am going to make a distinction here between work climate and the culture. Has the culture of the organization changed? No. It has just been pushed back a little bit while this new boss is there. The culture that you've learned is still there. If that bad manager stays in place for any length of time, that work climate will slowly become the culture. Ok? So every culture starts off as a work climate until it becomes codified, all symbols, all recognition. So, what our challenge is here is whatever the culture, work climate can change quite quickly with the right management approach. But we're going to have a lot of trust to do that if there's been a bad climate previously. But establishing the trust is one thing and get the new work in practices on the basis of this trust the climate is changed. This is good. But the old bad culture is still there and can swamp back. So the job of you guys and your managers is to hold on to the new work climate. Once you've got it, don't let it go. You hang on to it with your life. And that takes courage. It takes courage, truth, honesty and tenacity. And some of the work that I do with leaders in organization and I mean people of lower level of the organization, the leaders in the everyday work. We teach them how to have that courage, to look at the stories the company says about itself, to highlight the in authenticities. And as a group you can do that. You become the conscience of the organization until it becomes the consciousness of the business. This all sounds like touchy, feely, soft stuff. But this is your work place. Which one would you rather work in? I know which one I would rather work in. And it is your choice if you are sitting there waiting for the big boss to come along and say "We want to change the work climate." It is not going to happen. You can make a choice to start, get the right manager in the right place and start. You can have your one microclimate in one area, just need to start. I'm going to show you, some case studies in a moment where people did just that. But let me take the technical staff out of the way. If you are interested in the technical stuff and how to measure climate this is it. If you are just accepting the concept of climate you can switch to screen saver for the next two minutes.

OK. So these are the questions. I am going to look at engaging and learning first then leading and improving. So let me just say in the center here this is neutral neither one nor the other. But going better mass production, or better Lean is a completely different route map as I will demonstrate. This is one company, these are two development teams in two different countries doing exactly the same work. They are doing Agile, they are doing Scrum, they are doing Kanban, they are doing all of it. And they have somebody checking on a check sheet. Are doing it? So comparatively they are using the same method, but one team is massively outperforming the other, it is much more innovative, less waste as a great place to work. In fact everybody wants to go there and the other one they want to get away from. So let's have a look what the climate survey said. In the one location it was mass production. Let me just explain this. This is a weak thing, it is neither mass production, is a bit leaders have no control. It's a standardized, a standard mass production environment which you could have an excellent one as well. Ok? So a good standard one with a good base, mass production will be red all the way down here. Ok? But we are not. It's fragmented. It's not even a good mass production. It's not even a good mass production. So, let me explain. They have autonomy within their work, it's not too bad, they do some customer facing activity, of course. Intelligence gathering, very little intelligence gathering about the customer's world. Ok? Do they share intelligence within the team? Yes, they do. And they do it really well. They do it really well within the silo. Ok.

Organization and understanding, next to nothing. So, how could they share if they don't know where the other teams are? They don't interact anyway so their knowledge about the organization is very lacking. Remember seeing the big picture of your organization? It is vitally important to know how it all works. Sharing intelligence within the function, excellent and with local functions. great. Sharing intelligence with senior management we never see them. Now these are not my words, these are the words of live people just like you, just like you.

So let's look at the other team on engaging and learning. Remember they are doing the exact same work once they are performing. Levels autonomy still not great, costumer facing activity was good but in a Lean way they were looking for new services or products. They were looking for what can I sell you next off my shelf. That was a push. Here is a pole, intelligence gathering from the costumer environment. What's going on? Very little here. Sharing that information with the team. Their organization understanding, huge, which means if I've got a problem I know where to go. I don't need permission, I can just go and talk, pick up the phone, have a quick meeting with them, whatever it is. Sharing within the function, sharing with other functions and sharing with senior managers. It was routine, that the manager went to the work place and said "Hey guys, what's going on, what do you need my help with?" So they were removing obstacle here. This wasn't even a good command and control organization. This is what I call a command and hope organization.

Leading and Improving. Again performance management was weak. They weren't even doing good mass production so there were lots of loose things. The leadership approach was almost nonexistent. Responsiveness to customer issues was extremely high. Absolutely! All right. But implementing ideas to better serve costumers was low, this is strange. Not until you realize that what they were responding to is the customer complaint and once that's out of the way is back to the busy, busy, bang, bang stuff.

Improving. They weren't improving the service, the workplace and the functions or end-to-end. Look at this, end-to-end process employ functions. That's two teams working for the same organization, both using Agile, Scrum and Kanban and they ticked all the boxes. So just bear that in mind. I now want to move to the other side of the fence. We talked about the devon problems which I think is a totally made up thing, by the way, made up by the people who are trying to sell automation technology. So why better make up a problem and them give them a tool to do it with. Yes there is a problem between development and ops, but as we will see there is a lot of problems within every part of development and every part within the ops they are just as fragmented. So why pick on this little lonely one? OK. Is the same thinking end-to-end, and the reason is I honestly believe it's people selling technologies because developers are frustrated with what's happening in the operate space. So let's give them a tool so they don't have to talk to people. OK. So let's go through that.

You know the problems in development, I'm not going to stand here and tell you what you already know. I'm going to take you on the other side of the fence. We are going to their game bum and their problem here was quite simply. It's a global company, they needed to build servers and this was initiated by one guy in Philadelphia saying "I want to do an A3 on this because it's not working." And out of that came this exercise. And they did change the way that servers were procured, built, managed and everything else. But to do that they had to show the management what was going on, but they didn't know what was going on. They just knew it was frustrating and it was broken. How do we make sense of this? It's highly complex. So I am transporting you now over the other side of the wall. This stuff will be in the slide set. Needless to say it's a cross functional team from right across the business. We are getting them in the room to try and figure out what the value stream was. To do that we gave them a system. And I've used the system for a great number of years, at least ten years. It's a variation of value stream mapping. But I think this gives us a lot more information because in IT related stuff we have the technology stuff that we are dealing with, we have the technology we are managing and we have the technology we are managing the technology with. So it's all technology.

Ok. The live cycle process, phases, high level process, the area responsible for that processes, KPI's by phase, what data is required to take this activities, what tools or policy is required and other interdependencies. Just remind it, just think about that, don't study it too hard. That's just the key to what you are going to see. So, this is a team, these are technicians, mentors, architects, enterprise management people, all of them. Ok? Even some guys from procurement, from the parts, from networks, build the operating systems, load the applications, clear the space they need, all of that. So they are trying to

figure out how all of this fits together because they are all in different departments and the work is going everywhere. So you see they are starting to figure this out. And this work, from this point on is three days work, three days work. All right? These 12 people or more were in this room for 12 days. As I said to me "We are not coming out here until we figure this out." And after 2 days they said "We are not staying here anymore." Just kidding. The interesting thing was when we started talking about the whole value stream and stuff and started building this and a number of people said "Listen, my boss told to come in here and just talk about my bit and get the hell out." We said "I'm sorry, you have to learn the whole thing. We need to find where you fit in this. You need to know where you fit in this because we currently don't know." And after the first day he was then engaged and the manager said "You just tell them what they want and come back. I've got work for you." But he rang up his manager "This is far too important." So these are the guys trying to figure this out. The usual pain-point mapping. I am not in favor of pain-point mapping for a various reasons to complicate the. But they like the pain-point mapping so I said "Ok, go ahead." They learned afterwards the pain-points were symptoms, all symptoms, not causes. So we can now see the map coming up, the major phases. This is a simple, low tech, ok? This was all done with Microsoft bond paper. It's a program, it's a map you can load. So these were the different phases. You can put these the general activities, these with the different departments, and you can see different departments were using, doing the same activities so it's going in different ways. Sometimes it goes that way, sometime it goes that way, depending on who picked it up. All right? Then they needed to start these shapes here right across here were the measures that they started developing because they didn't have measures by phases. Because what we are measuring was just quantity. Was in, get it out, in get it out. But they were all doing that independently so there was no sequencing of the right server bills. Ok? And then this is the information that is required to do the work and these are the tools. These are lots of tools that people use to do the work. And I'll deep dive into that.

The thing that I want to point out, which was the revelation that that is where they took the order to build something. The order was coming from development guys saying " I need a system to do a concept, a cool bunch of things." They were testing things to destruction as well. Also we get orders because we need to increase capability. So, there's all sorts of servers for all sorts of reasons going through this. And you had all the procurement, configuration, and all of those sorts of things. But that's where the build starts, much to the surprise of everybody. What they looked at was most of the problems they were working on down here and overcome. And I'll show how they overcome lots of these problems was because they didn't have basic information here because this was completely empty. There was a preproduction phase that nobody had thought about. It was given to the techies to figure it out. Ok? There wasn't even a technology route map. Which ones are we going to select? They are free to select them. They were free to select different monitoring tools. These are all the monitoring tools. And even more, they run out, the guy was going crazy "Give me more stickies, give me more stickies." All right? It is funny, yes. But when you realize these guys were working shifts, handing over stuff. We didn't have an architecture that made sense. We didn't know where it all was. And we were all using different methods to control it. So different departments were crossing over with other people. The alerts from the enterprise management tools, that so many alerts which were all alerts, we've forgotten what they were for. But we didn't have time to go and turn them off, so they came in and we would check them. It's a false alarm, but we would check it just in case. And they learned to do it just in case because that one said "Ah, it's a false alarm." We'll guarantee that's the one which will bring the system down. And they say "There was an alert, why didn't you pick that?" You know, one in ten million. I am exaggerating, it was 20. It's really difficult. All right. So we needed to rationalize. Guys how can you do that? How is it helping security? Who will host the stuff? The security policy were very difficult as you can imagine. This is the world of the developers. I even doubt the decommissioning and disposal. Just for interest, they were coming up with measures. Some of these were quantities, as well. So make sure the production was going through the system.

These were what I call the golden five. Whatever measure you have, if you have these four, five things you

are probably on the right track. One it has to be a measure of how it is creating value for the customer. It also has to reduce the cost of the process and the end-to-end time. Velocity is the most important thing. You need to reduce the variation in delivery. You need to reduce the working progress. It's the working progress and getting rid of it that gives you adaptability. But the one that was the most important is whatever change you make it has to make the employees happier. If it doesn't make the employees happier, then the change you put in place is still not good enough. You have to make the employees' life better, not just here hit the target. This is what these guys were doing. Then they deep dived into different bills. Ok? So not only was it fragmented, not only people were using different tools, the work that was going through was being reworked and reworked and reworked and sent back. All right? But these guys had orders to get out and people were chasing them. But as we can see when we formalize that picture now, remember that, we are now formalizing it. There was a preproduction process and then the production phases. This is where the most of the guys were doing all the busy, busy, bang, bang. This is heroic work. So, let's just drill down the preproduction. These in yellow mean that we've got something but it wasn't fit for purpose. Red means that activity is completely missing. But when we look in the build in the actual production part, it is pretty good, but it could be a lot better. It wasn't fit for purpose but we can raise that up.

The next slide is very busy but you can't read the details but you can see the colors. That's what I want to draw your attention. So put in the production and the preproduction together what do we see? This represents all those tools, all those different ways of working. And yes, even they have gaps. So this is a very detailed level. These were all the departments that were doing the same thing but in multiple different ways. This is where they started to build. These red areas were completely absent because they had to work the value stream. People were giving these guys work but without the information about the standards that we were going to use, the selection of certain types of products. All of these..ok? They were improvising like hell. And they did a good job, but they had to be heroes. Heroic leadership in a Lean organization is a failure not a triumph.

We needed to go further. They needed to be able to give this information to the executive team. You imagine, you are going to tell the big boss "These guys, three days they got that information. " and there were lots of arguments, I can tell you. But they came up with this and said "This is what we wanted it to look like. We have to redesign for this, this is the measurement system." But they needed to present this data to the management team. They knew it was going got be difficult. So it's a bit like going to a mother and telling her "Your baby is ugly." That's what we trained them for, to give this information. This is the guy. But not only the map that you just saw. We were also looking at other behaviors. These are.. This is a measure here. All this stuff like our staff got trained, we've got good process improvement, a whole range of things and we said "How well do we say we do it?, How well do we say it to ourselves, the management say to us? Are we doing from 1 to 10? 10 being good. So, here it is, the blue line. As we say it good be better. This is really good. And then we say "What is it really like, really, really like?" By this time there is a safe environment to talk out.

And then we have the reality. There is reality. And between the two we have the gap. We have the gap between the perception and the reality. And then we ask a very important question. Ok. So what happens, what cost do we pay to the customer, to the business, to ourselves? What cost gets paid by us to pretend one way that reality is another way? And is out of integrity, lack of willingness, all the stuff, all the climate stuff because what we were saying was not like what we were doing. And everybody knew. Everybody knew but we weren't going to talk about it because it's not safe. So we pretend, we continue to pay the cost. So I asked a very important question. There is the pretence. It's getting to really important point. There's the pretence, there's the reality, there's the list of cost to you personally and to a black cat. Black cat? The black cat came into this because when we were doing the cost one guy says" The black cat gets it." What do you mean? Every night I go home and I kick the black cat. So for the sake of the black cat we have to change something because we are sick of being kicked. So, all of that cost, loss of integrity, loss of motivation,

boring job, loss of contribution, only do stuff when asked, the sense of futility. And then I said, "One question. If you're paying all that cost, there must be a benefit. What is the benefit?" And they thought about it for a while and they said "The benefit is this. If we take our responsibility and we get it wrong, we do not get blamed." And that's what we mean but creating a blame free environment. We have to do that. So this is serious science about how work gets processed and how we design work and who should be doing that. They change this dramatically. This is before. Those guys at that operation just engaging. It wasn't even great mass production. They could have said" we ''l go for an improvement objective." And they said "No, no, no." This has got to really change. We are going to have a transformation objective." And 18 months later, that's what you saw. Don't need the detail. Leading before mass production, decided the complete opposite to go. That's the summary.

So, in organizations we have to learn. If we want to improve this, that's improvement and it's nothing wrong with that but if you want adaptability, wiling contribution and innovation we have to redesign our thinking and we need to go this way. And Lean and Agile, even Scrum and Kanban, I correct myself, Kanban as practiced by some people I recently associated with has those elements, treating people best. Willing contribution comes from open, honest conversations. It's time to end the pretence. We've got work to do. Thank you.

Russ Miles: Pair Programming Considered Bad For Your 'Mental' Health

All right! Good morning, good morning. Has everyone had enough coffee to start things off this early? This was early to me anyway. About a week ago I was in San Francisco and I think mentally I'm still in that time zone so I don't know where that actually leaves me right now. But hopefully it leaves me awake enough to keep you up beat and I' basically the warm up act for Liz who is not here yet. She's busy doing her slides but I'm going to get you pumping and ready for Liz's act this morning which is pretty cool.

Ok, so, who writes code in the room? Who writes code every day roughly? Ok, there's some people in the room. Ok. Fine. Just want to get balanced so I know who am I speaking to, at what level and I don't want to go too deep and crazy into angled brackets and anything silly.

Ok. This talk is new one and it's something I care massively about compared to everything else I do which is obviously in comparison, I compare with nothing about. This one is a big one for me in that, to carry on what Paul was saying. I think he's absolutely right. People are going to be wear it, up going forward but it's going to be that strange marriage between people, technology processes and practices that we've been at notoriously bad at getting right in this industry. And what I'm going to talk about is actually a real tour of what I see as the major ailments of software development. I hope I don't make it too negative a start of the day. But maybe a froze down the gauntlet for everybody to start thinking about these things and maybe we can actually get better at them, and heal this strange patient that is software development.

Ok. Very first of all, I'd like just like to people more in the room will have no idea who this man is but everyone who'll stand on the stage like this will have a rough idea who he might be and this is a comedian from the UK. Unfortunately he passed away last week. He's actually a genius. I'm pretty sure it translates well into Polish, so if you haven't seen anything he's done, go and have a look. Absolutely awesome. And frankly he's one of the reasons I got in stage at all. Being an introvert myself, this is frightening and people like this help me get out there and actually do these things. I just wanted to put it out there and he unfortunately he passed away last week. Rick Mayall was an absolute genius.

So, who am I? I'm in a very privileged position to have been able to engineer my own job role. I used to be a CEO of several companies. I have this unfortunate habit of starting companies and then selling them. It's substantially painful. But, I found out that I as a CEO it's kind of a badge you get and you become this person that has to stand there and like spreadsheets and I can't do that. I can't, for love of the money, think a spreadsheet is interesting even when it tells me we're in profit.

So, my real job is pretty much either this or Iron Man. I'm working towards either end, I'm not sure where I' m going yet. Basically I'm a chief scientist which means I can be as insane as I like. It means I can enjoy our industry for what it is. It means I can do talks like this where I can talk very brutally about what we do. I don't really have much of an agenda other than to point out what's wrong or what's right.

So, this is the inhumanity of software development. Just to point out, there aren't many slides in this, particular slides, right? There's almost no slides. I am a big fan of not using slides. I think it's really nice to rather talk to you. I can put slides online just, you know, to cover it but it would be much nicer to have chats with you. And now that I've seen there are booo's all the way along the side here, this is just the precursor. After this is done come and find me. I warn you, I'm an introvert, I will hide. But when I do hide, hunt me down. Ok? Have conversations with me. I'm assuming at some point it becomes ok to have beer in this micro brewery. At that point is a great time to catch me. Ok. So, there will be absolutely no slides on this one and what we are going to do instead is first of all talk about the inhumanity of process.

So, we've spent, 14 years now, doing all sorts of efforts in how we manage our work and bring it to the table with the, I guess the target to actually adapting as we go. Ok? It's a big deal for me and what we do, I think Agile is a pretty awful term in many respects because first it has a lot of baggage. If you look at the root of the word it doesn't entirely describe very well what are trying to do. It has too many other options. I'm a big fan of adaptable. What I want to do is create software and systems that can adapt to the needs of human beings all the time. If you can adapt then I'm doing things right. So, I very much prefer the term adaption or adapting to Agile. But that aside, I think it's fair to say that are certainly some people in this room and genuinely in the industry that have a pretty good job with process. There's a lot of thought that's gone into it. I don't' subscribe to the negative view that we've just adopted from other industries, just a little bit of it. But I don't think that's all we do. I think we've innovated, we've worked hard. Process actually isn't that inhumane in my opinion. There are worse culprits and criminals out there. So let's have a look at some of those.

So, the inhumanity of practices. Ok. I'm going to start with the eponymous culprit which is pair programming. Ok. How could there possibly be anything wrong with pair programming? It has so many beautiful effects to it, right? We end up having quick feedback loops and everyone feels a lot better about the software we were building and you get to work with somebody all the time and that's human interaction and that's really good and important. I am going to draw you back to something I mentioned a couple of times so far. I am most certainly an introvert. That might seem odd to someone standing in front of 200 people. And it is. But I am most certainly an introvert. Once I finish on this stage I will flee to a corner to recharge. And when I say "hunt me down", that's almost masochistic. As an introvert you can't sustain long conversations with lots of people. There's an easy test for introversion but I don't want to get too much into character traits and what is and what isn't. But if you take for an example. If you could understand the question. When you go to a party, when you walk into a party do you sort of, even subconsciously, look to gravitate to have a one on one conversations with people rather than having big thumping conversations in the corner? Who does that? Who goes to have a little conversation on the side? Ok. So, there's a few hands going up. Ok. Great. So, it's about 50% that we will end up with, normally. 50% is usually about right for introverts. Unfortunately we have a world, certainly a culture in the west of embracing extroversion as the norm and we treat introverts or introversion almost as something that should be just overcome. And I think that does us a disservice.

So, coming back to pair programming, if you want to torture an introvert, make them have conversations all day long. Make them sit and feel very vulnerable. Deplete their energy whilst they're constantly interacting with other people. Worse than that, stick them in an intellectual pursuit like software development where they want to be creative and completely inhibit that creativity by making sure that all the energy is put into conversation and relationships with other people. This is where I talk about the cult of extroversion. I think is unfortunate that we don't realize that people are extremely unique and introversion, extroversion is one huge continuum. And even that, it can be broken down to many different facets. But if we just take that one, I think it's very unfair to say that you need to be more extrovert, you need to be engaging with people all the time. There are great examples in the world where introverts have done incredibly amazing things by being their introverted selves. Ok. We brought into it, we brought into this idea that we have to collectively brainstorm and group think and we all must work together and throw our ideas all together. But in fact, an awful lot of progress and great things have come from the lone thinker. And like most things in this industry, it's not as simple as we all do it together. It's going to be a lot of variation.

So, back to pair programming which is only one of the things I point out. Pair programming for me, I don't have a problem with pair programming. I do pair programming, but as an introvert, as someone who recognizes the fact they are introverted, I really can't do it all day. It will kill me. I will go home a quivering mess. It's not pretty. I pretty drown my sorrows. I blame the fact that I would like to have a beer later on introversion and pair programming, beyond fair, maybe. Ok. So, what's the problem here? It's

the mandatory pair programming. I don't know how many organizations are like that here in Poland but certainly in London there's a few organizations that have started to dictate, in a generic sense or general sense, thou shall pair program or thou shall not touch the code. I think that's a little bit dangerous not Lis because I'm not sure Lis the greatest code in the world. But also to be honest, if you're an introvert, that's pain, that's basically, well, it's discrimination. It's going to stop me wanting to contribute as much as I would like to.

So, first thing be careful with pair programming. The other one, actually the other two, two particular problems I'd like to highlight. Ok. This is mainly for people who work in software development teams daily and you may or may not have seen these effects. They seem to happen where people get very obsessed with their craft. I call it the "Coacher of keyboard clacking". Ok? So, if you go " What's the sound of software development happening? It is –typing sound- or is it silence? Or is it conversation? Now, my experience is a combination of all three but I would say the least productive is the first one (typing sound). This has probably to be the biggest keyboard you've ever seen. But, yes, typing away. That's not where software happens. And if you are in a position of management don't assume that because people are using keyboards at the speed of light that they are thinking. They probably hunt. Or if they are, that's brilliant, but it's pretty amazing. I have actually been for interviews where the basic rule is "How fast are you with the IDE?" I'm not. I'm terribly slow. I can't even touch type. But I don't care because software development doesn't happen there. Software development, for me, happens somewhere else. And unfortunately for every organization I've ever worked with software development, for me, happens in the shower. So, it's not an easy explanation that one. I just like to go take a quick shower to have a think. Never an easy explanation to most management types which is why I work for myself. But, yeah, software development happens when you go out for a walk. It's when you're thinking, it's all about thinking.

Design is another phrase for thinking about it. And thinking about it doesn't limit itself to one place. It certainly doesn't limit itself to the keyboard. So, the culture of keyboard clacking. If you are in a position of management, please,I would employ to please stop worrying about "are they hitting the keyboard quickly enough?" By the way, combine that with pair programming. I don't know if you've had this experience. You sit down to pair programming and someone is such a genius at keyboard clacking that they're moving to the next phase with is keyboard ballet. It is literally a thing of beauty but you have no idea what they've done. You'll see there's the navigator pilot scenario and you're supposed to be commencing on what they've just done but all you've seen is – typing very fast-. Done. I have absolutely no idea what files you've just changed. I have no idea what you've typed. And now it doesn't compile. So, keyboard ballet. Let's no optimize for that either, please. I would much rather we optimized for thinking in this industry. It would do us a lot better.

Ok, the last two I'd like to leave you with, is when it comes back to introverts on practices anyway, is to think carefully about teams. Teams in this industry have become something of a cult and there's good reasons for it. Theirs is plenty of evidence to suggest that working in teams is extremely productive and that we can come up with often better than the sum of the parts solutions. But remember, you've got roughly 50% of introverts in the room. And there's plenty of evidence to suggest there that introverts are more creative when allowed to work on something alone. So, let them walk away and have a think. And this is something that we don't do very often, it's almost become frowned upon in many organizations I worked with is he's not a team player or she's not a team player, she is going to think about.. They keep coming back with great ideas but don't they include us in that gestation period? If they're introverts, they can't. Speaking of this one, I can't do that. I come up with my best ideas when no one is talking to me. And if I try and brainstorm these ideas using a mind map or something on the wall I don't come up with great ideas, I just come up with verbage. Bleah! And that's not what you're looking for usually.

So, again I'm coming back to that introversion quality that many of us have. Just let people be individual if they need to be. The culture of teams can be taken away too far in this regard, in my experience.

Ok. So the other sorts of things I can bring out, I think. Retrospectives. Do you have any idea what a retrospective does to an introvert? Think of a rack where they're being positioned in a situation where they have got to open up completely and tell you what they liked and disliked. And worse they can't even think about it because they don't think when everyone's here. It's like an enforced party situation where you can't pair up and go and have a conversation with somebody. It's very difficult. So, we have techniques for retrospectives where we try to help people go off and think and have their moments alone. Those are really, really important if you want to get the most out of 50% of people there. Please do, I've been in retrospectives where I've watched. As an introvert you almost get a radar for these things. You can watch the introverts flinch when things are going in a big group huddle way. It feels very uncomfortable. By the way, it's a bit odd that I'm speaking to 200 people and I'm an introvert. Actually it's really simple. As far as I'm concerned, I'm talking to one person, whoever I'm looking at that moment. Ok? This is the technique of an introvert in public speaking. I'm only talking to who I'm looking at right now. Hi! And that works for me. I cannot possibly talk to 200 people but I can talk to one person by looking at them.

Ok. Some other things to talk about in the practices area. I'll move on quickly in a sec so I want to go to the next section. One thing to look at very quickly is this idea of craftsmanship and mastery. I'd like to put out that there's a little bit of a concern I have in calling anyone a master of anything. That implies that they're almost done or that they are always right. Speaking about someone who's got a lot of influence in some parts of the industry being open source developer and all that sort of thing. I'm very weary of my influence and power in those regards. And so, I can, I am very careful about what I say because if you are the master you get labeled as the truth. And the problem I've got with that is there are always better ideas out there, somewhere and they're not coming just from mastery. I have a bit of an issue with the craftsmanship and mastery side of things, too. I'd much prefer the Zen approach. This is an Agile conference, after all. I can talk about Zen, right? And no one is going to hit me. Ok. If you are for the Zen approach, the master doesn't know what the answer is, he's going to help you discover the answer for yourself. I actually kind of like that mode of operation. Even when I do software development consultancy or writing I work with people to figure out how they get to the answer rather than trying to give them my answer because they'd never get the right lesson from that. Ok. So, that's enough on practices.

Now, let's look at architecture and design. Still with me? Ok. Architecture and design in software. How can I possibly make this interesting? I'm not sure I can, but I'll try. Ok. So first of all, let's start with what architecture is. And that's something that has perplexed people in this industry for years and years and years. And we've come up with terms like "it's the big decisions", big being one of those wonderfully objective terms that we all agree on, right? What's big and what's small. A big decision or not it's the language you use. A big decision is the framework you use, a big decision? A big decision is that line of code you're in, a big decision. Well, in truth they can all look big depending on how much impact they have. I think that, so we can actually rebadge architecture for what it really is, which is philosophy. When you are approaching the architecture of a solution you bring all of your ideas, all of your baggage, all of your experiences to that problem. Whether you are architecting a process change or you're architecting some software, you'll bring in your thought processes to the problem. So the interesting thing about this is that all those processes are build upon some axioms at the bottom. You just think it's self evident. And sometimes we haven't really thought about those. So, the interesting thing in software architecture at the moment is, I think, we're still stuck in the Waterfall mode of thinking. We still believe with the right amount of time, the right amount of consideration and the right genius moment, we will come up with the perfect, perfect, ideal architecture. And every time that doesn't happen we assume we did something wrong. I think that's flawed. I think it's inhumane and it's not doing ourselves any justice.

The reason for it is, and this is going to get a bit wacky, I blame Plato. If you're going to blame someone, blame someone from 2000 years ago. It's a good idea, right? Because they are a long way and they really cannot argue with you. If someone appears here they can argue with me and I'm sure they will hate it. Plato, lets' blame Plato. Bad Plato. So, Plato is an issue because he got us sold on this idea that thing could

be perfect. There was the perfect form, this whole process of excellence to get there. And it was so worth it and it was the ultimate good. And in fact to be fait I think Aristotle had the same issues and most of them actually at that particular academy, school of thinking. They really, really believed there was a perfection out there. No one more than Plato. To be fair with Plato, at the end of his life he did have one of those moments that you dread as someone who's come up with a school of thinking and going "Ah, I'm not sure I'm right." And the problem was I don't think he was right, but, unfortunately he has completely permeated Western thinking right into our current software development. Now, we've just spent 14 years killing Plato in our processes. We've been realizing that perfection doesn't exist. We've been realizing that we need to change and adapt as we go because fundamentally we don't know what we are doing. It's research and development every step of the way. And if that's the characteristics of processes, Agile being built on these pillars, I believe, then we have to turn our lens back on our software architecture and go "What the heck is that thing doing?" We've, unfortunately when you look at software architecture is still thinking that you can get it right the first time. Still believes that we are getting the right architecture and big decisions right the first time. And everyone is petrified about those big decisions because we know deep down it's a lie. We're lying to ourselves. We do this a lot in this industry. We're lying to ourselves. We believe in the platonic perfection of architecture and design and it hurts us. So, what's the alternative?

Well, it turns out that if you look at things a little differently there is a perfectly good philosophical school we could have gone to. The stoics would have told us that we are insane. Ok? The best example of stoicism as I can quote it is Seneca's statement that, and Seneca was a kind of a militant rather successful Roman stoic. He basically said "You do not know. The next day's not promised to you." Ok? No, hang on. "The next hour isn't promised to you." Fairly negative view on life on the surface. But in fact it's real. Every moment of your life you are not promised the next one. Now, we know that we ignore it, we don't like to consider it too much. Seneca would advise us to think about it a bit more. I think in software and in our architecture and design we should be really thinking about these things. We don't know what's going to happen next. We need to build our architecture and design that helps us change because we don't know what's going to happen. We need to stop building architecture that is fundamentally in conflict with reality. And instead build architecture that accepts that we don't know. And there's a lot we don't know we don't know. And it gets worse. Ok. So, let's all become stoics in our software architecture.

This is why I call it the unreality of architecture and design because we spend most of our time ignoring the real consequences and the real possibilities with our software because we really don't know what's going to happen. I think I have a little bit of time to tell you a story about how this feels. So, who's in a software development team? Ok. Cool. Who does daily stand-ups or something similar? Ok. Sweet. And who does iterations of, they vary in length? I imagine all people do some kind of iteration. Imagine this and I'm going to ask you to trust me. Do you trust me? You've just met me. Trust me? Ok, shut your eyes. Close your eyes. Just relax. Let the coffee work its way through your system. Ok. It's sprint reiteration 10, to pick an arbitrary number. You just entered into your stand-up meeting and like just any sane human being you sat down. And you sit there and you're feeling good, really good. You have, you are acing it. This is sprint 10 and that software is beautiful. The customer is happy, you're happy, life is good, you can hear the birds in the trees. And they're singing your tune. And then, and then fortune or Fortuna comes to bite you. What happens is, whoever the decision maker is, producter or whatever label you would like to give to that person, they come into the room and say the one phrase you have been dreading "I'd just like one small change." At that moment, I'm going to take you on a little emotional journey now, at that moment the first reaction is anger. "How dare you? How dare, I'll cancel the sprint. I'll throw all my toys out the proverbial pram. Right now I hate you. How dare you tell me? We agreed, we agreed at the beginning of this sprint what we were going to do. How dare you change your mind?" That is swiftly followed by the next emotion which is guilt because deep down, really deep down you are supposed to be an Agile software developer. What does Agile mean? We embrace change. And so you know that you would like to accommodate this change and it's worse than this, right? It's worse because what's happened I call the

elephant in the stand-up. You've hit the situation where your architecture and design and all the code that you wrote yesterday which we colloquially call legacy is now in conflict with reality. And it's not your fault because you're not asked to think about how architecture needs to change. And that's something we need to start doing. Ok, you can open your eyes now if you still have a shot.

Ok. The very last thing I'm going to talk about that is what happens next without a doubt. What happens next is we lie. Ok. We use a couple of terms. The first one is "Yeah, we need to, we need to do some refactoring. You've heard of refactoring, right? It's a real term. We need to do that for about a month. Are we ok, are we good? No, no, you're going to get no new features. No, no, no. We need to do refactoring. It's a real thing. You've heard of it. We've got to do it. Stop getting g in the way of the Agile process, man. We're doing refactoring now." Now, I've looked at refactoring and by definition is supposed to be one small change that doesn't change the behavior of your system. If you need to do your month's work of refactoring, you're not refactoring.

Ok. Let's talk about what else you might use as an excuse. Let's give you that, too. The next one is a beauty. Technical debt where we've achieved. You've heard of the term technical debt? We've done that. We are moving way to quick and it's your fault. We build this stuff and now we have to redress the balance. We're going to pay some back. Yes, a similar message and you're not going to get anything for a month or so. And we've got to really prepay the system back. Once again we are cloaking the real problem which is that our software can't keep up with the change that's come in. I'm not saying that every change should be small and there are times when the product owner or someone like that will say "I want something small" and it's not. But what you're trying to avoid arte those situations where you have to lie, the urge to lie. When actually it's not, it should really be a small change but it's not. Ok. That's the inhumanity that's caused by architecture and design.

Last of all. The very much last of all. The inhumanity of doctrine. Now I realize I'm on dodgy ground now because I'm going to be starting to sacrifice some of the very, very things we hold dear. Not in a very bad way I hope. I have an issue with doctrine, I have an issue with manifestos. And we've got one and it's a good one. But there's a problem with manifestos very, very deep down in what they are. Ok. So, the problem I have with manifestos is that they ossify the opinions around a particular area. They immediately crystallize and make it hard to change something. So, my argument in this regard is that I believe that the Agile manifesto, the good thing that's done, are wonderful. What we haven't noticed is the bad thing that's done because to a certain degree. That's it, I'm done? I've got to go? Sorry. The manifesto, the challenge with it is that it ossifies. If you ever go to Ireland, ossification means something completely different. I realized that terminology wise, ossification means drunk which actually still works. So, I think manifestos are ossifying so they either get you drunk or they stop you from ever challenging these things. So, that's my argument really. I believe the danger of manifestos, and one of the reasons I really try not to write any or contribute to any or sign any is because I feel they immediately ossify and potentially can strain what can be done and invented upon. So, just something to put out there is that we really have won the better ones in this industry but I do think that we very carefully need to attack and move on from many of its thinking because there's lots of innovation on top that we can apply.

Ok. So, my remember and apply section. I'm not sure how much time I have left but I'm quickly going to get through this and then I'm out. I've got zero. That's a pretty solid zero. Ok. Very quickly. Introverts, watch out for them! They are out there. There's lots of them. I am one. If you want to talk to me about what makes an introvert come and grab me, I'll be in one of those booths hiding in a corner. That's what we do.

Understand and balance across your teams. Understand who you've got and what they do and how they operate. Everyone is so unique and different. Please, be human about it and look for those sorts of character traits. It will get a lot more out of your teams.

Embrace an individual's journey, don't just shoehorn everyone into teams if they, if you feel that's just the

way it should be done. Watch out for the introverts who would find that incredibly painful. And you're just not getting their creativity because that's not where it comes from. Most of all, be kind to people. And let's make our processes and practices and architecture and design consider humans and change the reality a bit more.

Ok. Being scientific and open versus doctrine. Ok. Team building is not one, you should get to know your people. And thank you.

Paweł Brodziński: Culture: The forgotten bit

PawelHello everyone! So, talking about culture I'm going to use this word like 100 times during the next 30 minutes. So you know it's important after this session. And it's not going to be kind of super, super, super practical because when we are touching culture, it's very contextual but I want to tell you why it's important and how to tackle the culture in your own, in your own contexts.

Let me start with, let me start with that. John Kotter in 1999 basing on his research said "About 70% of all major change programs fail." And by that time it was no new news. It was the same for quite some time. 20 years later we are here, 20 years later and if you look at research around success rates of change programs, it hasn't moved a bit. It's the same for more than 40 years already. It's, we are failing 70 to 90% of the time. There is one interesting perspective to those statistics. For the past 15 years we have seen a huge rise of Agile and Lean, approaches that directly aim to improve the way we work, to change the way we work and they don't seem to have impact on those numbers at all. Now, there is an obvious question "Why?" and my answer to that question would be that we focus too much on practices, tools, techniques, methods. We are looking for recipes. This is my experience from ACE! conferences over the years we would frequently talk about "Ok. Give me recipe how I should change my team." Well, it doesn't work. What we should be paying more attention to is what's beyond practices. What is the mindset of people who are adopting a change? What are the principles they follow? What are the values that they share? And when I'm talking about mindsets, value, these principles, I cannot help but bring organizational culture to discussion.

So, we define organizational culture as a sum of behaviors of people within an organization. And not only behavior but also what drives those behaviors. So, we would have visions, values, beliefs, norms, systems, so anything that forms those behaviors, those every day, those every day behaviors. When we are talking about culture versus practices, there's a very interesting experiment that I've run recently. During Kanban Leadership Retreat I run a workshop, by the way Kanban Leadership Retreat is an event where once a year leaders of Global Lean Kanban Community would go to discuss the state of the tools we have, to try to improve those. So, they run their workshop around failed changes and why we failed with those changed programs. And this is the end result of that workshop. So, whatever you see on the picture and it's more to the left, it's more related to culture. If it's more to the right, it's more related to practice. Now, this picture has two parts. The top part are root causes of failures. So, we start we fail, failure stories and then we try to find why we failed. And then, we distributed those sticky notes on the scale. So, any sticky note that is more to the left, it's like, it tells a story like "we lock the proper culture". And if it's more to the right, it says the story like "we lock the proper practices". Now, the bottom part is about solutions. So, once knowing what failed, we try to find solutions for those issues. And again, more to the left something is, it's more about addressing the cultural problems, more to the right something is, it's addressing more the problem of practices. Now, I see a problem here. This picture is heavy on those two areas which basically tells us that we understand, that the majority of issues that make our change programs fail are related to culture. At the same time, majority of tools and solutions that we have it's all about practices.

I see slight balance here. And then we tell that it's not the result of random Agile cultures, these are people who are pushing the envelope of Global Lean Kanban Community, community that has evolutionary change painted on its banners in a big fonts. So, there is a huge problem here.

So, when we are talking about culture, probably, sooner or later, this thing will pop up which is culture hacking. And I consider that a dangerous thing because the way it is most frequently understood is like" Let's change something here and see what happens." The way we hack some code. Now, the difference is you're not playing with code here. You're playing with humans, with their beliefs, with their behaviors

and there's no way back. If you change people's behavior usually there's no going back to the old way of how they were behaving. So it's very risky.

I would much prefer a more structured approach which is PDCA cycle, for example. PDCA comes from plan, do, change, act. So, the plan part is defining a hypothesis. If I change something, I expect some kind of result. Now, it's not only defining a hypothesis, it's also defining how we are going to measure whether the hypothesis is right or wrong. So how we are going, when and how we are going to tell whether the experiment worked or didn't. Only then we start to implement the actual experiment. And then we check its results and then we act on those results designing the next experiment because it goes, it goes one after the other. The important part is not really learning that's the experiment or the hypothesis was proved or disproved. The important part is understanding, is understanding what's changing in our organization, how we can act for our organization, how we can influence that change. And by the way, the same thing is true for practices. So this is another outcome of Kanban Leadership which is basically just stealing some of the ideas that were connected which basically says that if we understand the practices we use, we will learn. If we apply them in kind of mindless way we risk that we will end up in doing some sort of cargo cult.

So, let me use an example to explain that. It's morning, you need some physical exercises. So, please stand up, everyone who does stand-up, some sort of stand-ups in your team. I would assume that most of you will stand up. Yes, stand-ups, stand-ups are the most prevalent Agile practice. Now, keep standing, don't sit down. Keep standing if you ask yourself and your teams explicitly, explicitly why you are doing stand-ups. So what kind of value you are getting from stand-ups. Keep standing if you have this discussion with your team explicitly. It's nothing bad. Ok. The rest of you keep standing if you ask yourself and your teams explicitly, again explicitly, what value you're supposed to get from stand-ups. Why, stand-ups were initially a part of Scrum, and other Agile methods, you've had this discussion with your teams? Keep standing if you did. If not, please sit down. Thank you. That's interest..Thank you.

Now, that's interesting. It seems that majority of you doing those practices don't really pay attention to understand how they are working and why they are working. In other words you are risking you are end up doing cargo cult. Most famous example of cargo cult is from Pacific islands. So, during the Second World War American would come, they built military bases. They'd bring airplanes, ships, they'd bring all sorts of cargo like food, weapons, clothes, some of them they shared with the natives. Now, the war is over. Americans were gone, but the natives still wanted all those supplies. So, they figured out that if they copy all the practices the American did, you know, cargo would come. So they build airstrips, air control towers, airplanes, everything out of wood, by the way. And they were hoping, they were hoping, you know that cargo would come. Guess how did it work for them? It didn't. And if you don't try to understand the practices you use and just use one example, you're risking the same fate.

Now, let's go back to organizational culture for a bit. If you want to influence how people use different practices you need to work with your beliefs which is basically part of the organizational culture. If you want to change organization culture, what could you do? Organization culture is a sum of behaviors which means that you cannot just mandate a change of organization culture. You can't say "from today we are a learning organization." How does it work? It doesn't. There is another bad news actually because we cannot mandate the change of behaviors, either. I mean you can introduce a policeman, you can introduce an Agile coach who just keep, do, keep, keep taking care that you are doing the right stuff. And then policeman is gone and what do you do? You go back to the old way of doing things.

So, how can you influence the culture of change? Well, you can work on the constraints. What kind of constraints? Well, what is allowed? What is not allowed? What is encouraged? What is discouraged? What is rewarded? What is punished? Where are the structures? Where are the hierarchies? And , by the way, when you're answering those questions where are the rules? And then, what is acceptable? How far I can go breaking the rules before I get burned? Because all your rules are carved in stone, changing influencing

the organizational change would be very, very difficult. So, what kind of constraints we can work on? We have values, visions, symbols, for example. Let's start with those. So, we have values. Every organization, every company has some values. They claim to be, to care about all sorts of good stuff . And those values will frequently end up in stuff in mission statement or visions.

So, another exercise, another experiment. Please, stand up, everyone who works for a company that has one of those. Most companies do. I know at least one that doesn't. The one I work for. Cool. Now keep standing, don't sit down if you can quote the mission statement from the top of your head. 1,2 ,3 that's all. 4. 5. Thank you. 6. You're awesome.

I see a huge problem here. You don't seem to give a damn about the values of your companies. You cannot even quote all those nice things or those values are written down. But you know what? it doesn't really matter. We'll be back on that in a minute. Now, let's look at some values. This is nice cloud of values that we can use. There are some of those that would be quite common across different the organizations. So, like customer satisfaction, high quality, people. Like let's pick those 3. Which company wouldn't claim that they care about customer satisfaction, high quality and people? Well, let's check. Please stand up everyone that works for a company that claims to care about customer satisfaction, high quality and people. If someone is not standing, well, it may be weird. But what would I know? Ok. Now keep standing, keep standing. Don't sit down if the leaders and mangers in your organization consistently act every day in a way that shows that they do care about customer satisfaction, high quality and people, consistently act every day. Now, be brave. Cameras are directed here. Except of that one. Ok. Thank you.

Now, not only you don't give a damn about the values written down in your mission statements, but a big part of you actually disagrees with claimed values presented by your company. I see a huge problem with authenticity here. Those values that your companies claim to have, no, these are not values. Remember that? Organizational culture, sum of behaviors, not sum of words in a mission statement. Different thing. So, organizational culture values that are valued by your company would be presented, represented by the behaviors everyday behaviors of the people in that company. Now, you may ask yourself the question "what kind of values are represented by those behaviors?" and I will leave you with that thought.

Now, another constraint that we can work on and system thinkers would probably disagree with me, are people, I mean organizational culture, sum of behaviors. Behaviors of whom? Of people. Now, when we are talking about people, we hire people. We build our teams. So, when I'm talking about the the culture and people and hiring I cannot but mention hiring for cultural fit. Now, this comes as frequently misunderstood, this comes as frequently misunderstood as "let's hire people who we get along with." "Let's hire people who we feel good when we talk to them." That's just a bad idea because we get on well with people who are similar to us. If we hire such people we will end up with a very uniform culture which means our teams won't be very creative, won't be very effective in terms of dealing with complex solutions. That's not exactly something we want to achieve. The way we tackle, we approach hiring for cultural fit into our logic is we look for some people within some canon, some general cultural canon for a company. But at the same time they are as different from anyone else as possible. So, let me give you an example.

If you ask me " Would it very likely that we hire an Indian? Unlikely. I think, I think that in the majority of cases it would be too big cultural difference and having such a person would more a disruption for a culture than augmentation of a culture, at the same time whenever we are talking with candidate we are looking for different characters, different passions, different interests because this is what adds to our culture. At the same time you want them not to disrupt the way we work already. Because we want to avoid group think.

Group think is psychological phenomenon that basically makes teams or groups of people work to achieve very fast or very quick consensus or to act for conformity. So, if you have that, you get penalty on creativity,

you get penalty on effectiveness. So, we don't want to, you, we don't want to do that. We want to avoid that.

Another problem with hiring is that we tend to hire super freaking heroes. I mean we look for individuals who are extremely skilled, extremely intelligent, this is a perfect candidate. Now, the problem is that those guys don't work alone even if they are introverts. They end up working as a part of a team. Now, we should be paying more attention to how they would help a team become more effective. There's research around that. The research that says that there is this thing called collective intelligence that it is much more predictive in terms of how team would succeed if accomplishing complex tasks than individual intelligence would be. And we don't. I'm happy to discuss that late during the event, what makes teams collectively intelligent it's not individual, it's not, these are not the traits that we usually pay attention to. And by the way, this research shows that the more women we have in our team, the better the collective intelligence. And it's not about having a woman in a team, it's about having as many of them as possible. Again, I'm happy to propose an open place session on that if you want to discuss that in extent.

Now another constraint that we can work on is leadership. And I'm not taking about leadership as, meaning position of a leader because for that we have another word which is a manger. I'm talking about leadership as the act of leading people. In fact not only the act of leading people but also the opportunity to lead. I mean, if I'm a line worker, a developer, a tester, whoever, do I have opportunities to lead teams, other people? Now, in most companies leadership would be organized in this kind of way. The leadership would be organized in a very structured, very hierarchical manner. Leadership opportunities would be assigned to positions of mangers. So, they would have power to lead.

Now, let me run one more experiment with you. And you have been standing up and sitting down for quite a while so this will be a thought experiment. So, no physical exercise required. Ask yourself the question "What would happen if in your company now all the hierarchies, all the structures would disappear? Everyone will be peer to everyone else." Now, obviously your company wouldn't extinct over night. So, something would happen. Would there be hierarchies? I would say yes. Even if there are no formal structures, some hierarchy, some structures would emerge. Now, what kind of structures would emerge depends on what kind of culture you have in your organization. So if the constraints that form that organizational culture are reasonable, if values and principles are healthy, those structures should be healthy as well. If they are dysfunctional, you may recreate whatever you have right now. So, in other words, the answer to that question that I asked you may give you some kind of insight, what kind of values, what kind of principles your company follows.

Now, I asked you to do that thought experiment which basically was floating with no management. No matters on this idea that a company doesn't need managers at all to be run. And there are examples of such companies. In IT industry we do the Valve, the guys that do games or Eachshop, you probably know this one. From other industries, Samco, Morningstar, W.L.Gore, guys that created Goretex. You definitely know what Goretex is. They don't have mangers at all. It doesn't mean that they don't have hierarchies. It means that their hierarchies are emergent. In other words they have to care about the culture, otherwise those organizations will quickly become dysfunctional. Now, this is important because the more you are towards a no management out of scale and this is a scale, by the way. It's not a binary choice. We either do have mangers or we don't. It's a scale between very rigid, very hierarchical structures and no management. You can be anywhere and the closer you are to no management out of scale, the better the distribution of leadership is, the more opportunities for everyone to take over the leadership and do that. And you want to be closer because the closer you are to that scale the more empowered people are. I hate that word, it's so bloated. It's like I empower you to bring me a beer, thank you.

But serves as a proxi here for a, for how much of a change, people have to take over the leadership, how much power they to change the organization that is around them. Whether they can do the right thing when they know what the right thing is. Now, obviously the culture, especially when we are talking about

those cultures which are closer to no management is difficult to scale. I mean it's easy to have no mangers when there's three of us. It's not as easy when there are thirty of us. It gets pretty damn hard when there are 300 of us. And I don't even know how it works for 3000 of us. Now, at the same time, I consider growth as something that is so common but shouldn't be for pretty much any organization out there, growth scaling up is one of their goals. Should it be? I mean growth is not only a risk for those emergent structures, not only for this no management out of scale, it's also risk for any culture. Whenever I hear a company say "I want to grow by 100% in a year", I'm like "Are you sure?" because in a year, if you succeed in a year, you'll have a very, very different company. A very different culture because if you hire like crazy, which means 100% in a year is hiring like crazy, basically controlling how the culture evolves and where it evolves it's almost impossible. If you grow slow, you can, you can get people to know the culture, to learn the culture of your company. So, there is not that much risk of changing in the wrong direction. If you hire people like crazy, well, sorry.

For those of you who are working for big organizations and I know there is a bunch of you here, there is a hope as well. Everything that I said about organizational culture, I was addressing to the culture of the whole company. There is this idea of culture pockets. Culture that can exist within a bigger culture, that may be very different. And these kind of things happen either when a part of company works in isolation, I mean, a branch in a different country, a branch in a different city or a team in a different office, but these things happen also when there is a strong leader that can create for their team they can create a bit different environment. And it may be, this culture pocket can be better, whatever better means in this context. Maybe better than the culture of the whole organization or worse. So, think and ask a boss. He can create, he can create hell for the whole team even though the rest of the organization can be, can be pretty healthy. So, the same is true, all I said is true even in a smaller context. Except culture pockets are not as sustainable as the culture of the whole organization because it's enough that few people are gone and culture pockets disappears. It's enough that we have a real organization and the culture pocket disappears.

Now, what I tried to do with this session. I tried to explain what is organizational culture, why it is important for every company, why understanding the existing culture may help you understand why so many of change initiatives fail. Why some of them are doomed to failure from day one because they are not aligned with what is existing culture of the company. Understanding organizational culture in your context will also help you to become a better leader.

And last but not least. Understanding your culture around you and understanding what you personally expect from a company may help you to realize that in your current organization you won't be able to pursue your goals. And it may be good time to move.

Thank you.

Martin Burns: Dive into A3 Thinking

Michael Hogan: How SAFe Differs from Scrum

Michael Hogan: So, booing and limited of less time here maybe, maybe before lunch and plans. I'm pretty sure I had a dream about this situation before. But the good news is that is catalyst for change and imprompts. We're going to do that. I want to walk through my impressions of Scrum and SAFe. These are my impressions and in order for that my impressions to be meaningful for you, I want to start off by first walking through a little bit of who I am and what shaped my perspective on these issues.

So, my foundation is in industrial and systems engineering and I did that at USC, and industrial engineering is the root of where a lot the productivity improvement methodology has come from whether it's Deming of a team in Toyota. And it focuses on statistics and on different processes and improvement techniques. And what's nice about industrial engineering is it gives the practitioner a foundation in the statistics and a mathematical bagging behind these methods that we use. And that creates a little bit of freedom, to kind of go outside the lines and stick closer to the manifesto of Agile or the manifesto of Lean rather than having to use the codified frameworks and implementations. Without question, ok?

And so, then I went into systems engineering for a software development organization and I was helping to run operations for a team of 100 people, we were inside an organization of 800 people, that was inside of an organization of thousands of people, that's inside of an organization of hundreds of thousands of people. So doing that I got to see some of the design challenges that come up, some of those communication and scaling challenges and some techniques that help to resolve those.

I liked helping to resolve those so much, I switched into an internal consultant role and started working with theory of constraints and Lean and scaling that out across a large engineering organization. And I was lucky enough in that process to get to meet Mike Cohn when we were getting trained for Scrum, getting to meet Dean Leffingwell giving training for SAFe and being able to talk about some issues with each framework with those gentlemen and what the impact that had on my thinking in a couple of slides.

And now I work primarily in a new product incubator, trying to help teams come up with ways to move from idea to getting a product commercialized much faster. So, my starting out point here is "Scrum alone is not a sufficient framework to be successful with development." Ok? And I like Scrum a lot. I mentioned I went through theory of constraints, and a lot of the literature there is very, it's narrative driven, it's story driven to goal books or novels and it's really hard to turn that into practice. And when I found Mike Cohn's work it really helped to make theory of constraints actionable through the Scrum model of managing work. But at the same time, you know, when we switch to Scrum and we try to move away from big design upfront requirement documents, one of the things that makes that possible is the testing and some of the quality tools that document our code as artifacts of development process. And if those aren't in place for an enterprise, especially there's some risks if you go Scrum only. So, my position is that Scrum alone is not enough.

So, how do Scrum and SAFe compare? And as Paul said it's kind of complex. SAFe is a huge topic. Scrum is a huge topic. So, I want to focus on a few meaningful differences. They relate to the challenge of scaling something and getting it adopted by an organization. So, the first thing that we see is architecture. That's over there. So, in terms of architecture I had a chance to talk to Mike Cohn about it. How does Scrum handle architecture? How does that work? And where is it happened? And a lot of that conversation centered around that concept in a smaller team. There might not be much need of architecture. But when you're trying to coordinate the work of hundreds or thousands of employees on a given topic, it can

become a little bit more important. And what Mike brought up is that from his view, the Scrum approach was not necessarily designed for those types of activities. But, if you had to do it, it would probably be good to do it with an upfront 1 or 2 iterations that focus on architecture and design. Freeze that and them move into the development work managed with Scrum. So, I've highlighted that yellow here because it's a little bit of risk area where maybe Scrum doesn't handle architecture quite as well as it could.

In SAFe architecture and design are handled with architectural epics and with architectural runways. The architectural runway is the concept that in order for the development team to leverage the architecture, it has to be sufficient that they know how to build with it. And they use the example of landing an airplane. If you're going to land an airplane you have to have the runway in place first. And using a new architecture, the analogy is landing an airplane. The architectural epic is how those architectural runway changes are rolled out into the organization. This is in yellow, too, which is kind of my caution color in the presentation because it doesn't necessarily address where this happens.

So, next we have managing development. So, Scrum uses Scrum to manage development. So, that's pretty straightforward. SAFe also uses Scrum to manage development but they modified Scrum. And they modified Scrum with normalized story points and that probably should be in yellow here. It will be in yellow later and we'll come back and see why that's an area of caution.

When we look at managing the program level, Scrum uses the Scrum of Scrums. SAFe uses the Agile Release Train and they focus on features and weighted shortest job first. And there's examples of both techniques working.

When we talk about managing at the enterprise level, Scrum doesn't really have that language and SAFe introduces this concept of the epic Kanban. But it's still a little bit abstract as to how that would actually be implemented by a practitioner. But at least people are starting to talk about it which is useful.

When we look at code quality, Scrum by the books stops at the acceptance criteria on the user story. And the real challenge with that is that if we are really going to get rid of our design specs upfront and use user stories for the Agile development and we throw the user story away, which is what we do if we are by the book in Scrum, then we lose that documentation of our product unless we have some of the extreme programming quality tools like Automated testing, Auto generating documentation and these sorts of things. So, I like that SAFe brings those in and says "Yeap, we're going to have acceptance criteria but we're also going to make sure that we have our XP technical practices." And you're not going to be successful as an organization if you don't bring those in, too.

Politics and policy is also very important if you try to adopt something in a large organization. And if you're trying to get it adopted as a framework that you can use to sell to policy organizations like national governments, state governments, city governments SAFe addresses this partially. And I'm rating it here as partial because I think they handle the internal politics right now in the current framework within a corporate organization. They're still needing to work on getting acceptance in the customer organization and getting the acquisition process, policy and guidance in place and accept it so that companies can use SAFe with confidence that their customer will buy with their developing if they're working in an RFP context which is common if you're developing for the government.

And finally there's the financial model component. And the financial model is important because this is how the engineering and design department is able to communicate in a meaningful way with the CFO of a finance organization. And, you know, if people have read "The Phoenix Project" which was not my favorite book, but it makes a nice case that you need to be able to interface between finance and design and ops if you're going to be successful and if you're going to be able to rely on what your designers are doing around the needs of the company. So, SAFe brings in this concept of cost of delay heavily borrowing from Don Reinertsen's work. And this is the part that got me excited about what SAFe was trying to add to the conversation about Agile.

So, I want to explain this briefly to people who may not have seen it. So, what is cost of delay? So, cost of delay is the cost of coming late to market with the new product or service that you try to deliver. And we see cost of delay through three primary signs, I guess. One is theft. So, if you are late to market, your idea can be stolen. And where we see this is any time Apple is coming out with a product and we start finding out about what cases the vendors are designing, and what shape the next iPhone or iPad is going to be before it gets released. That forces Apple to deliver quickly otherwise you can start having copycat devices pop up before you ship.

Obsolescence is if you're a Blu-ray manufacturer and you really want to get this great Blu-ray player out but all your customers have moved on because you took too long and now they're watching their movies on Netflix.

And lost sales is that delta between, you know, the market audience that you could have had if you'd released on time and the market you end up with when you release late or with a delay.

And Reinertsen talks about the human flow diagram as a tool for visualizing this and there's a great blog post with the URL's down here. It's by Mike Griffiths called "Creating and Interpreting Cumulative Flow Diagram". For some reason the Pdf comes up higher in this search result in this blog post. And I'm not going to go into it here right now, but it's a great reference to look at if you're looking for a tool you can use. It's very simple, low data to get it going and will help you to see where the risks are of delays in your process and try to identify where to focus if you want to get rid of those.

So, we also talked about how SAFe added architecture concepts, code quality concepts, policy extension concepts that go a little beyond what Scrum deals with. So how do those help? First you want to look at does it mean to scale something and again where the context is thousands of employees up to hundreds of thousands of employees.

Geographic scale. So, multiple continents, multiple cities. Public policy integration. Getting the government to buy or even local policy makers. And it's not necessarily because the government is buying it, it might be, you know, a situation like Tesla's facing in the United States, where car dealerships are posed to Tesla selling direct. There's a policy issue that has to be resolved for them to be able to sell direct.

Task variety and product variety as the company grows and takes on more initiatives. And supplier integration. How do you include the suppliers in this process to make sure that your vendors who you're subcontracting work to are able to participate? All these issue come into play as we're scaling and some of them might be more significant in hardware or mixed hardware software projects.

So, as we attempt to scale there are a number of barriers that are common. I don't think I've captured all here but I want to highlight a few and I want to look at how SAFe attempts to address some of these barriers. So, sponsorship is one. Having a sponsor for the initiative is like number 1 on almost every change methodology that I've ever encountered and yet so many times we go into that change without having that sponsor and we fail. And then we wonder why didn't that happen? So, SAFe attempts to highlight that by bringing in the sponsors and making them part of the picture. And that both says you need to have a sponsor and it also makes the sponsor feel included as a stake holder in the change that you're trying to bring to the organization.

Confidence. People need to be confident in what you're trying to do and so we see is case studies coming out because even though as, from a statistics perspective whether someone's selling Taco Bell burritos or they're developing software or they're developing lawnmowers or something, all those design cues kind of look the same. But for those people there's a lot of uniqueness involved in that and they don't look the same and they're going to ask you for a case "Is there anyone else out there in my industry, in my unique situation who's been successful with what you recommend that I try to do?" And so again, the reason people seek these cases is this issue of uniqueness and one of the interesting things about SAFe is that it's not necessarily super prominent in the framework but there's language about having hooks in the

framework for tailoring to tailor to the organization. You don't have to use SAFe exactly as specified. It's, it exists as a template to help accelerating but even Dean Leffingwell on his seminars for coaches would say "There's hooks in SAFe where you can tailor to meet the needs of the organization." And that helps people to adopt the change.

Language. The communality of a language is really important when we try to scale across dozens or hundreds of teams. And having something like the big picture to communicate that can help because people need to know what each other are talking about for something to grow.

Education. Having experienced both certification programs as a coach, I think SAFe certification framework is more favorable to coaches. And that might be why it's kind of picking up some steam. And there are some risks with that because maybe the barriers to getting certified that can be a little bit lower in some cases. But it's important that certification program exists, the learning path is clear and it's kind of integrated and that's nice.

Policy. So, when I got trained for SAFe, the Software Engineering Institute was at the same training. They had a representative there. The person was there because they were interested in "Can they recommend to the government accounting office or the US DoD type organization or the US, you know, the food and drug or the Heath Organization to adopt Scaled Agile Framework as an acquisition policy approach?" And the SAFe leadership and thought leaders are really putting in a lot of effort to make those discussions happen and try to get that acquisition approach to come into being and that's important in terms of allowing large organizations to invest in rolling out a change effort. So, that's very important.

We talked about quality and XP tools and then tooling. So, tooling for me here is a risk area. We see the terms of SAFe being introduced into rally and into version 1, things at the portfolio level, things at the program level. And the challenge here is if everyone on the team doesn't have access to the tool, in my experience and in what I have observed, typically the team really isn't using the methodology because somebody else is statusing for them and putting data into this tool. So, I see this as a weakness right now.

So, a few aspects to be cautious about. The normalized story points bothers me. I think there is a lot of value in having that understanding of the team as a unique entity working together and what their independent velocity is. And I think that if you take the time to understand velocity and how it works, the issues that a story-points normalization seeks to address kind of drop away and not be as big of a barrier as they might seem. Weighted shortest job first I really liked as a concept. I'm not sure we know yet how to do that because it requires understanding the value of the epic, or the value of the feature. And often time for organizations coming from earned value management, value equates to what it costs to develop and that's not going to get you the right way to the shortest job first. So, there is some work that needs to be done in terms of "How do you do this?"

We talked about tools. I'm concerned about premature adoption as an acquisition framework. SAFe may or may not reach that stage. But, you know, if it did, as it is today, I don't think it gives us the jump we're looking for over the legacy acquisition framework which is the Waterfall method and kind of a milestone designer for these because the current implementation doesn't solve those issues that make projects fail in that context. I think they're moving towards it, though in future iterations.

And, then lastly, not as important as the others maybe, it bothers me that there is this emphasis always on face to face. We live in 2014. We have amazing collaboration tools available to us. And if you follow 37Signals now Basecamp and read the book "Rework", they talk about how they built a team with people all over the world. It allowed them to get the best people. And that skills differential of the people they brought into the organization outweighs maybe some of the communication difficulties thst they face and they're also able organize the work in an intentional way that leverages where people live and the times of day that overlap and don't overlap. So, I'm not a big fan of the insistence of face to face.

So, my conclusion on SAFe is that it's very useful today as a framework for enterprises to get started

with the transition to Agile but further iterations may continue to make it better. And I want to briefly explain what that might look like, one possibility. So, where we look for the end state? Where might SAFe be going? Where might other framework be going? And a lot of this comes from Toyota as project development. Definitely it shapes SAFe. It seems like it also shapes Agile thinking. And so, if we accept that we want to achieve the outcomes of flow, of value, we want the elimination of wasted resources and activities, we want respect for people in our team, then it's interesting to look at "Have we made an assumption here that Toyota does design the same way that they handle production?" So, we need to look at "How does Toyota get there?", "How does Toyota do design?"

So, what we see, is Toyota doesn't use point-based design. MIT has kind of looked into this and there's only a couple of papers. It's hard to find this stuff. It's kind of Toyota open-sourced their production process because that helps them interface with suppliers. But they didn't open-source their design process as competitive advantage and MIT's done a little bit of looking into it but still there's not a lot of literature out there. And what happens is in this point-based design situation if you miss your milestone because you focus on one type of engine for your car, let's say a hybrid engine, and you're developing the engine whether it's Agile or Waterfall or whatever approach, and you arrive at the production readiness decision "Are we going online with manufacturing this thing?" and that engine doesn't work and there's no backup plan because it's point-based design, what ends up happening is your schedule slips because you have to redo the design effort "Hey, let's go back to the internal combustion engine. Let's start from scratch." So, what MIT has found that Toyota does is a set-based design. They keep, and I fabricate the engine example because it's convenient to talk about, but what Toyota does is they keep multiple technology alternatives in play throughout the development of a new vehicle. So, if they want the electric engine, they'll have a hybrid engine as maybe a middle step and they've got the internal combustion engine that they know how to build and they can fall back on and they mature all three in parallel. And what that does, is it gives a set of potential technologies. Those technologies are modular. They can be switched out for one another and that means when they get to that decision point that says "We need to pick an engine today and start the factory or we're going to miss our target shipment day and we're going to have loss sales." they are able to move forward with the technology that is closest to what they want but it works today. So, maybe they don't get to the electric engine, but they've got the hybrid working. They can leave the gasoline engine behind and introduce the hybrid vehicle.

So, my challenge to everyone as an Agile community is as a group how do we challenge the assumption that design is the same as production? And if we acknowledge some of the uniqueness around design, how we then incorporate set-based design, the Lean design that we observed at Toyota? How do we incorporate that? So, it's a feature for the Lean production methods that we already know how to do and in software that applies to devops or maybe maintenance, maybe some other tasks. How do we also bring in the Lean set-based design whether that's in SAFe or any other framework that will let us, that will boost us ahead, that will get us to the next level of effective development.

So, thank you.

Olaf Lewitz: Reduce politics and improve culture

Olaf Lewitz: I hope you are well fed and watered. I want to welcome you with my whole heart. When I entered school I was very open-hearted and open minded and got told very quickly that that's not helpful for being successful.

About 15 years later when I entered the work place, I tried again to be very open-hearted and open-minded and again that didn't work out so well. Past 20 years in the work place I managed to regain being open-minded, open-hearted and actually talked about love in the work place without being embarrassed and I want to share with you how that came about. I want to learn on the way because one of the main things is that all have a very individual idea of what's normal and we all share a very big fear to actually show our normal to the other "normal" persons out there.

So what I thought, I got this message, not in exactly these words because I grew up in Germany as a child that you deserve t love what you do and that you are loved and experience that love for what you do. And society, school and work tried to beat that out of me. Thank God not very successfully over time and it let me to, as Paul already said, label myself as a trust artist because I want to help others unleash this whole heartedness because I think our society, our workplace and our organizations are lacking it and we're losing a lot of human potential and creating a lot of pain that way.

What this whole heartedness means in terms of an organization? I'm talking about workplaces where we love to belong. A friend of mine in Brazil, he has a company slogan "I'll make you love Mondays" and that's exactly what I mean, that we have a workplace that doesn't feel so different from the rest of our life, that doesn't feel so alien, that doesn't feel like we need to bend ourselves to be successful, but a place where we can show up, feel we are acknowledged, valued as we are authentically and where we allow others to do the same, to feel the same, to experience the same.

I wonder if Russ likes this one. It was created by 3 people at the conference this year. We are talking about the manifesto and what's important and we came up with the idea that's really only the first 3 words. We value people. Period. Thank you. Everything we do that doesn't add value to the people and helps the people to add value is probably not helping in the bigger picture. There's a lot of conversations going on about systems, how to change systems, how systems evolve, how people play a role in the system and in all my years as a developer, as a manager, as a pupil, as a teenage scout leader, as a coach consultant, whatever, I've learned that most of us continually underestimate their ability to change a system. My parents gave me a good example of how to treat work in your life. They basically accepted everything they had during the day and complained about it in the evening. That's the pattern I adopted when I went into work and when I was thrown into a consulting situation which I hadn't asked for because like Russ I'm an introvert so standing in front of people I don't know and telling them what to do didn't come naturally to me, it still doesn't which is why I don't do it. But I learn that there's actually different kinds of workplaces because I saw so many different companies, and there's actually some people who are successfully changing systems. So over the past 20 years I learnt an important lesson in terms of why we underestimate our chances to change a system. It's because we very, very much underestimate our own ability to change ourselves. You've probably all heard the Gandi quote who said ''Be the change you want to see'' and he's also shown how hard that is, how much pain you can endure if you really want to have sustainable change. And I truly believe that change always starts with me and any change of a system I'm going to be a part of, even as a coach, as a consultant or whatever, it's going to involve my own transformation first.

So why do we underestimate our own ability to change? I've been looking into a mirror, honestly, multiple time of my life I didn't like what I saw. It was not as beautiful as what that mirror is showing. What I basically saw was that the assumption I grow up with as a teenager that I don't need to be arrogant because I'm no better than the rest wasn't really showing. I didn't really dare to tell that to anybody, it wasn't a helpful sentence to begin with, I was young. But over time I saw that many people although they know in their head that they're not very bad, that they are kind, intelligent, helpful, etc. they don't dare to acknowledge that they're special, that they're worthy, that they are allowed to be kind to themselves. And this seems to be a struggle that we all have. So how does that happen?

Why are we so afraid to show ourselves? Why does it hurt us so much to even think of talking about ourselves authentically, openly without any agenda or mask or persona that we push out in front of us. Where's that pain coming from? There is medical research that shows that a lot of pain in human beings, a lot of illness, is created because a conflict between two main drivers in human beings: one is the driver for authenticity, the driver for our own goals, the drive to be ourselves as much as we can be, and the other driver which is in conflict with that is the driver for attachment.

We don't dare to show ourselves because we want to be successful, we want to be liked or loved, we want to survive and we constantly have this feeling and also make the experience that as soon as we dare to step out of our masks, of our habits, of the expectations that we assume other people to have that they will hurt us. So we bend, we bend into roles, we bend into expectations, we create habits, and even when the situation that brought this habit into being is long passed and the influences are not working anymore, are not relevant anymore. We still keep that behavior although it doesn't make us successful now. So we create false identities because we sacrifice authenticity for attachment and in a corporate context that leads to a very, very big dysfunction, a big dysfunction that in my experience is the biggest source of waste we have in society and organizations and that's why I'm interested in talking about it, in sharing my thinking about it and helping you to spot it and to work on it.

When we enter an organization we have someone pay us for our presence. We trade a certain amount of our own freedom for a safety. You are allowed to ''boo'' if I mention this word. I think that is OK, right? I want to get paid every month so I give up a certain amount of control of what I do, how I spent my time to the organization that pays me. It's a fair deal. I think that in many situations we are not entirely honest and not fully conscious about how much we give up for what in return. And the result is that we are scared because we are trying to feet into something we didn't choose. We chose the workplace but we didn't actively, consciously choose all the things that came with it and we have a hard time dealing with all the judgment that is so prevalent in our society and in our workplace, right? We are assessed, we are compared, things are right, things are wrong, things are good, things are bad. Very, very often this is not situational in a context where it could be helpful, very often this is a very general judgment and that leads to shame. Shame is one of the most powerful emotions, one of the most powerful drivers, and it's very, very painful so we try to avoid it. And the dynamics that are created are called oppression. Oppression is according to Augusto Boal "a situation where there is only monologue although there could be dialogue". And this is exactly what I'm talking about when we enter the workplace. There isn't an explicit contract to trade my time and my presence and my creativity for money and there's multiple dimensions for this and there's a whole load of expectations, many of whom I probably make up because I don't ask what people expect. And all of that stuff that we don't talk about, all of the freedom I give up that is not explicit, that is not conscious, is oppressive. And this is harmful and this creates waste.

The waste it creates in organizations is politics. Politics is all the time and energy we spend not on results but on covering our asses. Making sure we look good, making sure the data looks right, making sure somebody else is responsible, right? Somebody else's problem, are you familiar with that dynamic? Right. I honestly think that processes and roles are mainly in place to ensure this. If we have long lists of RACA : responsibility matrixes etc. We know that it's not my fault. So I won't be blamed and I don't need to feel

ashamed, right?

I want to do a quick test with you. Could you stand up? I know we've done that already but this is the after lunch standing up and if someone actually asked you again the question you were asked in the morning why you do the stand up I give you one good answer: the brain gets 20% more oxygen when you're standing up so it's especially good when you're getting tired. So my friend Michael coined the term politics tax for the percentage of time and energy people in an organization spend on covering their ass instead of focusing on results. So a few questions: how much of your energy, how much of your time do you spend on looking good, do you spend on making sure something is not your fault, to bend data to fit expectations, to cover up that use or to cover thine ass? Please sit down if the amount of energy you spend in your organization is below 20%. Wow I'm impressed! Below 50%? 80%? Thank you for being so honest! I had a conference in Berlin last year where actually 2 guys were standing up when I asked below 80% and I talked to them afterwards, they were really in a lot of pain.

What's the choice we have? You 're familiar with the matrix, right? Red pill, blue pill, do you want to continue pretending or do you want to have a look at reality and see how for the…..goes? Do you want to open up, have an honest conversation, talk on eye level, get the roles, the expectations, all the habits to the side, see each other as a human being and see what we can do to achieve what we want to, right? That's the choice we have. So who of you is familiar with real options? Ok. Few of you. Real options is a system that you can use on multiple levels, the first level is for decision making, second level is for getting awareness of when in your life, in your language, in your thinking there's still coercion that stops you from doing what you want to do and lets you do what you have to do or think you have to do. And there are multiple other levels as well. I've covered these two.

The first statement is ''Options have value.'' That means the different things you could do are valuable maybe in different ways. It also means on other level that it's valuable to have options. You don't want to be in the corner like this guy, right? The worst situation you can be in is when you don't have a choice anymore or when the only choice you have is between different bad options. We had a lecture in Germany two weeks ago that was a situation I really did not like.

The next interesting thing about options is that they expire and it helps to be aware of how long an option is valid. I took this picture because sometimes expiring options can also transform into new options or a different kind of option which can be very interesting. I'll cover that in a minute.

The third thing is be conscious and aware of your commitments. That's where the coercion comes in. I bought a movie ticket, I have to go to the cinema. No. A movie ticket is an option. You can go to the cinema or you can spend your evening with a more valuable choice if that comes after you, bought the ticket. I have to go to work at 8 o'clock every day. No, you don't! You won't die if you don't. The only things we have to do is breathe, die and pay taxes, most of us, right? The rest is choice. And the whole interesting thing about the real option for me is getting more and more awareness of when we have a choice and why we make it. Praise differently, let things immerge, rather than focusing on the things you need to do, have to do, etc. Focus on changing the context so that beautiful things can emerge. Be open to what's coming at you. Trust that what can happen will happen! It's true that options expire but in a complex environment new options will pop up in the next minute and make sure you're aware of what is coming so that you can make use of it and exploit them.

Run experiments and celebrate failure, right? A commitment doesn't mean I'm committed to do this thing for the rest of my life or for the rest of the project. Make sure that you can change your mind, make sure to create frames safe to the fail experiments so that you can learn. That's how the culture comes in, right? How we deal with learning, how we deal with failure, how we deal with experimentations.

If you want to new more about real options, there's a great book, one of the greatest business novels I've ever read by Olav Maassen and Chris Matts. It's called ''Commitment'' available on Amazon and

elsewhere, highly recommended.

So what are the options we have now? We have this problem of politics, of spending so much time individually and as organizations to pretend to not look at the facts, not look at truth, not dare to be open, not dare to be honest. What are our options, what can we do? Some practical ideas? Reminder of the problem we're afraid and being afraid is not a very good state to be into act. So the first thing that I'm going to recommend is share your fear. Share what you're afraid of, open up a little more, trust and find people that have the same issue and talk about that. The greatest thing to overcome fear is to stop feeling alone so as soon as you get out there and say ''Oh, this is the story of my work life. I became a manager, a boss of a company with about 50 people when I was nearly 30. I hadn't lost for the job. I obviously gladly took it. I was young and needed the money, etc. and I found out that I had no clue what to do as a manager, I had no idea. The only thing I knew it was I was not meant to ask because I was the boss. That situation should be it, right? I f I had known this back then, I would have asked for help, I would have asked other people, my co-boss who came in the same situation as surprisingly as I had. I didn't have any conversation with him, I just assumed he would do better and try to keep up.

Story telling. The more we share about the experiences we make, the fears we have, the struggles we have, the successes we have, the more people can resonate and chime with their stories. This again contributes to the not feeling alone thing and it enables learning because we get multiple perspectives. And the multiple perspectives allow us to change our story. Someone was telling me on the break that he uses a scene from an animated movie to let people think differently about their situation and their team. So an animated movie with different funny characters and the people on the team then picked characters who's who. Oh, he's acting like that guy, he's acting like that guy. Oh, I'm acting like that guy, funny. They're getting a new perspective, they can choose a new story, they can change their reality by picking a different story to explain it. The human brain is quite malleable that way. Use that. Sometimes it also helps to just focus on what influences your perspective, right? I told you the story about my parents and how they shaped my expectations in the work place, to realize that, to talk to them, to talk to other people, very powerful, very helpful.

I developed a kind of feedback. It's called hero feedback, to offer a hero for the story of a friend, of a colleague as a present. For instance somebody might remind me of Sam in The Hobbit, no Sam the hobbit in the Lord of the Rings. Very helpful, very considerate, very brave but he is always in the second role so people don't see him. People don't recognize his contribution to the story. For that colleague, he makes that impression on me and makes me give him a new perspective on how he works, how he acts as a team member and he may say, oh maybe I'd rather like to be like Pippin, make a few jokes or whatever , right? These stories help. Story telling is very powerful. This is just one specific method you can use.

Temenos is a container for story telling. It's a method I've used for about a year now, with nearly 500 people and nearly every time it's a profound experience where people come back and say ''Wow I profoundly changed my own perspective on my life" because this container basically contains what I've already mentioned : story telling, sharing the influences that shaped your expectations and habits, how you think about life at work then sharing how you experience the current situation, how you work together, how do I disappoint my team, how does my team disappoint me, very powerful thing to share in a team. And then, once you've seen where your expectations and habits come from, which of these fit now then it enables you to look into the future and say "Ok, which of these things do I actually want to continue, do I want to develop and which of these do I want to let go of?

Another gift I got from my parents, they gave me a lot of humbleness and a very, very low self-esteem. I decided about a year ago that the humbleness is very good, it serves me well and that the low self esteem can just go right back under the Christmas tree, I don't need it any more. And that was a very helpful and easy decision after I realized it this way.

One thing we are in general very afraid of and in my experience has been very powerful for me and people

I worked with is to disclose more of ourselves than we usually do. A method or a set of methods you can use to do that are the core protocols, but there's other ways you can use as well: talk about yourself, right? Don't buy into this bullshit of ''don't bring your personal life into work''. Right? That only creates insanity, trust me. Share what you need, share what you want, do that in your team so that you actually find out what you want and the funny very obvious but hidden from many people, news is when you know you want where you're able to articulate it you chances to get it actually increase by a magnitude. So very, very simple tip but you need to share to find out.

Another very simple tip, another from the core protocol by the way ask for help. It's not usual, it's not normal and it's so helpful. So whenever you're stuck or even before you're stuck ask somebody to help you, right? Most people are glad to help, helping makes people happy. So the expectations that we have and the myth that we are kind of teach when we are small, that people think less of ourselves when we ask them for help is a myth. Ask someone for help and in most cases they will gladly help you and you will be so much better in the end.

One last piece of advice. You know where a bottleneck is located in a bottle? So in any system if it's a hierarchy, look for the bottleneck, leaders go first. So if you are in charge of a company if people follow you because they think you are a leader or maybe you are in a leader position your change will enable them to change and will make things so much easier and smoother if you start to open up and start to be honest if you start to ask for help. What I didn't dare when I was a boss myself. Try that, it is immensely powerful. If you do that that's my value proposition for you, you will get what you want at least much more of it, even though you don't know what it is, you will much better than now know who you are because you get all these perspective and all of this sharing and this reflection and I want you to be proud. I want you to be proud of yourself as a human being and go out and change the world. It is not that hard as you thought it is or as you might think it is.

Stay wired, normal is overrated! Oh, there is one more thing, if you happen to use rugby inspired practice in your organization it's not about this thing, it's about the ball. Thank you!

H□kan Forss: Flow Thinking

Hakan: Good afternoon! Have you ever felt like this? Hm. Well, I'm trying to give you some tips of how we can get out of this trap.

Who am I? My name is Hakan Forss. I'm a Lean and Agile coach at the Vega Group in Stockholm. I've been doing this Agile stuff for some time, Scrum and XP, but for the last few years I've more gravitating towards Kanban and Lean because I think it's a wider concept that we can apply even outside the IT part of our value streams. I'm a passionate cyclist if I have the time to actually do it and I especially like going uphill. The thing that makes it going hard to go uphill is they are like barbeque.

But enough about me. I will tell you a story about PSP and the Kara company. These guys are testers that are testing a very complex software in a very complex hardware situation as well. And let's see if you can maybe recognize yourself. Sp, what I'm going to talk about here, you will see that we will build up what you can potentially call a value stream up. The green breaks will be the value adding things, the things we are asking people to actually do. The yellow ones will be non value adding things but we need to do it to be able to do the green stuff. How it sets up the process. And then we have the red ones that are mostly waiting time. And in this example we have some blue ones as well. That would be when the supporting team comes back for some information or some requests to the testers.

So, we will start with the PSP company and they are setting up this very complex environment. They have to build up the environment. They have to load the software, get it ready to do the first test case. And they start running the tests and unfortunately they discover a problem. So the tester take the information they have, they write it in a Berg report, submit it in some electronical system and hope someone will pick it up and actually work on it. But when we have done that, we can't just sit idle and do nothing. So what we do is to actually switch over to the next test case. That means that we need to tear down the environments and we need to load a new piece of software and when we are doing that the support team coming back and say "We need some trace, trace, traces from where you execute this so we can get more feeling of how it works, what the problem is." And the tester says "Well, I'll do that as soon as I am done with this case, test case that I am working on right now. So, please wait and as soon as I am done, I will do that for you." He starts running the test case and unfortunately he discovers a bug in this version as well. So he files a report and sends off that to the support team. And then he switches back to the original test case, rebuilds the environment for the first version and when he's doing that the support team, another support team comes back and say "Oh, we need some more logs from the system to see what really happens so we can analyze the problem" and the tester says "I will get right back at that as soon as I am done with the first test case. " And he unrates some traces, sends that off to the support team and then he moves over to rebuilding the environment again for the second test case he runs. He gets the info back and while he is doing that, the support team comes back with a fix and he says "I'll try that as soon as I am done with the second test case." And he runs it and there's still some problems and he's sending it off to the support team. He switches back, rebuilds the environments, testing it out and while he's doing that, the support team come back and says "Well, I have a fix. Can you please try it?" And the tester completes the test run and fortunately it actually works. So he's done. He switches back and he completes the second one.

Do anyone here in the audience kind of recognize this situation? I get a few nods at least. Some of the companies that I worked with this is very, very common. So, let's switch over to the Kara company. So, they would do the similar thing. They would do the set up. It's the same. We have the same fault so they will find the same problem. But instead of actually writing it down in an electronic system, the tester immediately goes to the support team and says "We have a problem. We have a bug here. Can you, please come over and we can look at it together?" But the support team is unfortunately available and the

Kara guy he then says ”Ok. Fine. I’ll ask my coach to come here and we will actually do some process improvements work while we are waiting for the support team.” Because then I don’t have to tear down the environment. So, as soon as the support teams come back and once I have some information, the environment is up and running, and they can together sit and find out and get the additional information. And instead of moving to the next test case waiting for the fix, they get started on even some more improvement work while they’re waiting on the fix and eventually the support team comes back with the fix and because the environment is up and running, we can immediately start running the tests again and we can complete them. And in this case the first test case is already done. So, then we switch over to the second test case and we will repeat the same pattern again.

And as we can see we have completed the first test case much earlier in the second version compared to the first. And we even completed both of the test cases faster in the second version. And the ones actually performing better in terms of the whole system here is not the one that’s keeping itself busy all the time. It’s actually the Kara company that’s not focusing on getting the work to a completed state instead of just keeping busy. And as Deming said “A bad system will beat a good person any time.” And in this case people are working equally, they’re putting all their efforts into in but because of the design of how we do work, the PSP company is actually performing worse that the Kara company and the Kara company even has time to do process improvement.

So what we really need to do is to shift focus. We need to shift our focus from focusing on the workers in the system and keeping them busy and instead focusing on the work. It should be the work that’s going, the value that we are creating for the customers that should be in our focus. So the camera should be on the work and follow the work and not the workers themselves.

There is a great book called “This is Lean” written by Niklas Modig and Pär Åhlström from the Swedish School of Economics and they created this module where they talk about resource efficiency on one axel and flow efficiency on one axel. And the nirvana state of the process would be if we could have all our resources occupied all the time and the work that we are trying to, the value that we are trying to create for our customers should be occupied all the time with no waiting. And of course reaching nirvana is a little bit hard, right?

So, let’s look at this. So, we have the PSP company is really up in this quadrant. They are looking at keeping the testers in this case occupied all the time. The test environment is occupied all the time and the support teams is occupied all the time. But on the other side we have the Kara company. They’re focusing on the work. So, the work is occupied as much as possible, but not the workers. So, the Kara company is focusing on flow efficiency first compared to the PSP company which is focusing on resource efficiency first.

And these are really two different business strategies and sometimes being resource efficient might be a good thing. If you are a still mill and having the actual oven, not to be utilized all the time it would be very expensive because it’s very expensive to heat up the oven. And if you need to cool it down you have to clean it out and all that; it’s very expensive. So, in terms of economics in this case, keeping the oven busy all the time is a good way of looking how to operate this process. And to do that, what we need to do is to have a large amount of work waiting in front of the worker, so they will never run out of work. And we also need to have a buffer between workers and stages in our process so the next step in the process will not run out of work. What will happen with this is that the work going through the system will actually take quite a bit of time.

But if you look at another business, the way you are running your business would be the emergency services. How do we want the emergency services to behave? We typically want them to be responsive more than just being occupied, right? We don’t want the emergency service to say “ Well, I’m busy taking care of something else so you have to wait.” And your house burns down. So, what we typically want to do is to have a more focus on being available when there’s some demand. So, setting up that system would

be, we would try to avoid to have a queue of work in front of the worker. And we also try to avoid to have a queue of work between the workers. And Lean is really about the business philosopher where we try to focus on flow efficiency first before we try to move up towards high resource efficiency. Because this is typically much, much easier than moving from a higher resource efficiency towards high flow efficiency. So, how do we do this? We need to shift our focus. We need to plan for the less than 100% utilization of people and resources in our organization.

So, what we need to do is to plan the utilization of high priority work should be less than 100% and maybe a rule of thumb could 70% depending on the variation that you have in the system and this really acts as an insurance policy for shorter lead times. But it also creates options because if you are not scheduled to work on high most important things all the time, if something more important arrives, you can switch over and start working on that. So, it creates you options to do important stuff as you need it. And reducing the utilization also helps us in terms of handling variation in this work that we do, software development or other product development it's quite hard to know exactly how long a piece of work will to take in one part of the process. And we don't all the time know the arrival rate of work so what we can do by actually lowering the amount of things that we have planned in the system we can handle this variation and make things flow evenly through the system.

But the most important past as I see it is that we create space for improvements. If we don't schedule up to 100% we create the space to do process improvement so we can grow the capacity in the system. And combining this you will have something like this as you lower the utilization rate, the plan utilization rate in the system you can handle the variation and you also have time to do innovation and improvement work.

Another thing that we can do that comes from a flow base system and usually used in Kanban implementations is that we will shift focus of how we run our meetings. So, a typical Scrum meeting we will gather around the board and we will focus on who is working on what. What did you do yesterday? What should you do today? And maybe we discuss blockers. In the flow based way of running this kind of meetings you will typically do, you will first try to visualize the whole process, the steps that you have. And the second thing you will do is to focus on the blockers. What is stopping you from progressing? And you are focusing next on what you can finish today. So, a typical meeting in front of flow base board or Kanban would look something like this. That we will start from, typically, from the right and we will go upstream. So, we focus on the things that we want to have as close as possible to being at a complete state.

The next thing we will focus on will be the blockers. What is actually stopping us from delivering what we have planned to do? So, what are the things that are creating delay in our system? Then if we have time we will focus on things: What can we complete today? How can we move this thing from analysis to development today? How can we set up the system so that will happen?

And last, and typically if you have the time we might start to reason about the things we want to pull in next. What are we shifting focus from looking at tasks and the workers, more towards how we can complete the work and the value that we are creating for the customers.

And connected to this is to shift your focus around blockers and impediments. So if you are tracking your impediments on your boards what would you do with the blocker stickers or indication on impediments? How do you do that today? When is result, what do you do with the stickers? You throw it away? It's all done? If you do that you are missing something. If instead we take those blockers and we put them on a board of what the blockers and the impediments has been and you can build a diagram like this. You have a great opportunity to find out what are the delays cost in your system so you can start addressing them and then focusing on the things that are causing the most delay. And this is a simple count of the blockers but a team that I worked with, they actually positioning the sticky notes here based on time. So instead of just saying they count, they are actually the distance between the notes would be the time occurred by, occurred by the blockage. And that way we can easily see what is causing the most delay in the system.

So, you should really see the blockers and impediments as the gold and the gems in your system because they are the opportunities to improve the flow of your system.

So, the last thing I will talk about is the work in process. I probably need to take some water. Are we back? Yes. So the next thing I will talk is the work in process, the WIP. And I will talk about the thing that I think is the hidden side of WIP limits. So WIP limits will help you limiting WIP in your system, will typically help you increase flow in the system. I think the even more important part, the ones that is actually below the surface, that we don't talk about that much is how we can improve the system.

So, imagine yourself going to a canal. Do you come to the canal? And we are looking at the stream of water and it's streaming quite fine. We don't see any big disturbances in the flow. But what will happen if you start to lower the water level? Maybe you see some turbulence and you actually see a big rock. And if we lower the water level even more, we will see that the turbulence and the rocks are actually impeding flow quite a bit. So, by lowering the water level in the system, by lowering the amount of things that you have ongoing will help you see the things that are creating the blockage in the flow. So by lowering the WIP limit you typically will get more pink stickers, blockers or impediments that you can start to address. So, if you are drying up in terms of pink stickers in your process when you do improvements, try to lower the WIP limit even more because then you will see even more problems. So, lowering the water level in the system uncovers process improvement opportunities.

So, how do we set WIP limits then? Well, you can see this as a way of organizational friction. How much friction do we want to put on the organization? So, if want to have low friction on the current system, well, then maybe you said " A WIP limit on the average WIP that you have in the system today at 2." Because you just said it there, that means that it's very, very rare that you will hit the WIP limit so you can still operate your process just like you do today and it's very rare that you will hit the WIP limit.

If you want to create high friction in the system, you could typically set the two to have the amount of work people, workers that you have in the system because then it forces you to have pairing and things like that, that maybe you are not doing today. But that would create more friction on the organization. But this is only the first step and where you cannot get started is based on how much capital do you have to implement this in your organization.

But the next step is really the important part, is on a regular basis we should adjust our WIP limits to create more learning. So Don Reinertsen talks about this. He says you should on a regular basis, maybe monthly by monthly you will adjust your WIP limit by 20 to 30%. And that can be both up and down to create some new learning, but typically you will start to move it downwards. And when you do these new problems, new process problems will pop up and they will be evident for you and if nothing occurs, nothing pops up, there's nothing to resolve. But if you find problems, you start resolving them. And when the root cause of these problems are removed the process should stabilize and you're able to operate at that new level and then you just go back and repeat. And this way you can really start to really learn about your system and how you can improve the flow of work going through the system. What are the things that are stopping you from having a good and even flow through the system?

So, flow thinking for me is thinking in terms of flow efficiency first before we start to raise their resource efficiency so we need to focus on the work, not the workers and we don't plan far 100% utilization. It should be less. And we should really think about blockers and impediments as great opportunities to find out what is the problems cost in the system so we can fix them. And limiting WIP is really about learning. It will help you manage the flow, but it's the learning side that's even more important and many of the tools in the Lean community might look to be something above the water line but it's something else if you really look below the water line So, we have a Kanban system with limited WIP but we also have other things like even out of flow in the system that is really about trying to understand how the system works.

So, the question to you. How do you want to operate your processes? Do you want to operate in the high resource efficiency side first or do you want to operate in the high flow efficiency side first?

And with that..Thank you soo much.

Marcin Floryan: #NoLearning

Marcin Floryan: No, this is not a picture of me, this is a picture of my friend, Ian. I've worked with Ian a few years ago and I was, oh no, look at Ian again, we were doing an Agile transformation and I was really new into this stuff so I would go to all the conferences and I would read all the books and I would do to Ian who is slightly more experienced and would say ''Ian, look at this book, do you know how much I've learned?'' and Ian would listen to me very patiently and then would go ''Ok, so what have you changed?" And I would go '' No, no! I've learnt all this stuff. I'm yet to change everything, but it's fascinating learning'' and then I would go to a conference and I would go to Ian and I would say ''Ian, look at all these stuff that I've learnt!'' and he would go ''Hm. So what have you changed?" And really I did not understand what he mean. But hopefully thought the talk today we'll try to explore that and find the answer to that question, what did Ian mean?

So, some years after that I went to work for an organization called Emergen and for about a year we were trying to build an educational product. And at that point a really, really had to delve into what is it that we mean by learning and how do we learn best. This is just one of the books that we've produced as a part of that one year effort. But it really started opening my mind to what might have been the question that Ian was asking me.

Right, my name is Marcin Floryan, you can find some of my thoughts on that blog. I currently work for this fairy animal. Actually no I don't. I work for a company named ''Compare the market''. And I've got a little question for you to start up with. So, who of the people who are here thinks that you are paid exclusively to learn? Right? Do you want to raise your hand if you think that's the case. Right. I can see 1,2 hands up. Don't worry I was prepared for that.

So, let me ask you another question. When you start a project, how many of you know exactly what the end result will look like? All right. Nobody. So when you start a project how do you know exactly what set of requirements you will implement? Nobody again? So when you start a project how many of you know, before you start how many production servers are going to be? I can't believe it, nobody is raising their hand over here!

So, for me the process of going from the state where you don't know something to the state where you actually know how many servers are in production, what you have built and what requirements you've implemented, I call that ''Learning''.

We live in software development land which is products development for me. Learning is what we do, going from an unknown to known and I would say that all of you here are paid exclusively to learn. Because learning is what we do every single day, even if we don't recognize that.

So I thought it was really brave of Paul to invite me here and I will ask him to cover his ears now because I'm going to tell you: today you are not going to learn anything. Ignore that. I'll tell you more in this conference you are not going to learn a single thing, and this is not because the speakers aren't good, they're excellent, right? They're the best you can get. Let's try to explore. Why might I'll be trying to put thought that heresy. And for that we need some great philosophers and I like how Russ started us and let's go back to the ancient Greeks. I cannot teach anybody anything. I can only make them think and then to Socrates' kind of quote, Plutarch had a beautiful explanation why that is the case. For the correct analogy for the mind '' It's not a vessel that needs filling, but a wood that needs igniting.'' Right. I'm hoping to ignite a few branches tonight.

So, in order to learn and to actually battle this no learning problem we need three things. We need a pencil, some cogs and a thumbs-up. Makes sense? No, no of course not!

So, what I would like to show with you today and if that's the take away you take, that's it, we're done. Learning to be effective needs to be primed, deliberate and validated. Ok? Let's explore what that means.

Primed learning: so, very, very early us as human beings discovered that there is this land that we live on and there are all of these oceans that we would like to explore and there are actually more oceans than the land. So we started king of sailing out, further and further away. Until we actually built ships big enough so that we can sail from one land to the other. And that was beautiful but unfortunately we realized that it was a little problem as we started sailing further and further away. That was the problem.

Usually, on an average length crew voyage, half of the crew would die of scurvy. If it is estimated that between 1500 and 1850 about 2 million people died if scurvy, 2 million sailors. So rather a big problem. So here is Lord Anson who took a bunch of ships for 3 years naval expedition to fight the Spanish and on his expedition, on these beautiful ships he took about 2000 men, all right? So I would like all of you to stand up now, to try to kind of emphasize what happened with this men. So we've already been standby up a lot so I'll ask to do it again, please, right? Stand up everyone, please. You represent 2000 sailors, the 2000 sailors that sailed with Lord Anson for three years, right? Now, how many of you think come back home? I would like you sit down if you think you made it home, all right? Some people are sitting down so what I'm going to ask you is, I'll tell you. It was 600 of them. A third made it home, right? So let's make this a third of the room, you guys can sit down, please. The rest of you, I'm really sorry, you didn't make it, ok? But there is one good outcome. You went fighting so even though you died, it might have been honorable death if you actually managed to kind of fight some enemies. So, how many of you people standing you think actually died in battle? How many do you think should be standing? How many of you should be sitting down? So, all of you just sat down, please stand up again. Three men died in battle, all the rest died of scurvy. Thank you very much, sit down, please!

Now of course, I've asked to stand up and sit down to kind of experience this a little bit but you already know. We only did this little exercise so your brains got a little bit more oxygen so you can actually absorb this talk a little bit better.

So here comes James Lind, a Scottish guy who was trying to actually find the problems and he was the first guy who did clinical experiments. He took sailors who were suffering from scurvy and he divided them into nice little pairs. And each pair would be fed with different food. And he found out that one pair who would get citrus fruit was almost instantly cured of scurvy. Right? So, what he did, he shared that with everyone and what people did: that can be the food, we'll ignore that!

So there's James Cook, right? Two big expeditions about 1750-1760. He takes a ship, sales around the world, discovers Australia, comes back home, people think: ''Wow, he's a great man!" He goes away, takes his beautiful ship, sales around the world, comes back home and guess what? Not a single man dies of scurvy. Ok? Everyone's surprised what James could do. Well he had plenty of this on board, that was carrot marmalade and he had plenty of this, some sauerkraut. So, actually he took all the things he thought might influence this, but you know what? Actually no learning, Cook got a medal for it, but the rest of the navy just ignored that outcome and for another 60 to 100 years people kept dying out of scurvy because they didn't have the right food when they were sailing.

This is Vienna General Hospital around 1840, ok? And there were 2 clinics in which babies were born. So, you think, you know, there were just some experimentations with the sailors and some observations. So, let's look what happens when we have data. Clinic one: this is mortality rate of mothers who are trying to give birth in those 2 clinics. Which of two clinics would you rather give birth to a child in? Right? I bet it would be the second one, right? And there was one guy working there, Ignaz Semmelweis, a doctor, a physician who was trying to really crack this problem. He was really, really uncomfortable that in one of the clinics the mothers would die much more than in the other one. So what he did, he tried to experiment and see what was the difference and it took some moms and years to discover. But eventually one of his friends died because he was stabbed by a scalpel by one of his students. The difference between the two

clinics is one head students that were being taught medicine, the other didn't. And in the clinic where the students were being taught medicine they would go from the autopsies to the mothers who were giving birth. Semmelweis went '' How about if you guys did this? Just wash your hands, see what happens. This is what happened. This is the mortality rate. Dropped off the cliff! Right? Fantastic!

I'm really sorry. Do you know what happened to the guy? He got fired! Because the physicians could not bear the fact that they could have unclean hands. That was completely out of their mind, it was completely so out there. So he goes somewhere else, to other hospital, where he tries to repeat the experiment. The results are exactly the same. So what does the other hospital do? They put him into a mental institution where he dies. No learning there. So after Semmelweis we have this idea of Semmelweis reflex which is our instant reflex to reject any ideas that do not fit with what we've already learnt. This is why we don't know. Right?

So what does it look like in software? Here is Kent Beck, and this is his quote about 20 years ago about functional language: ''In spite of their conceptual and mathematical elegance, functional programming languages never caught on for commercial software." They're dead. Anyone doing Java Script today? See the life!

So, you know, one of the reasons why we're not learning is because it is impossible for everyone to begin to learn that which he thinks he already knows. How often do we think we already know these things? So, learning to be effective needs to be primed, that means you need to be prepared, you need to have plans. You're learning before you start.

Anyone knows what this is? Looks like a spreadsheet. It is a spreadsheet and I borrowed it from Jurgen Appelo, and this is the spreadsheet that Jurgen prepares to track his learning when he's writing a book. You think that's his spreadsheet? That's what it really is! That's how detailed he really is about his preparation. Right! So learning to be effective needs to be primed, deliberate, validated.

Let's look at deliberate learning. My daughter is learning how to do ice skating and she's doing these lessons enough to after a set of lessons she goes to next level. And to go to the next level she needs to have completed a number of exercises. So, after she was about to go from level 2 to level 3 and the teacher said '' You are really good, but you need to do this one exercise better to go to level 3.'' So we go together on ice and she goes and tries to do some more exercise and what I see her do? She's doing all the exercises she already knows how to do, not the one her teacher told her to improve on. Ok?

Anyone's seen a developer's CV where it says 10 years of Java experience? Ok. They're doing the same thing over and over again. 10 years wasn't enough to realize that Java was the dead end something, no disrespect to any Java developers. You can put any language you want in there.

So actually, what we need is not just practice, we need deliberate practice. And the difference between normal practice and deliberate practice is that we practice not what were good at, this is what you're going to do, just working on your code every day, it is the deliberate practice which is practicing the things we are actually not good at, putting effort into improving our skills and our performance.

So, there is a gentleman called Mihaly Csikszentmihalyi. A little bit of Hungarian notation there for you. He came up with this idea of flow and this is how he was trying to explain what this deliberate practice is. So, the deliberate practice occurs when the relationship of your existing skills and the challenges you face is just right. Not too difficult, not too easy, because if your challenges are too simple for the skills you have, you are just bored. But if the challenges are too difficult for the skills that you have, you get so much anxiety, you can't do anything. You have to find the right sweet spot.

I've been talking a lot about practice but when people say about learning they usually don't necessarily mean practice, they often mean knowledge, right? So, obviously mastery of the knowledge alone isn't sufficient. We have to have both. And this is the one thing that I really learn at VFQ. That for our learning

to be effective we have to do both: theory and practice and I'm just going to put it here in front of you and I encourage you to go and research that a little bit more. This is something called ''Kolb Learning Cycle'' and that's all the universities using today to teach their students. What they're saying is in order for you to learn, you have to go though the whole cycle. And the cycle you can start it wherever you want. Usually it starts with some concrete experiments you've done something, you've heard something. Then you have some reflective observation, right? You do abstract conceptualization so you turn what you saw in what you think is a model in your head and then you do some active experimentation on that subject. This is why you guys cannot learn anything today, because thers's no way you can go thought the whole cycle, just during the conference. Hopefully you'll get a lot of ideas, but you have to think about them and you have to put them into action.

Now Epictetus also tells us something important about this deliberate learning. Now this is translated from Greek and even in English it doesn't make sense, so what he really said is '' If you want to do something, just go and do it, but if you want to stop doing something, you have to start doing something else." We are really bad at unlearning things. Children are brilliant learners because they don't have unlearn anything and we've not usually in that comfortable position.

So, I have a little story for you to illustrate that. A long, long time ago there was a beautiful castle and in that castle lived a king. And that king had an alchemist as many kings would, right? And why would the king keep the alchemist in his castle? Because he was hoping that the alchemist would find the right formula for turning whatever iron, ore into gold. But the alchemist kept failing so the king get really anxious and goes ''Now, that's it, you have 6 months to come with the formula so I can actually get my gold.'' So, the alchemist goes away, thinks about it and says ''Dear king, dear king I think I have found your formula. This is your formula but in order to get your gold you have to follow this formula really, really carefully, step by step, on your own. But remember, the most important thing of all, when you're doing that you have to remember never, ever to think about the dragon that lives under the castle.''

So I'm going to set some expectations and I'm glad that I've already seen this curve twice. As you go into deliberate practice, as you start your learning you are going to go from here, down here and up here. And the poor Java developer that I was referring to earlier probably sat over here. Never made the jump to the next level. And this looks very beautiful but in practice it won't look like this. So, this is an example of Jerry Wineberg tracking his pinball abilities over many, many years and his improvements.This is what the graph really looks like. And that's what you should be prepared for, that's what you should expect when you go learning something. So, learning to be effective needs to be primed, deliberate, validated.

Let's talk about validated learning. My children have quite a few books and for some strange, strange reason every night when I go and read a story for them, out of all the books that they have they pick just the one. And I open the book and I sit with them and I'll read ''Who lives inside this little house? It's Toeffl, poor Toeffl doesn't quite notice how lonely he's grown." And on, and on, and on it goes and every time I make a mistake my children know I've made a mistake. And if I close the book and I try that and try to memorize this, no chance. However, there's this book and this book I can just read even now, even today after 20 years. Foreign language. I think many of you might remember that. So what's the difference between the book that I read for six months every night to my children I can't remember it and a little poem that I read 20 years ago and I can still recite it here in front of you even though I'm still shaking nervous, right?

What is the difference? The validation. I had to read this in front of a teacher, I was tested on it. Fortunately my children only test me slightly on the book that I read to them every night about Toeffl. The important thing that we're trying to battle here when we try to get our confirmation, our validation of the learning is the confirmation bias which is our natural tendency to confirm what we think is true. We read a book, we think it had some really good ideas, "hey, Presto, I've learned it, right?" So how do we confirm these things, how do we make sure that it's not just me who thinks it is the case. Olaf was talking about how do

you get someone else to help you out. Some experiments with psychology students about training, it turns out that if you take students, put them through training one team and another group of students doesn't get the same training. Even if their training is worthless, the students who went through the training will remember that it has improved their abilities, their skills and that they had done better after the training. Even if there is no observable difference, confirmation biased.

Another think we've trying to fight here is something called the Dunning-Krugen effect which says that if we don't know that much, if we're not really very good, we tend to overestimate our abilities. Novice developers will think they're better than they really are. However, season really good developers will tend to underestimate their abilities and apparently this is tied to the fact that the young, novice developers overestimate their own ability but the season developers will overestimate other's abilities. And we have to remember that our memory is a save on read memory, is save on recall. If we want to remember something we have to recall something, but not only just repeat it, but actually try to recall it and use it in a context. Get that practical element into your learning.

All right! Any questions? No, no, I'm not allowing you for questions now. We'll have the open space for that. Questions are a very important thing, a very important way to validate your learning. And how many of you will be asking questions and how often would you like to ask a questions and you don't? How many of you have been thought at school not a query your teachers? Not to ask them any questions?

So, going back to Ian, now I understand why he was asking me '' What have you changed?'' because he wanted me to really learn and he already understood that true learning takes the full process and then only by demonstrating that I have done something with my learning. Only when I do that, only when I change something can I truly say that I've learned. All right? Do you guys remember? Primed, deliberate and validated. Repetition is good. Primed, deliberate and validated. Effective learning.

So, I'm going to finish up with this theory of learning entropy and this is just my idea. So, what I've been doing just now was this: these are the ideas that I've been knowing towards you. That is not learning, that is just a random piece of blocks that I'll have to pick up later on. But in order to actually build something from his blocks I would have to put some energy, some effort into it and I want you guys to remember that you have to put effort your learning in order to build something from the random facts that arrive into your brains. All right? So for me, when I started on this journey and I wanted to share my ideas about learning one very important validation would be to actually come up here and share those ideas with me. So I'm here because I'm trying to validate my learning about learning and I'm going to finish with this great quote from Seth Godin: '' If you've done something with what you've learned, then maybe you know it''.

Primed, deliberate, validated. Maybe. Thank you very much!

Kasia Mrowca: The Art of Saying No

Kasia: Hi, guys. I hope that you can all hear me, especially those sitting in the back. And I have surprise for you, if you're sitting in the back here's a nice lady with microphone. Just please, stand and show you. Yeah. I will be asking questions and if you're sitting in the front those questions will not be asked for you and if you're sitting in the back this nice lady will pick a victim, I'm sorry, a volunteer to answer a question. So, beware and better go closer.

And just, you know, starting my presentation I heard that the best presentations have an animal in slides and it's a particular animal called a cat. Because I don't own a cat and I own a dog I decided to, for the dog. I hope this presentation will be awesome. And yeah, what else?

Basically I love skiing and because it's a sport people cheer each other during the competition just to feel the strength of the crowd. Because it's my first talk in English I would love to have a cheer from you, so, maybe some applause, or, you know maybe some cheering and laughing. I can't hear you, guys. Oh, excellent.

So, moving on. We'll today, we'll talk about requirements and the art of saying no. And I have a few basic exercises to begin with. And your task will be just: read what is in the slide. So, the first task is really, really easy. I can't hear you. I can't hear you. Louder. It's easy. Come on!

Audience: Yes!

Kasia: Excellent. And now it's the hard task.

Audience: No!

Kasia: I can't hear you.

Audience: No!

Kasia: You know, you're the masters. You don't need me so I'm going. Bye! Oh, of course it was a joke. Basically we'll talk a bit about psychology and why actually saying no and disagreeing is quite hard for us. We'll, you know, talk about how to hear, to hear the needs. And, you know, we'll be talking about visualization, so translating from the business language to the IT language which you are all familiar with. And also we will talk about arguing and how to give arguments and in the end I will share some few tips and tricks if we have enough time, of course.

Basically, why should you care? Why should you see this presentation and pay attention? I'll give you one minute to read. So, basically SAP is my favorite example of a software. It's quite expensive one and in the beginning of development of this tool, in the early 90's there was a company called Foxmeyer Drugs. They basically bought SAP and if they only bought it, not implement it, they will probably survive and were keeping up. And they were quite stubborn and they bought it. They tried to implement it and fit it to their process and it didn't suit. SAP is an enterprise resource planning system and in those days there were like more manufacturing planning systems than enterprise and it simply was for manufacturing companies not for a pharmacy company. And because they were struggling with this, they went bankrupt. It's quite nice case study for this because it also shows that if we have responsibility among all people in the company and, you know, and we are cooperating then we are moving forward all together. And basically this talk is not about teaching business people how to do their business because, you know, they are experts. You're supposed to see there a YouTube movie so now you it's supposed to do, you know, to have a laugh or cheer. So.

Audience: Yey!

Kasia: And sometimes it is actually hard to know which feature is needed or which is not needed and this quite nicely reflects this situation. And returning to my SAP example. If we have a system which is basically for accountants and their beginning, the beginning of the SAP, it was a system for accountants and you decided for example to overwrite this system because it's quite old, maybe it's not flexible enough and you have in your company a visionary who says you "Ok, we don't have ready product, yet. We have a bit started, a bit started here and another bit here and it's not working yet but he's keeping with new ideas for new features. One of the new brilliant features is how about implementing Google handouts in our SAP system?" Brilliant idea. You know accountant have to talk to each other so we can allow them to talk. Maybe it's a brilliant idea but if we have, you know, bits of software started and not finished and our basic produce is not working and we are talking about new and fancy features it usually won't work. So, it's the time to say "Ok. It's a brilliant idea but first we should finish something. Deliver it. Ship it to the customer and then making new features." So, it's the basic example.

So, why you should. No. basically to support business decisions, to show that there are some constraints from our technical perspective that no analyze to deliver is for example as quickly as the business expects or from our perspective it's a waste. So, waste of time, waste of money. And it's not our money, it's the business money and they usually don't understand this because. Ah, we will see during my presentation why they like a business, don't understand it. So, I want to express also that we all here are responsible for creating a software and I really strongly believe that you are really well talented people here and you want to create a masterpiece not just piece of software about something which is really excellent and users like to use it. So, returning to this SAP example. Just imagine that if you have a system and your users hate it and you know comparing it to this, you know, for example to the SAP and how do you feel it? You are not feeling well, so it's sometimes really worthy to say that something, you know, is missing, something is not suiting. And if you just disagree then you can take action. If you are just keeping silence, you deliver it to the customer and he won't be satisfied.

So, there are commoner possibilities to great, great software because we all, you know, I think we all work in Agile, at least our companies think so. So, basically how many of us, of you deal with customers on daily basis? Please raise your hand only. Ok. Quite a lot of you. So, maybe the second question. How many of you is working in, you know, some kind of Agile methodology? Ok, it's interesting. Basically, Agile is about communication and communication is about customer also. So, it's quite a big difference in relations. But basically how many of you deal with some kind of requirements, with operating them or something? Just raise your hand. So, quite a lot. So, you know, if you're dealing with requirement, if you're dealing with customer it's also your responsibility to say that something is missing, that something is not ok.

Moving on how to actually say no and why it's so hard with about psychology and especially about sociology, about why no is so scary. Basically for us as human beings disagreeing with someone is scary. I know that you are all brave, that's why all are sitting in the back and want to have questions. Oh, there is, I see one volunteer. Ok, could you say why saying no is hard for you? If of course it is. The person who just raised the hand. Silence. Oh, ok.

Man: Now it works. Saying no is hard for me because I'm a harmony addicted guy. I don't like conflict at all. So if somebody asks me something that sounds reasonable I want to help and I want to make it happen.

Kasia: Yeah. Basically I want to show you that saying no it's not about conflicts so I hope that we'll go to this part.

There is also a thing called conformity and it's the best way to explaining about an example. In the early 50's there was a guy called Ash and he made this basic example, this basic research. Here is one line and here are two lines and there were several participants. Four of them were paid actors and one of them was an observed person. And then people were asked which of those three lines is exactly the same as this one.

And actors were of course paid to say certain response and for example if actor say "Ok, the same line is A." Another say the same. And finally what do you think? What the last person said? I need volunteer from the back. Oh, we have some nice volunteer there. But you don't have to raise your hand to just..ok. if you don't want to have one volunteer jus speak up loudly. What do you think?

Man 2: C.

Kasia: Oh, there is a C. Now, usually people were hesitating on saying A and 70% of people say A. So it's quite a big number and in the contrary if they were not saying loud their answers but they were writing them down, only 1% of participants make a mistake and choose the wrong line. So, you know it's quite a big difference. After the research they were asked why they choose the wrong answer. And they usually they say "I don't feel that I'm expert in this to, expert in this topic." , "I don't feel comfortable." Basically they don't feel comfortable and they don't want to disagree with the group. And sometimes if you just keep silence and you see that something is wrong with requirement is also like this thing and basically you can easily overcome it if you know there is this effect. You can actually make a different choice than the group. And it's ok, nobody will kill you. And I think in the end they will like you because you have your own opinion and that you are able to express it.

So, what else is cause that no is so hard to say? First of all assertiveness. So, the way of expressing our own opinion is usually like misunderstood and it's supposed, people think that it's aggressive. It is not aggressive if it is polite, if you say it "I'm sorry. I don't feel comfortable with your opinion, with what you are saying about and I have different opinion." It's quite ok. Of course if you're trying to say to people "Oh, come on. You're stupid. You're requirements are stupid." this is aggressive. If you just say that you have another opinion, it's ok and it's actually worthy. Don't be sacred to do so.

The next thing, criticism. Also people don't know how to give constructive criticism and sometimes they also say something like "Ok. This is stupid" and nobody knows what exactly they should actually do that's have better results. And we say, if we say that for example in our requirements there are missing acceptance criteria and maybe these acceptance criteria can make this requirement better for us to understand. Then, you know, it's more ok. And of course we are reading self helping books in order to make our communication better and it's not always working and I sometimes wondering why we are believing in, you know, writings of people who are not psychologists and if you are for example in a conference, if it was coding conference and I start talking about code and I'm not programmer, probably the audience will be "What she's talking about? Why to believe her?" Yeah? And with those books is something similar. Don't believe in that kind of books and the worst part of, you know, teaching yourself communication is that you think that you have bad communication style. And I think it's not true because all of us have their own communication style. And I think it's ok if you're not a market person to negotiate contracts and you have your own maybe a little bit awkward style, but you are able to talk, so you are able to communicate with people, then it's everything ok. Just move forward and don't try to use any of the fancy stuff that marketing people do because it's usually, it's usually only awkward and artificial and not working well. And because it's not working well you just keep being frustrated, frustrated, frustrated. So, please, remember have own communication style and go forward with talking about requirements and really important thing, try to hear the real needs. A customer has a lot of whishes like I say about this SAP example and this Google handout inside it. Ok? It's nice wish, but we sometimes should to wonder if our business is really needed. If we of course, we can deliver everything and they and then give the appropriate bill to the customer and make him, you know, to went, go bankrupt. But it's not the point and usually we kill, let's say, one customer, we won't get a chance to have another. So, unless we are SAP.

Ok, because my time is running out I also have few active listening tips how to, how to actually listen to another person, but we'll do it really quickly. And basically there are some stuff which are more needed and things that are really not necessary during the business relationship especially. So, basically if we have conference code, please, do not do multitasking. Multitasking is writing e-mails and talking. So, if you're

agreeing requirements, please, pay attention what other people are doing and saying. Don't check e-mails. If you are in a conference room, just don't you know, tweet or maybe your Facebook things on your phone just to pay attention. It's quite basic thing but it really works. Moreover, of course, ask questions about, the questions will talk about more later and I think that today and yesterday there will be and were a lot of things connected with asking questions that you should know.

Really important thing is also to summarize, clarify and paraphrase the things that you are talking about with the customer, client or whoever you talk to in order to agree requirements. And if you are paraphrasing, just use your own words. Sometimes there is also technique that encourages you to use the same word as the client did. But it may not work if you don't know the business vocabulary. I have for example from my life and I was agreeing terms connected with "taxi" but in the airline business. In airline business, taxi is this period of time when an airplane is on the ground and it's, you know, between taking off and going to the terminal and between and basically it is. And I thought that taxi basically means driving a cab, you know, like to the airport. So, there are completely two different concepts and, you know, it was misunderstanding. So better use the words that you know best. Yeah, so, there's everything that you can use.

And of course do not interrupt the speaker. It's a good advice that we usually forget especially if you are really keen about some particular technology and I want to say "Ok, you have really nice solution for this." Then we are just start to talk about technology. It's fine but we have to remember that we are, have to, you know, just dig in the requirement and talk more about it, and listen about it, not necessarily just overwhelm speaker with how fast the technology is. Ok.

So, now it's I think the most important part of my presentation. So, how to boost our communication and I love charts, schemas and things like that. Basically you have to remember that notation is not really important. If you have some remarks how to read actually a chart then it's ok. Remember that maybe you know notation like UML. I know UML but my business partners don't know UML, so who cares that I know it and I present it, and it's, you know, too complicated for them. Simplify charts and use something which is inspired by notation and remember to give legend to our people to understand those things. And basically I prefer to keep documentation in this way because it's easier to read this chart instead of document with 100 pages of description in a plain text without single table, without single picture. So, basically I keep it like this. And also when I agree with a customer about terms I also present this and talk on the basis of this. So, use the charts.

And if you are using user stories and you are bored with it, you can use this way, this thing. It's called story boarding. It's about, basically about drawing the same thing that you would write. So, how to use our feature that we want to implement. It's a fun, it's like to enhance creativity during the team. It could be an experiment after this, a conference, just to go and have fun and try to draw it. And moreover you also pay attention to customer, so it's more, more, you know, more focused because usually if you are using, you know, this standard formula for user stories we sometimes forget about our basic user that the stories, you know, address to. And when we are trying to draw it we are actually get more focused. So, yeah.

Mock-ups. The next favorite thing. If you don't have a tool you can use only pen and paper. You can also draw them on a blackboard or whiteboard. But remember, take picture afterwards. And you can also use a tool. I recommend tools that make Pdf files because first I use to do mock-ups in ..or visual studio and then show it to the customer. And guess what? They thought it's the ready application. They could click it. They said "Oh, come on. It's, there are a few features missing but could you deploy tomorrow?" and, you know, it's sometimes really hard to say them "You know, it's a mock-up. It's not working. It's only a concept." "You know but I can click it. I could put data there so it's working. It's enough for me. Really. I don't need any other fancy stuff. Just deploy it." Yeah and that's the discussion really especially with people which are not really familiar with IT and computers. And really from the business perspective there's a lot of

like this. So, better those which you can click things so you can have a flow there are excellent because even the less familiar IT user want thing that it's a real application, that it's a real software which works.

So, use the tools and if you have those visualizations just ask questions. If you have this feature that you really don't want to implement. Just make a smock-up and show that it won't work properly for the users. Because if you're just talking and you don't have "evidence", it's stuck. Then you can show it and really I don't like approach which I met recently "Ok. We just implement some features. We have no idea how it should working, but we'll keep implementing it." And maybe it could be ok if we have a customer who validates if it's good or not. But if we don't have a customer and we are for the fifth time implementing the same feature and implementing it taking it, taking like 2 months, it's no, it's not really effective and it's only wasting the money. It's better to make a mock-up and show it, ok, it's not working, we should change something. And remember not to be so perfect because there are great US people who can help us with making the software nice. It's only about how it should work. So, basically these were the basic how to communicate and how to translate those business needs into our technology language and vice-versa. So, if you have drawings, mock-ups, then communicate. It is really easier, especially if you are try to compare solutions. Just imagine that you have this feature that you really don't want to implement and some alternative. You can show and this mock-up. There is this alternative. It's excellent. Give it to our business client or customer or whoever, just to click for it and make them, you know, to love this.

And if this doesn't work, what can else we actually do to discourage from doing nor really beneficial thing? There are, you know, several ideas, and one of them is using story points. And now I need another volunteer. How do you think it's a good idea or not? Simply yes or no. Ok, yes, ok. Just move on.

So, you use story points and give high estimates. And yeah, I think it's a common mistake, we use to give high estimate if we want to scare our customers and why it's not enough. Basically customers doesn't understand story points. You know, you can try to teach them. You can say them "Ok, it's this and this. You shouldn't add it. You shouldn't do something, you know. There are only the measures for us. "They won't understand it, they have their concept of numbers and we should more or less take this bitter truth and move on. And moreover, our business clients are not stupid people, maybe one big estimate would be enough to scare the client. But if we always give big estimates, guess what, then they thought "Oh, my God, everything is so hard. Hm, they are lazy. That's why they give big estimates. They don't want to have challenging tasks." Or any other explanation because it's, you know, keep continuing. Moreover they love to map story boards to the things they know, for example to the money, to time, to hours, whatever. You can choose. They have special spreadsheets to map it and they have special magic formulas. And the most important thing, they don't know that the implementation is not beneficial for them because it's, you know, generally we are a team. I know that we are working in different business models and sometimes we are working in internal IT department and , you know, it's more feel like a team. And sometimes we are outsourcing company, sometimes we are offshoring company, and sometimes we are using any other model and sometimes we are just selling box to the market and we don't know our customers. But, basically, we are kind of a team.

So, what we can add to make this process easier is we can ask our customers to give business value to us, connected with feature and of course you can estimate risk. You can argue that story points contains risks. But, why not expose it? It would be easier for our recipients. Basically, if we have only thing like this our customer "Ok. 21, it's a number." And if you know a Fibonacci number, you know that it's actually quite big. If we give things like this, so risk in that t-shirt sizes…and you want to calculate it. But they calculate it nevertheless, but still. And if you ask about business value, then basically you have more information. And if you give this alternative and show that, you know, we can reduce the risk, the business value which is estimated by customer is likely lower, but still it's more worthy to, you know, take less risky thing and move on and have, you know, also quite big business value. It is also useful if you are product owner and want to justify why you are throwing away a feature because maybe it's not worthy to have a lot of

effort and take a risky thing, it's really low business value. If we don't force people to think about business value, they tend to say that everything is important and everything should be tomorrow. And, of course, our clients could behave like us so they could give us big estimates for the business value always. But we should communicate.

Ok, so, the art of giving argument it would be like quick because we now know how to present it. We should, you know, identify alternative and basically show it. And, you know, compare them. So, we should compare the things that are inside the alternatives because maybe this need here is really, really important. We cannot just change the story, the feature, because this is a crucial part of it. So, basically what these things are doing, usually we are all busy people. If we have visual indication then we can make easier choice and also please, add recommendation. And because recommendation also is some like check point that the business actually take the right decision.

So, because I ran over the time, I'll just look really quickly through the tips and tricks. Diplomacy is all about, I said before so just encourage people take decisions that you want. So, remember that users will use your apps so create a masterpiece not the ordinary software. And if you give too many options to choose to business, they probably won't probably be able to make any decision. So, alternatives, yes, but, you know, make one alternative. Then it'd be ok.

And quick summary. Don't be scared to express your opinion. Remember that if you have your own opinion, it's not mean that you are aggressive. Discover you own communication style. Don't be afraid to ask. Boost your communication by visualization. Ask, you know, nice things like risk and business value. Present alternatives, and it's really, really quick finish.

Thank you for your attention. And now cheer up Zuzi because she's next and she's have really awesome presentation. So, cheer up. I can't hear, you! Come on, guys!

Zuzi Sochowa: Mastering Retrospectives

Zuzi: So, good morning!

Audience: Good morning!

Zuzi: Ahhh, good morning! Can you please stand up? I really loved this practice yesterday so I would like to try it. So, how many of you think the retrospective is just awesome? You can sit down. How many of you think it's just a good practice? So, sit down. And the rest? You're still staying. Do you think it's horrible?

Audience: Booo!

Zuzi: Boo. Ok. Well, retrospective is an awesome practice, at least for me and I will try to explain why. What's wrong with that? Well, they're going with other the slides. I can still see them here. So, if you have good eyes, I can show you my slides here. The thing about the retrospective is that, if you have good eyes, there is a round table and. Ah, it's here. And my first experience with the retrospective was in US about ten years ago and I didn't like that practice much. I've been sitting at a table like that with my colleagues and we've talking about the same things which we like, loud, and which they would like to improve. So, I'm a bit introvert. So, I don't like to speak public about my feelings which they forced me. And these American colleagues, they've been kind of afraid to speak about their feelings because their manager was sitting there. So, it was not really a nice thing, but look at it, it got such a nice animal like that and this nice friend, understands what you are talking about. So, we passed this round and round and can talk to this animal and say "Ok. I like the last sprint this and I think they should improve that" and he understands. So, it makes it a little bit better. It is the real master of retrospective, by the way. Look at it.

So, it was not really anything good. And it happened to me that I am one of the very first sprints, we did the retrospective and my colleague from Prague came together to the US with me and we've been sitting together at the retrospective there listening some boring stories about really how to reply to those questions and say nothing. And then about the third person was my colleague and he just tell "Well, you know, I think it's good that we finally have a let and we can work together because previously we've been sitting in these cubicles and don't interact at all. So, that's what I like and what I would to keep." And then he said "Well, but there is something which I don't like much and I don't understand maybe. And one of them is the point estimation." He just didn't have a clue how to use and I think we should more cooperate so how about to try to pair programming. The habit to do this, this and that because I don't like that behavior. He was the first person who just speak up and say what he really feel. And that's something very surprising happened . The rest of our colleagues were kind of afraid and ok. Nothing happened really. So, let's join him and let's tell what we really don't like. So, from that time it changed a little bit and after some time we start like retrospective. We start using it and then I came back to Prague and teach my Prague colleagues how to do the retrospective and what to do with it.

Sometime later I just realized that, that it's about looking into the mirror. At that time they call it reflection meeting, anyway. And if you try to look into the mirror to yourself, you'll most likely see a bit different picture than you have in your head. In your head you are awesome. And in the mirror you sometimes see that your hair isn't good enough or maybe you have some problems with your dress or whatever. So, it's a good practice to look into the mirror and move the mirror to your team and say ok, how I see all my different colleagues and what's really inside those people. So, looking into the mirror. Sometimes you have a different idea about yourself and your colleagues and it's rather this. And there is no single truth. It's just about the feelings, it's just about reflection. But that's kind of thing.

Then, later on, I found out that there is a good framework which is good to follow during the right retrospective and it's about the facilitation. So everything you facilitate, the first step is to do a small introduction to explain the people you're facilitate "Ok. This is a retrospective. We do this, this and that and at the end there would be an outcome. "So, that's very important. It brings some kind of certainty to all the people who are listening to you, who came to the retrospective and, you know, might not feel comfortable, and so on.

And if you happen to be doing the retrospective over the phone with your colleague in India or America, or wherever else, there is one activity which you should do. The very first moment when the retrospective starts after you explain the format, you just to the weather check. A kind of a checking. I'm in. the retrospective has started. And the weather check is about that every one of you will just say if you were a weather, which kind of weather you would be. And one of you would say "I'm a sunshine." And the other one would say "Well, I'm a thunderstorm." And the other one would say "I'm a mid cloudy weather. " And another one would say "I'm the morning fog." The main thing is that everyone will speak up at the very beginning of the retrospective and they will just mentally forget on all those things they've brought, the retrospective. They are just in from that moment on. And the other thing good is that you pretty much understand that that person doesn't have a good day, so you might be a bit careful speaking with him. It doesn't say if it's personal or from work. So, it doesn't really offend you. It's just easy to say "I'm a thunderstorm today." You should be a little bit careful. But it's a good thing.

Second step. After the introduction which was really short honest stuff, nothing really important, general ideas, and at the beginning I was quite happy with this round table stuff, sending this round token thing and listening to each other. I think I'm quite good at moderating events so it worked for me. However, sometimes I've got the problem is some other teams that it was hard to follow and it was kind of awkward for people to speak happens and so on.

And once I've got a team which says "Well, we are best team in the company. We don't have any problems. We don't need to do any retrospective" I say "Well, ok. Well, you are a perfect team, you can be really such awesome" of course we have some tiny problems that, you know, they solved them right away. So there is no need to stop and do the retrospective. So, I was like thinking what to do as a team and this is a framework which helped me a lot. So, they draw on a big flipchart the chart and I ask the people whenever something happen which you feel is interesting just try to stick it and put it on top of that flipchart. So, as the days go just create such a chart. So, you all see what's going on and we can talk about it at the end of the sprint, which we did. At the end of the sprint I add to these two smiley faces. There are not smiley faces here and asked the people to vote how do you feel when that thing happen. Are you happy or not really? And I found out that it's really strange because you would assume that if someone feels good, the others would feel good about it as well. But we found out it wasn't the case. So, they have some things to discuss and this board contain a lot of funny stuff like Petrus play the same song over and over again even if you don't like it. And then another card says "Well, …the song, which is cool. We love it." So what I did I just put those stickers and give it to the people and do some small contest like "whoever gets the most stickers wins". There was no real prize, it was fun. And finally that team start to like retrospective. So that was kind of a good.

And then, the time goes and I've been empowered of some Agile transformation group in some bank and that Belgium guy who drives it, he said, you know, we are running Scrum over this Agile transformation group. So we are doing stand-ups, we are having back walk, we are doing some groomings and we are doing retrospectives. And I said "Yeah, yeah, I know the retrospective. I'm quite curios which kind of animal they are going to have." But there was no animal, there was no round token or anything.

They just used this kind of star and what he said, he said "Well, you know, write a sticker and put it on one of these segments." So, if you want to start with, if you want to do that part of thing a little bit less, a little bit more of if you want to stop it or if you just want to continue with that, and please use a pen,

the marker pen , I mean, so we can all read it. And make it one or two words so it's easy to read for all of us. So, we did and at the end of that activity I was like thinking "That's really cool. It's kind of a checking activity as well." At the very beginning you ask people to do something so they write a sticker, then you ask them "Can you please stand up?" and once you're down, go to the board, stick it here and read it aloud which is kind of a good. So, everyone of you has to stand up, do something, go there, read it. But it was kind of less personal and less difficult for people to do so. So, I tried the next day and it was kind of a good. The team of my other client they like it. And from that time I've been using this framework. But, sometimes you find out it's still boring and it's still not enough. So, I've been thinking. Ok. Let's use this because sometimes the testing problem, for example, repeats all over again and again and again so we need some specific retrospective to talk just about testing. And you can still draw the start, stop, continue, less and more thing all over it. But focus on the testing, which helps.

But, it's still not enough. Sometimes a team were saying "Well, we are bored with the retrospective." And it was good at the beginning. It's cool. We like it. But, you know, after some time it's all the time the same thing surround. We are not solving big problems anymore. So, I've been thinking how to make it change and different and how to involve a team. And the thing is creativity. So, one of the concepts which you can read over the internet is to draw a boat. And still to do stickers which involve the team but you can identify the thing that stops you. Put it into the anchors. The more deep, the more stopping. And you can identify the winds which just helps you to move forward. Again if you make it very far, over there, it means it's a very strong winds, so it helps you a lot. So, it's a concept which brings creativity into the team and which helps you to look at things from different perspectives again. And I was afraid to use it at the beginning, but then I found out it's quite a good thing, but it doesn't stop here. It's about the creativity, so if you just use the boat over and over again, it will not work.

So, the thing is the next time I said "Well, let's do it differently this time and let's identify the things what makes me smile during the last sprint and put the cards here in the middle of the smiling face. Because I really would like to know what makes you happy. And then another thing is what makes you unhappy, what would you like to improve or change or don't like put it outside of that smiling face. So what I'm saying doesn't even have to be complementary. We don't care about start, stop, continue, less and more. We don't care about what we like and want to continue with or want to improve. It could be what makes me smile. That is one category and the other one, well, let's talk about some changes.

So, once I've been preparing this presentation I've been like thinking "Let's ask my clients if they have some pictures from their retrospectives." And I have got these three pictures which I would like to share with you. So, this one is kind of usual, you would say, but still has those two faces, there and there. And it's good and bad. But these 2 faces are kind of, you know, they've been playing and they enjoyed during the retrospective. You see, it has some personality. So, if you draw it, even if you do it this way, you just bring some energy into the team and help you to have more fun. So, that's first picture.

The second one is the boat which I just draw. But, you see it's not just the boat, there is a captain over there, there are some things here. They try to make it personal and funny. But then I have this picture. This grandmaster she sent it to me and said "Well, it was drawn by a whole team." So what I just said when I came to the retrospective I said "Let's draw a sports car. A Ferrari." So here is the sign of Ferrari. I don't know if you see it, if you recognize it. That's real Ferrari. And you can see that one wheel is blue, second one is red so it was really drawn by different team members. So, they all came into that place and draw that car and she said at some point of a time one of our producers said "you know, we should draw a parachute which will prevent us from moving fast." And then at the end someone draw the wall. Once we hit it, we are dead. And what I like on that concept is that I never really tell them "Well, you should be creative and involve the whole team in drawing a picture in retrospective." It's means just talking about some basic retrospectives and they invented it by themselves. And they're becoming fun, they enjoyed that. And there's a team which is really healthy because they can having fun during the retrospective.

They are self-organized to create even the content of that retrospective, the format.

So, let's go back to that thing. Part of the second stage of creating ideas. Sometimes it's happened to you that you have so many ideas from the team that it's very hard to make it kind of clear what we are going to talk about. So, I always invited the team next to board and say "Well, look at those cards and think about how to group them, how to make them one single group with label so we can talk about the testing, we can talk about the review, we can talk about the product owner." And then another step is again all team next to the board and say "Well, do the dot table thing." So all of you have 3 votes or 5 votes, depending on your choice and you select which idea is the most important for you. Again it's a team who decides so. It's not a Scrum master anymore, it's a team who's deciding. And that's kind of a good thing. But he whole retrospective and the whole facilitation stuff on that meeting is about extending the scope. So at the beginning we just knew we were doing the retrospective to learn something which should improve our processes. And then they extend the scope and they have 5 ideas. And then out of those 5 ideas we generated options had to address those areas as to how to improve them. So, after all, in this picture we have 16 options. So, they really extended the scope. And of course we have the most options from the idea which has the highest number, the highest number of the votes. So, it's the most important for the team. So, we spent more time on that one.

What I realized that sometimes it's easy to generate the symptoms. It's very hard to identify the root cause. So one of the favorites I like because it's nice fish bone stuff draw is not only ask what's wrong, what should be improved, what we are going to do about that but is ask where is that happening and who's responsible for that, and why is that happening and when. All those questions will just help you identify real options, what you are able to do with that. You're not still selecting what exactly will do the next sprint, you're just generating the options so you are just expanding the space. So, the most common question is "what else?". Another format which you all knew because it was last ACE! T-shirt is 5 why's. So, ask why 5 times in a row and understand the root cause. I don't have to explain that, I guess.

So, the next step once you understand the root cause it's about the action items. And the previous steps were important, was good but you often hear teams which are saying "Well, in our retrospective we talked about it. We complained a bit how hard our life is", but there would be no action item in the end. It's just you know, have a nice talk, nice conversation. But that's not enough. The goal of a retrospective is to create concrete action items which are assigned to concrete people teams members and they are able to finish those action items within the next sprint. You don't have to solve the huge area about non functional product owner for example, but you should address it.

And there is one thing which is kind of team health meter. It just says how good your team is. And you can count it. So, how many of your action items is going into the team. If it's half-half you are not healthy enough. If it's like 95%, that's what you want to achieve. You want to have all your action items coming to the team. So you may want to say "We, as a team want to have a different product owner." Well, if you just say "Someone should change the product owner on the role." Nothing will happen usually. So, what you may want to agree on is: I will go to the product owner and explain him how to change the behavior. That's actually an item which is going into the team. And that's what health a team are doing taking is are taking the responsibility. And we can change.

And there is another thing which kind of say if you are healthy team or not. There are lots of studies that successful teams despite if they are Agile or not, they are having 4 positive things for every one negative. 4 positive to one negative. So, you kind of have to help the positivity to grow and make it visual. So, one of the things is to do the positivity wall and if you like something, just put a sticker in, write it here or on the flipchart. We have no bugs, we have good sprint results. We just have fun or went for beer on Friday which was cool. I've been on holiday. If you see it, it just increase the rate of success of the whole team. So make it visual.

And finally, we've been talking about starting and it's time to do the closing. So, it's integral part of your

retrospective. At the end you should say "Well, there's the action items which we all decided to take into account and we will deliver them during the next sprint. Very fast. What a very important. And that was all about functional teams which are willing to speak. They might be afraid, so, they might need a friend, Winston, but they are talking at least.

But sometimes you have teams which are not talking at all. They are afraid or they don't trust each other. They don't trust their manager. They have personal problems with some other colleague. They got fights in the past. So, those teams are not willing to say anything in a retrospective. So what so you do? Just draw such a nice picture, stick it on the doors to the retrospective meeting and as a team to put a sticker at the personality they feel to be.

So, first of them, over there is a explorer. There's a person who's really active. You want to have all team members as explorers. Of course, that's the ultimate goal. They are going to the retrospective with high energy, they are speaking loud. They are saying "wow, that's cool. We will do this and what about if we do that."

Then we have a shopper. Is the person who is going to the retrospective with this cart thing, shopping cart, and think like "Retrospective, yeah. Sometimes I learn something but in general I don't expect much." They are more quiet, they are quite easy to involve. So, if you have high energy you can involve them and they just forget they came with these low expectations.

And then you have that nice person and that person is saying "Well, it's better to be at the retrospective than at my work office because I can have a rest, I can just, you know, sleep mentally. It's easy. I don't expect anything and I'm not going to be involved, I'm just having a rest. I'm on vacation" it's still not too bad. You can still get that involved, on board. But then, there is the last picture.

It's a prisoner. So, that's the people who are saying "I'm here at the retrospective because my manger said I have to go to the retrospective. I don't like to be here. I don't want to be here. I just hate it, all those meetings. And I don't want to speak up because I don't trust the other team members, for example." And if you happen to have the team full of prisoners, it's better to stop and say "Well, I'm not doing the retrospective anymore. Let's talk about us as a team and what I usually use for that teams which have personal problems and prisoners there, I just talk "If I were you I would not like to be working in your team." Let's talk about what makes us team good. Let's talk about the values. So, what do you think it's a good team? And I say something after some time. And what makes you flourish? What makes you awesome team? You personally, here. And then we talk about what makes things difficult. So what are we going to do if there is something which is going wrong and then accountability? What do we really do to actionate them? And there's kind of a fair retrospective of designed partnership on ends, on how we are going to work together and be together. And then the next time, it's chill to remind a bit if they are able to do the regular retrospective.

So, to summarize it, it's about the change. I used retrospective format to make from individuals a team which is cooperating, sharing and having fun. And , the last thing. Well, do you know that grog is very smart animal, right? It can say no, accidentally. If you ask the frog to jump into the boiling water, it will say "No, no , no, I'm not going to boil myself up." That you accidentally happen that the frog is sitting in a pot, which is cold and you start heating below it, it will never notice.

And now, please, stand up again. So many of you happen to be sitting in boiling water without even realizing it? You can sit down. Well, I did. I've been working for a company and I didn't like that job anymore but I was just thinking I like that company because ten years ago it was a great company. And I've still been working there. I should have jump off and quit. And the retrospective helps me to jump off, you can sit down, to jump off the pot fast so I'm not boiled up. So, think about that frog and do the personal retrospective often because that's the most important thing. It can help him but it can help you as well otherwise you will be boiled as the frog because you haven't even notice you should change. Ok?

That’s me by the way. You can recognize the colorful hair. Thank you very much and enjoy the rest of the conference.

Gitte Klitgaard and Lilian Nijboer: Why Change is So Hard

Gitte: Hello. I've never been talking to a microphone before so, please tell me if it's not going through.

Lilian: Thank you for taking our time to sit here and not outside in the sun. We really appreciate it. If we talk too slow or too soft please let us know, or too hard. First I'd like to introduce ourselves.

Gitte: So, my name is Gitte. I'm an Agile coach and I'm from Denmark. I work with a lot of different companies about how to introduce change and my main topic is ''How do we make the world a better place'' and "How do we help people to have courage at work and in their personal lives?"

Lilian: So, I'm Lilian. You can find me on twitter with double l's. I'm an Agile coach, as well. I'm from the Netherlands. I love to help people to have more fun in their work, to become more effective, to make happy customers so that day actually are pleased with what I do and they go to their work with pleasure every day. If you want to know more about ourselves, we only have 30 minutes so we're not going to say anything more just find us around and talk to us. So first we'd like you all, in good tradition of this conference to stand up. I would like you to change something about your appearance, anything.

Gitte: Ok. So, now I would like you to change something about the person to the right of you.

Lilian: If you don't have anyone to the right of, you just find someone to back of.

Gitte: Ok, thank you, you can sit down now.

Lilian: Kasia mentioned this morning that she did not have many pictures of cats, as she's supposed to have so we're going to make up for that. This is the first one.

Gitte: This is the general reaction of people to change. Why? Why do we need to change? Why do we need to change again?

Lilian: It's a natural reaction and it's ok. Sometimes people think that people who have fear or who have resistance to change are difficult people, people who do not really want a change but often that's not this case they just need more information. They have problems with the uncertainty, we all have problems with the uncertainty. I don't like it when my supermarket changes everything around because then I walk in and then I have to think about what I usually do automatically like here's the milk here's the bread, now I have to look around. Where is everything? So, that's one issue and it's laziness to be honest, we are lazy. Gitte will tell you more about this. And on the other hand, there is uncertainty, about what will happen, what will be the case even if I want to change it myself like, I want to go to the hairdresser, it's my choice to go to the hairdresser but there's some discomfort because I don't know what the outcome is. So it's natural to have resistance to change. I would like to ask you all "Who'd change back what they change to themselves already?" Wow, pretty good. "Who'd change what someone else changed for them?" That's pretty good. Usually it's even harder to have change that stays. Of course, you are a group of people or Agile and you like change, you embrace change so any change will be sustainable really fast. But, to get people to change is the first step, but to keep the change sustainable is the next step. Often people say, I saw it on Twitter a few times, people don't mind change, but they mind to be changed. I think people actually do mind change even if it's from the inside, because it offers discomfort. Tom was saying that ''people don't mind the change that they offer themselves''. I think that's because they've already been through a process, they already have been through a process of changing their perception but not their behavior, yet, because perception is the most difficult thing to change. They've already been through the first process.

Gitte: And it also has a lot to do with something we heard yesterday from Liz that "once we change something, we actually change our identity''. And that makes it really hard for us to change but it also makes it hard for other people to accept changing. So I've been through a great change process since I came to Agile community. I used to be terrified of talking in front of my own team and I came ''Oh, I'll go to a strange country and talk in front of 100 people, don't problem''. And actually my surroundings find it really difficult because my identity changes, so even if we embrace the change ourselves, it is hard enough. Our surroundings also have problems embracing our change.

Lilian: So, we also want to mention, what was mentioned before by other people, changes about people. It's not about a new structure, it's not about a new way of working change. It's about people, and we should take into account their resistance, their fears, their objections. So, we should always take them very seriously because without the people we won't get anywhere.

First of all, we will want to introduce to you some reasons, why resistance is there, and then from our own experience we will mention some of the stuff that we think could help, help ourselves if we feel resistance to change or help others if we fell they have resistance to change. First one is fear. It's fear. Automatically when the things change some kind of fear come over us. I want to divide that in two categories, one is self doubt. Can I do this? It's like when you go to new school, when you went to high school you fell fear because you don't know ''Can I handled this school? Can I do all the stuff they ask from me?'' So you fell sort of fear to do something which is very natural.

I once worked for a company that, that had a lot of systems, a lot of old systems, and all of these systems were going be transformed so they only had like 4 or 5 systems left. And one of the older programmers, he was around 60, he had a lot of resistance to this. He didn't want to do it, it was like 10 years ago, so I was around 29, and I asked him ''What are you afraid of?'', which is really a bad question from a 29 year old to ask a 60 year old, because you check ''I'm not afraid of anything, I can do anything'', but in the and he told me that he wasn't sure he could learn a new language, and it wasn't foreign for him because he was still doing assembler and all the new systems would be written in other language, in newer languages. So I ask him ''Do you want to try?'' because it was almost on tension level. So, if he didn't want to try, we wouldn't force him. But" Would you like to try?", and he said '' Yes, I would like to try if I have the chance to opt out if I think, I just can't do it''. So, we gave him some security and he got some training and then in the first minutes when someone started doing small things, changing small things he was sitting next to them and just ask questions. Then, he would do small things himself with someone sitting next to him and in the end he could do some small stuff. Before he got turn 64 he didn't get into really big stuff but he did small changes. The good thing about that was that he was still part of the team. If he didn't want to do this we would set him a side somewhere and he would have been miserable for the last 4 years of his career. And he was really against me, I hope he wasn't against me personally but against the change I brought. He saw me as the Persian bringing it, and when I came back 2 years later I walked in and he was like ''Oh, are you going to help us again? I'm so happy to see you!'' So, suddenly he was a big fan, it's nice to have fans sometimes.

The other side is uncertainty. And uncertainty can be real, just as self doubt actually. Uncertainty is more about "What will this mean?", "What will this change mean to me?", "Will I lose my job?", "Will I lose a certain position I got within this organization?". In one organization where you are there was uncertainty but there was one certainty: no one will lose their job. It was really important to get that uncertainty out of the way so people will get a bit more relaxed. There was another certainty, though. They had to apply for a new job so they would be placed into a new position. And actually the uncertainty of not knowing if that would happen was worse than the certainty that knowing it will happen even though it was not good news for them. They really saw this as a bad thing. It's like when you're waiting for a train and it's late, you don't like that it's delayed but it's easier if you know why it is delayed. If somewhere there is a sign that's says there was an accident and it will be like 10 minutes later. So uncertainty is worse than

hearing something that's not fun.

Gitte: So, the other part and some of the things that does is that our brain is really, really, really lazy. So, we kind of have two ways of thinking basically, I'm not going to go into a lot of neurosurgery but one part of the brain is the one who does things automatically or almost automatically, and that is where our brain likes to be because it doesn't use a lot of energy. So, there are o lot of the stuff that we do, that we don't think about. So, if I have to stand here and think about "Ok. So, how do I actually move my legs?" So, I need to lift it and moved it over here at the same time as I'm talking and the same time that I'm holding something then I would be in problem. So, the brain has this mechanism of making things automatic, so it doesn't use a lot of energy on it. And every time we are exposed to change and every time we need to learn new stuff out brain needs to work harder which means that the natural reaction of a brain is to say '' Can I just do what I'm used to? It's just nicer and more comfortable and it's easy.'' So basically our brain is also very lazy. So, when we do all these changes we need to learn new stuff and it will take up a lot of energy and if we are not careful we will use all our energy. So that's also one of the reason is that our brain is so lazy. And there's a good reason for that because we need as much energy as possible to survive. And it's the same thing that makes us scared. And there is a reason to have all these fears because it is to help us survive.

Some people talk about not being afraid, but once you are afraid is your brain telling you something, and of course we have this initial reactions like if we see a big animal we other supposed to kill it or run away, at least try to kill it. And that's how our brain reacts. But there is a reason for this, there's actually a reason. There's a dangerous animal next to me. So, when we feel the fear we also want to look into and see why am I actually afraid of this. And is it because is just my brain being lazy and not wanting to learn something new or is it something behind it?

Lilian: Another reason can be cognitive dissonance. It's already been mention by Marcin yesterday. He called it cognitive bias is actually that our brain has tendency if we get information that doesn't agree with our believes and our thoughts that causes discomfort. You can see it with people who smoke. If they see another research that says that smoking will make sure that you die younger or you get lung cancer or any other disease, people tend to say ''Yeah my brother in law did this all the time. And our grandfather he smoked and he became 96 before he died." So we try to get different information to make, to justify what our behavior and our believes are at that time. There was research done in the 50's with a cult that believed that the world was going to end. And we've seen these cults before. They have a certain date and everybody, the entire cult was sure that the world was going to end. The date that they set the world was going to end, the world didn't end because we are still here. And the researchers thought that then because they had contrary evidence, that they will say '' Ok, we were wrong. Apparently, the world didn't end.'' But that didn't happen, they find new information to justify what they did and they got the date wrong. And there are two ways to cover cognitive dissonance. One thing is to get information or justify what you are doing to get it into the right direction, the other one is to try and change your believes. This is the more difficult one, as we said we are lazy, but that's the one you really want to achieve. You could also see that was a research done, with people who had to do a really tedious job. At the end, one of the group, half of the group is given a dollar and they were saying ''Ok, here you have a dollar. Would you please tell the people after you, that is was really great job?'' And the other group, they get 20 dollars and they said '' ok, would you please tell the group next to you that it was a really awesome job?" And then, after that they asked the people separately ''Did you really find it a nice job?". People who got a dollar actually changed their perception because the dollar wasn't enough for them, to justify what they were saying. 20 dollars apparently it was enough for us to justify to lie, so we don't change our believes. One dollar wasn't enough so we had to change our believes. So, it's really important if we get new information that we give it in a way that people can use it to change the way they believe at a moment.

Gitte: Yeah, and one of the problems that we also have is a lot of our culture is about having a zero fault

culture. And there's a lot of blaming. So, basically also sometimes when we tease Agile we say ''Ok. So, you go into the sprint and you commit to this sprint goal and you say that you will do this within the sprint and once we fail someone will go say ''Hey, you promised me that you would deliver this in a sprint''. And what I've seen in Denmark also is that during all this crisis this has just became more and more apparent in all the companies because all of a sudden they start firing people like in banking in Denmark. Banking has always been like if you had a job in banking you're good for life. You leave a bank if you choose to but there's no firing. And during this crisis we actually started having people getting fired, which meant that people were afraid of making mistakes. And we tend to have this, we tend to have this. We want to have this zero fault culture. We want to have the zero box software and we want to be perfect and we also do it as people not only in software. We do it as people, we want to be perfect people.

And there's always this about not having faults. And that also means that if I try something and it doesn't work, I fail. So what we need to do is to find something where it's ok to fail, where it's ok to make an experiment. And you can start up small. You can do something really small like, I tried to make the coffee yesterday and it was really, really bad in our apartment. That's really small. And I could be easily like "Ok, I'm the first one up and I need to make coffee for the other one so I have a responsibility." So, I could have panicked about that, and some people will do, even if it's a small thing that you've never tried before, but tried it out. So we need….

Lilian: everyone run because the coffee was awful.

Gitte: …yeah the coffee was awful. So, the coffee was really, really awful. Sorry, sorry. Today I had Zuzi instead and she made good coffee. So, but I tried it out and that's ok, and nobody said ''You made really bad coffee!'', except for now, but that's Lilian. So that's also something that you can do to have changes is to have build a culture where it's ok to fail.

Lilian: Another one, this is from Carter. Well, he mentions it a lot, no sense of urgency. If people don't know what they should change, why should do change, because they are told so. When people change because they are told so, that will became false sense of urgency. People will run around and they will make all kinds of, of, they will do a lot of work but they don't know what they are really working on, what is the sense of urgency. The sense of urgency should be a positive thing, it should not be a negative thing. If you go to your organization and you say '' Oh, my god, is really going wrong we make a shitty product. All our competitors are doing a way better job and if we go on like this we will all lose our jobs, we will get fear and fear paralyses. So, your sense of urgency should be your friend not your enemy, so what you should do is you should think of, look around what are our opportunities how can we became better, how we can became better than our competitors, look at it in a positive way. It's like I'm an Agile coach, if I would think like ''Oh, my God, Gitte is so much better'' and I can never became as good as she can. Then I would paralyze. I would stand still. What I should be thinking is ''Oh, my God, Gitte learned this and this and I think it really helps her. Maybe I should do that as well." And I look at my clients what do they need, how can I help them even better. That is what motivates people. If there is no sense of urgency it's often easy to see. There are a lot of meetings if there is no real sense of urgency with no outcomes. If you ask someone like ''Really, this is important?'' and he answer '' Yes, it's is really important. We should, I was make a meeting for next week'', that's like saying to your 5-year old when he said he needs to go to the bathroom ''Ok. We need to go to 2 more stores that we'll go home and you can go to pee." Then, apparently you don't see the sense of urgency. So, the sense of urgency is really important, and it's also important to do it in a way that people on the business floor actually understand it. I've seen it happen a lot of time that a CEO would mention what the sense of urgency is in the bad way, like ''We are going down. There are no sales anymore." And they do it with figures that people don't understand. So, they like walk away and they think like "Ok. Apparently there is a sense of urgency but we do not really know where it comes from." So it's also important that the sense of urgency it's clear.

Gitte: And it can also be the feeling of loss. So can be the feeling of loss that if you change something you

will lose what you already have. And it can be all kinds of things. It can be your identity. So, if you are the top programmer in the group and everyone comes to you and now you say "Ok. We are going to do Agile. We are going to work in a team and we are going to have that everyone works together and nobody is going to be the top developer", then you lose your identity. So there's a lot of places where you can have this feeling of loss. And again a lot of this because we like to do things automatically. So, once we cannot do this anymore we get a feeling of loss. And we have a contrast.

Lilian: Actually there was some research on the feeling of loss and it has the same stages of mourning. At first there is like denial like ''It's not going to happen, it doesn't have anything to do with me'', and then there it comes the guilt and angry like ''What I've been doing wrong all this time? Wasn't it right what I did?'' and I don't want to change it's been right the way it was and in the end if you are lucky you get acceptance and moving on. So, did it.

We are not saying that this list is extended, there were more reasons why people can have resistance to change. What we like to do now is from our own experience, tell you what we think could do to help people who experience resistance to change, or yourself if you experience it.

First thing is take fears and abjections seriously. Like we mention, we are talking about people so even if you think it's not rational for, to them it is important, to them it is rational. So, take it seriously it's the first step. Involve everybody, and involve everybody does not mean that you have one meeting in which you tell this is our plan, this is what we thought of and this is what you are going to do. Involve everything means from the beginning, because it is what Tom said '' If the idea comes from the people themselves, the resistance to change will be much less''. In the general people themselves know very well what has to happen, even better then management, who sits somewhere above everybody. So, involve everybody is also means that you cannot just have a meeting and say ''Ok. So, we have to change. Go ahead, do it!'' You have to give some guidance, you have to say like ''Ok. I think maybe we together we can see some areas in which we need to change and we can form some groups.'' And in the beginning you will guide them and then you say ''Ok. You can take over from here.'' There needs to be some guidance. You cannot expect from people what are being doing what they are tell to do for a long time, to suddenly do it themselves, to suddenly became self organized and know what to do.

Give information; it's the most important thing. Communicate, communicate, communicate. If you can't communicate anything, communicate that you cannot communicate anything. Just always keep saying what's happening, what's the progress, what does it mean for you. And also means that you have to give the information in a certain way. When people were told that they were getting, they were not getting fired, but they did get a different job, we sent out letters to the people who were getting, who did have to apply. We did this, we thought it through very well, we thought, we did this on a fri…, we first sent out an e-mail that today people will get the news if they would have to apply for a new job or not. Then on Friday when they got home they got the letter. So, they had the entire week-end to think about it and on Monday we had a meeting with all these people. The people felt horrible, they felt like they were put aside, like they were fired. I learned from that like we should have done it in person. There was a reason why we did not do it in person because we were afraid that if we're going to say to someone it will say all the project managers are going to be laid off and need a new job. But still we handle with people and we don't want to have people feel like this. The way you communicate it's very important, the way you give information.

Provide education and guidance, this is the fourth of self-doubt. This is actually kind of easy to help with. You can give people education, you can give them guidance, it's much better, easier then uncertainty, because uncertainty will be there. We will keep changing so there will be uncertainty because we want to continue this improvement. We want kaizen which means we will be changing. So don't try to comfort people in uncertainty, try to teach them to handle uncertainty.

And the last one, be open and realistic. If you do know there will be people laid off, if you do know some

things will happen, that will not be good news. Do mention it, do it in a specific way like I mentioned, don't just throw out an e-mail. But it doesn't help if you keep it to yourself. In general just someone will hear it, there will be gossip which will be much worse because we do not like uncertainty. We prefer the certainty.

Gitte: So, an important thing that is also when we go into change, is that, as I mentioned before, we like to be in a comfort zone. That's where we do things automatically. And when we go into change, we need to move outside of this because if we don't go outside of this we will not move out of the automatic system. But the important thing is to move out only into the learning zone. Well, things are a little bit dangerous. But, it's ok. And then you can go back. An important thing about actually moving a little bit, learning something and then going back to your comfort zone is that makes your comfort zone grow. And sometimes, it won't. Sometimes you'll go out, you'll try something and you'll say ''Fuck, I'm never going to do this again. It's so horrible, I might do it again but it's still not me." But, if you don't go back, what happens is that you'll keep moving out here. You'll move out into the danger zone which also means that when we go and change organizations if we just keep moving, keep moving, people will move out into the danger zones and they will get stressed, they will be demotivated, they will be really scared. So, when you do changes, and this also counts for yourself.

So, as I said before, I was terrified of speaking, I mean 7 years ago I got a call from my manager. You need to be able to say something in a room with your team. And I'm talking about my team I was working every day if we had a meeting I could not speak. So, what I did was I went out into schools and talk to girls about what's it like to be a computer scientist as a woman. Because I knew that, I was a woman, I am a computer scientist. How does this work? And I went out into schools because that was really safe. There were little girls there just listening to me. They would be around 10, 12 years old. So, that was kind of safe. Then I started moving into education fares, big fares where people would come and listen to what is computer science and I would talk about it there. And I kept moving back into my comfort zone and kind of-sigh-, relaxing and figuring out what went well. So, that was kind of my way of starting this change. And as I found out I had all these more successes and it was ok. And nobody is actually going to kill me if I say something wrong on the stage. I became more and more confident and I kept moving so now my comfort zone of speaking in public is really, really huge. If somebody asked me ''Could you go tomorrow and talk about retrospectives at a conference with 2000 people?'' I would say ''Yeah. No problem.'' Even though I don't have a presentation ready. So, that's when I got it. But it doesn't have to, it is not always that you can actually grow this. A way of growing this is taking all these baby steps.

Lillian: You can actually compare this. Yesterday we talked about flow. If people are too long in a comfort zone, they will get bored. If they are too long in a danger zone, they will get burnt out. So, we have to try to keep them in the yellow zone. They have to try to keep themselves in the yellow zone. There's actually also have a deep psychology if people have a fear of going outside. First you take a little step, you go out to the front door and you go back. If that feels comfortable you go to the grocery store on the corner. This is actually used for other fears as well, also for arachnophobia, but I'm not going to try it.

Gitte: So, what can we do? So, one thing that we can do is create a no blame culture. So, the prime directive for the retrospective is ''We truly believe that anyone did the best they could do with the resources at hand.'' So, if we grade this non blame culture, and something goes wrong, we take it, this is something we can learn from. So, creating a culture where it's ok to fail, where it's safe to fail, and make small experiments so that it is actually safe to fail, and sometimes we will fail big. But if we have a culture that's zero fault culture we will not make experiments because we know we will be punished if we fail. So, if we can make a culture where's ok to fail and not be blamed, then it will help a lot with change. Another really important thing Lilian also said ''Take people's fears seriously.'' Because you can go into an organization, into a team or whatever and say ''That's just stupid. There's nothing to be afraid of.'' That's not going to help anyone. So, take people seriously and listen to them, actively listen to them. Don't

listen to be able to respond, but listen to them so that you know what makes them afraid or resistant to this change. And you might even learn from it because they might have a valid point. And give them time, it takes time. You're not just going to be ''Oh, we'll take it to a 2-day grandmaster class so now we will be self-organized. Go do!'' It takes time and it takes time to build these things in.

Lilian: So, this is me.

Gitte: In my former job I worked for IBM. This is me being an IBM consultant at a client. So, I'm in India, at least I try to be so I do the crane. And this is what I'm doing here. So, this is my new apprentice. And this is one of the tools that I personally use when I go out to a client because of the fact that I'm not afraid to be a little bit silly makes people comfortable. Because what they expect a lot of times when they hear an IBM consultant coming out, first of all they expect a man, in a suit, really stiff, talking about serious stuff. And then they get me. What makes them relax because I'm not as scary. I know I haven't advanced in IT because I'm a woman. I saw a lot of men don't see me as competition which makes it a lot easier to come in and help them. But also the fact that I'm not afraid to do silly stuff. I'm not afraid to have fun. I'm not afraid to be who I am. It's part of what helps people realize that it's ok to do stuff. And also when I do mistakes, if I do have a training and I do a mistake, we'll see it. This is a really good example. I just said this wrong and I'm going to learn from it and that's what really you need to build into your culture. So, using myself as role model to help people. So, I'm kind of crazy enough to apply this to myself.

So, yes, that's a real tattoo. An d I am terrified of needles. So, I got this tattoo because to me this being brave enough to be myself and to embrace change is so important to me. So, one thing that you can do is also have a little courage and courage does not mean to remove all fears because as I said before fear are really, really important to us. Because they teach us something, it is stupid to go to somebody and ask ''Would you, please, stick me in my arm with a needle?'' that's stupid. So, there's a reason I'm afraid of needles. Put some ink in my arm because my blood will really like that. There's something wrong about that. So, the fears are ok. So, you think I'm talking too much?

Lilian: Our time is almost

Gitte: Oh, damn it. Ok. So, I'm talking too much so I'm giving the word to Lilian.

Lilian: Thank you. I thought you were done. Another thing is that if you feel you have resistance, if you feel that you are scared find other people. Talk to them, see how you can help them. It was Liz talked about yesterday. Make friends, make friends in other teams. See how you can help each other. Ask for help, don't be afraid to ask for help. It's not a weakness, it's a strength. Try to create networks. Try to connect to people. It's really important one.

Gitte: Yeah. So, ask for help if it's yourself. Ask for help. Take small steps. And listen to your fears and figure out if there's something real behind that. It might just be that you are afraid because of the change but there might be some reasons behind it.

Lilian: So, the wrap-up is ''Give people the power that do guide them. Aim for the heart, don't aim for the head. And don't booo over resistance. It will only cause more resistance and you will fail. And again we are dealing with people. And people respect, they deserve respect.

Gitte: So, if you have any questions, please, come talk to us. We will be here the rest of the day. Thank you very much.

Monika Konieczny: How to Wake Up and Feed the Mysterious Motivation Beast

Monika Konieczny: Hello everyone! It's really nice to see you. It's really nice to meet so many old friends and make new friends here and have so many vivid discussions about various topics. So, what I wanted to talk about with you, what kind of stories I wanted to tell you today, it was related to motivation. Usually the first thing if you think about motivation is this. It's the carrot and the stick, right? We usually think about the motivation in this way and as much as I like carrots, I'm not kind of sure if I like the sticks, we can go in more deep discussions probably. But this is the first, well in many cases, at least in my case, that's the first connotation. Then we think if it works or not. Yes! So this is the combined version, right? So you have a stick which is created from the carrot so that's the most efficient way of having the motivation at parts. If any of you knows where I can find this guy, I would be more than happy to have one.

What other kind of other connotation we have when we think about motivation? To be honest with you this time when I was thinking about the presentation, when I was thinking about this talk, I started thinking about only me, about my perspective, about my connotations. So, this is one of them. So, when I think about motivation of course Mr. Maslow is coming to my mind and his beloved pyramid, well, a little bit updated to the current state of my mind and probably there might be also other people sharing my view here in this place. When I started having problems with motivation I thought maybe there were some research, maybe there were some brilliant scientists who did some research, who were trying to discover what it is all about. How can I change it, how can I empower my motivation, how can I motivate myself? And what is probably more important, how can I motivate other people? So, I looked into the books and I started screaming very, very scared because it was, oh, my God, there were so many people, so many bright people doing so many research, doing so many experiments, having so many ideas how the motivation mechanisms work, how they can be applied to the daily life.

Probably you know most of the names which are on the slide like Herzberg's theory about the hygienic parts of the motivation and internal parts. So that the hygienic would be what the salary, the office, how is it organized and everything over there. And then when everything is ok with the hygienic parts of the motivation, then the internal motivation can grow. Then there was of course, there is the theory, theory x and theory y related to motivation. Theory x is that people are just bad, they don't like working, they prefer to be lazy and they prefer to just do nothing. There is also the theory y that people are great, they like working, they are self motivated. They like to be in, they like to participate in the management process, they like to be creative. So, a lot of conferring ideas, right? There is Pink with his drive theory. There is Adams about the balance theory that we as people we compare us to each other if the amount of effort we put. Do we get as much as other people? Do we get enough for ourselves? So we are doing constant, a verification of what we do. Mayo was experimenting when he was trying to distinguish if the light and external things are, how they are applying to the work. And of course Taylor who was looking at the workers. He was thinking how he can motivate the workers to work, to work harder, to work better, to be more efficient. So, when I started looking at all those research, the results, great papers, great books, I was "Oh, my God, how can I apply it to myself? How can I do it? It's all conferring. I am not sure if it will work."

So and I just thought "Maybe I will try to experiment on myself and that's going to be interesting." So I did. When I started the experiment, at some point I felt that I'm trying to self manipulate. Pretty bad feelings because in many cases motivation is somehow connected, well connected in our minds when we thing about, at least in my mind, when I think about motivation and motivating other people. There is this

grain of manipulation. I'm not always sure what is the correlation between motivation and manipulation, where is the barrier and how not to cross it.

So, after reading all those books I almost felt like reading this, so "7 easy steps to live your life the right way" and of course we all love those kinds of guides. Probably a lot of us read a lot of them and they helped us a lot and our life is so much better after reading and applying all the magic rules. No. At least it didn't happen in my life. After reading the research, the books and everything it was all great. Now I have much more knowledge. Now I understand some mechanisms better and in most cases when I was reading each and every theory I was like "Oh, yes, that will apply to me. That's how I work. That's the way how my internal mechanisms work." I was almost like one of those medicine students who are reading about, well things that can happen to human beings and they discover they are so much sick whereas they are not. So I thought that maybe, maybe there's somewhere inside of me there's some little alien called motivation beast and maybe, maybe there's some way to wake him up. Maybe there's a way to somehow negotiate with the beast to act as I would like to, as I would like it to act. So I said "Oh, great let's wake it up. Let's wake it up. It's going to be great. Just let me think what could be the alarm clock." On a daily basis alarm clock is not my favorite item near my bed. I really hate the clock, especially the alarm clock when it rings when I am not very ready to wake up. So, when I started using the sort of alarm clock to wake up my motivation beast by using all the great rules, great recipes which were found in the books. It was more or less like this. So, the beats really woke up but it was totally not under my control. So I didn't feel very well. Then I felt like "Oh, God, if I do the same mistakes to other people while trying to motivate them I can also wake up their beats inside and they will not feel really great. So, what to do?

That's a better version of the beast, to tame it a bit, to make it a bit more fluffy, to make it a bit more nice to the owner. But how to do it? The alarm clocks kill dreams. It's so very true, right? Every single day the alarm clock is killing our dreams and it's making them go away. That's why I usually, I was trying to get into this habit that after waking up I'm trying to write down the ideas and the dreams it never works as great as I would like it to be. So, you need to be very cautious with this alarm clock. The first question which I started asking myself is why do I want to motivate myself? Why do I want to motivate other people? The first answers coming to my mind were of course all those great that I'll be able to do better things, they will be able to change the world, they will be able to achieve more, to be happier, to be more satisfied. It is going to be all great. But it was just, you know, just words? They are all very great, they are very.. It's just words. So then I started .. that could help me. Oh, I'm thinking that my voice is going up and down. Interesting. So, I started thinking what for? Something is wrong? Something is wrong, right? I don't know. Something is wrong but I can still talk. Whatever is happening, that's a weird thing.

So, that's the motivation beats which is trying to overcome and oh, God. So, yes, it is interesting right now. The alien is really somewhere inside. So, I was thinking what for? Why do I want to motivate myself? What is the ultimate goal? And I think that that was the most important question. Not why but rather what for. It's much more difficult to find the answer to it. So, when you want to start motivating, motivating yourself, motivating other people, what I did, the first step was after all those messy steps related to trying to apply all the theories and trying to check all the things which were not working, I was getting less and less satisfied and I was very frustrated, I just sat down and started asking questions to myself. It's not an easy exercise. It's quite hard to answer those kind of questions. But if you know why are you doing it and more important what for are you motivating others, are you motivating yourself, it's getting a bit easier.

What is your definition of success, right? Is there success that you want to achieve? Maybe it's not all about the success. Living in this culture of success, everybody wants to achieve success, everybody wants to be a successful person. Success means happiness, means balance, means all the other things. So, there are different doors which you take, there are different definitions of success, right? Like do nothing, do everything, do anything. You need to think about yourself. What is your answer? If you think about work, because work is a very important part of our life, right? We are usually very focused on the work. It is

giving us a lot of satisfaction. It is also giving us a lot of pain because of various reasons. And you need to think what is work for you, assuming that the motivation part you want to use for your work.

So, in many cases at work you may feel like this. At least I felt like this. So, I knew that I was going somewhere and a lot of people who are going together with me. So, we were all together in it but somehow we didn't. Why are we going? What is the goal? Where are we marching? Why are we doing all of this? And it was very difficult, it was a very painful process.

I would need two volunteers actually because I wanted to show you one thing. I promise that this time nobody will be painted or anything bad will happen, at least I hope, assuming the technology will not try to kill me once again. So, I will need those two volunteers. By volunteer I mean a person who will stand up. I know that you are good at standing up because you are training all the day long. Ok. We have one volunteer. So huge applause for the volunteer. It's all for you.

Man: ….

Monika: but you're my favorite tiger. So, we will need one more volunteer I would like to join us over here. Who would like to join Iacob? So, let me find a volunteer because I'm sure that I will see one somewhere there. Usually they are hiding somewhere in the back. So, yes, you look like a volunteer. Hi. It's nice to meet you. I'm Monika. Come on. Let's meet with Iacob. He's your new friend, you'll like all together. You'll spend all night together probably discussing a lot of Agile things. I'm almost sure. Ok. They've just made friends. So, huge applause for the second volunteer. So, I what you to feel more or less like the guys, like the guys over there. So, Iacob, it's not just because you have glasses, because both of you have. So, we will do a short experiment, hopefully you will not see a lot at this point.

Man: I can see a thing like that.

Monika: But you can't also breathe and I'm not sure if that's the part that I, I would prefer if you'd breathe. It' s going to be easier for you. Ok. Oh, everyone sees everything. That's cool. So, right now. Yes, sure, I'll try not to break them. Ok.

Just to make sure that you have some sort of tool, power to. Ok, you have very huge hands, but. It's a compliment, of course. It is a compliment. So, you have, we are in a situation that Iacob is right now is a developer so he needs additional skills, additional power to make sure that he will be able to fight with all the obstacles. So, for the next few minutes you are going to be, you're going to be, you're going to have opportunity to be Iacob's, Tomak, you're going to be a product owner, a person who has, who has a sort of a vision and who has a project to be done. So, here is the project that needs to be done. Now let's. Is there anything I need to do to make other people see the project we are doing? Because you are probably very curious what is happening over here. So, the guys over there, the magicians, are doing their magical job. So, Iacob, is going to be now a bit more difficult, but you will manage, ok?

Iacob: Ok.

Monika: Ok. You as, you Tomak as the manager. I'm not sure if I can take the other microphone so that you can give some short instructions to Iacob to make sure that he is able to, to make sure that he is able to deliver the project. So, what Iacob has, as you are not seeing it, yet, is something like this. It's a very, very simple game, maybe you're connected. It's a wooden box with some holes, plus there's the ending hole which is the ultimate goal and there's a small silver bullet. So, what Iacob has to do is, having Tomak's guidance, he needs to get the ball to the final hole. Let's see if it's going to happen like this. So, you as the manager, as the Scrum master who is helping with removing the obstacles, you need to help Iacob with making the project come true.

Tomak: Ok.

Monika: And then the silence started.

Tomak: So, that's really tough task. I will, to make our language common, I will ask you to turn right or turn left, or go up or down. Ok?

Iacob: Like this? Left.

Monika: So he's trying to go left or right as you can see.

Tomak: Turn to right to move the table right. This is excellent. To turn left means to go the other way.

Monika: Look at what is happening. You can see on the screen up.

Tomak: To go up means to go further. To go back means to go back.

Iacob: Exactly like in playing.

Monika: Yes. It's exactly like in a play.

Tomak: Yes. This would be like flying planes. We will go up and turn a little bit left.

Monika: Exactly like an airplane. Look at their relationship.

Iacob: So, up.

Tomak: Further left. Once again. Up and left.

Monika: Ok, something is going wrong.

Tomak: Left. So, go right further and quickly left. Monika: No matter how Tomak is trying, Iacob is failing. In other words another failure in the project.

Tomak: Further, slowly further. Left. Again. It's really tough.

Monika: I'm only waiting for the moment when Cuba will start punching someone but that's why.

Tomak: Monika, I'm getting all,

Monika: Yes.

Tomak: I have the feeling that I'm getting demotivated on doing that because it's really tough for me.

Monika: Because Cuba is not able to do it? Oh, come on. You're a great manager, you'll be able to motivate your

Tomak: Cuba is great at doing this but it's really tough for me to give him a guidance.

Monika: Why? What is the biggest problem that you're dealing with?

Tomak: That

Monika: That he's not listening to you?

Tomak: That he's not seeing, he's not seeing the goal of this and not only seeing the goal but only we don't see why we do it.

Monika: So, as the manger or the Scrum master you should help him avoid the main obstacles which I hope was this or maybe giving you glasses would be even better, most probably. So, let's do it this way. Ok? Perfect. So, are you able to finish the project right now? Any chance? Too bad you cannot see. He's extremely exciting. Iacob is really trying to put the ball into the right hole and it's almost there, almost there. Well, it's just a matter of seconds because he's extremely patient, he's doing great. Yes. He did it.

Tomak: Yes.

Monika: Applause for the guys. The project has been just finished. Ok. So, problem solved. Thank you so much. So, as you started saying, you didn't feel really well with the project, right? Because Coba was

really demotivated because he did not see the goal, you were demotivated because..Why, why were you demotivated?

Tomak: Because I didn't see the reason, I didn't see the way to do it properly with his eyes being shut down and I didn't see a reason why we do it that way because it was really tough to complete the task. So I didn't, I have not seen the way forward to complete the task.

Monika: Mhm! Ok.

Iacob: We had some general understanding about what does it mean sharply left or slightly right, but.. So, we know what is up, right, left, down, but what the hell does slightly mean? It's not the same for him as it is for me.

Monika: Ok, thank you so much. Thank you for your feedback. Thank you for being the volunteers. Once again the applause for guys. Ok. So showing you this very, very short task and I'm really and I kindly appreciate your help, guys, because you were really great. So, in this kind of a short task, in this kind of a short game, yes, I know, I'm always into games and I cannot, you know , it's hard for me not to see the world through the games. But, using this kind of game you can easily watch the motivation mechanism. So, whenever you are blinded, you don't see the goal, it's very hard to be motivated to finish the task. It was extremely frustrating for Cuba because he really wanted to help Tomak with the project. He really wanted to accomplish this. But, unfortunately, because of his blindness, he wasn't able to. He was listening carefully to the instructions. He was following all the instructions, but it was too hard. Whereas Toamk, as you could see, he was even touching Cuba. And by touching, I mean he was trying to touch the tablet to help him. He was desperately wanting to help Cuba, but in the end it was rather not possible.

And that's something that is also happening in our daily life. It's something that is happening in the projects, right? It's so, in many cases, the team, we feel blinded. We feel blinded because we don't see the ultimate goal of the project. We don't see why this business decision was made or not. We don't feel as we are the partners, as someone is taking into consideration our voice and what we think. On the other side, the product owner, he understands a lot, he sees the whole spectrum. He has the full context but it's really hard to share this thought and to share all this information with the team. Even in all the bugs, even if, in all the training it seems to be easy because well, communicating is easy. It is not. In case the team doesn't see the problem of this type and you as a Scrum master or a coach you see it, sometimes it's just good to use this kind of short game. So, it's all about the motivation because once Cuba was able to see, once he had back his glasses, he was able to accomplish the task very, very quickly in a very efficient way and he seemed to be quite motivated to do it.

So, this is the thing. If we are being managed and we don't see the goal and there is this kind of situation, at least I don't feel good when someone is trying to micro manage me, when someone is just trying to give me very short instructions what I should do without involving me into it, without building, you know, a partnership, and giving, and giving me the opportunity to make some decisions and to really make the project happen, then I feel like in this case. So, what happens, what helps me in the daily basis having a rotated Scrum master, having a sort of rotated manager within the team. So, that, it's not only me who is managing, but also other people are able to mange me so that I fell how it would be in their shoes and they can feel how it is in my shoes. Then, there is a much better understanding. It's much more motivating for both the parties.

In this type of cases I need to cope with a gazillion of emotions. Some of them are positive, some of them are negative and it's not easy. I'm an extremely emotional person, which is good and bad, right? Because in many cases it's easier for me to show my enthusiasm, to show my passion or to show that I'm extremely unhappy or unsatisfied with something but in the other cases it's extremely difficult for me when I need to tame it. So, it's like in the elephant and rider theory, right? The emotion is the elephant and the more cautious part of myself is the rider. But in many cases no matter how hard they try to tame my elephant

and to tell him where to go, how to go, what to do, well, it's the elephant, right? Extremely powerful, extremely intelligent creature and that it's the animal, it's the elephant which is leading.

Well, when I was starting to hearing my internal voices and trying to understand myself and trying to understand what is motivating me, was that whenever I was sending the rider for the vacation, when there was mainly the elephant that was working for me, that is working for me. When I'm in an environment where I can let my emotions to lead in many cases because at least till this point I sort of feel that I can rely on my intuition in many cases and in many cases it's a better decision than going through a very, very long analysis. But this is what's working for me. I'm not saying that it will work for you. What is important is to find some space, I usually like doing it on the roof of the houses, on the roof of the skyscrapers to meditate, to try to listen to myself and to understand the forces which are inside.

There are many days, ups, there are many days when I'm being chased by the, many things, right? So, many urgent things, many priorities, many issues, many challenges and what I need to do to feel motivated. In this kind of state I'm totally demotivated, because it's just too much. I like stress, I like it under the pressure, but it cannot be like a beast which is just chasing me because that will make me extremely demotivated when I know that the goal that I have is just not possible for me to reach. There was a time when I was thinking about the motivation in the kind that I am motivated when I can be better, but better from other people, right? It was fortunately quite a short period of my life, but there was this time. And at this moment I had two options. One of them is that I'm better than the other person because I'm better. The other is that I'm better from the other person because the other person is worse than I am. And it's an extremely bad feeling. So, I was able to motivate myself by doing this kind of comparison but for a very, very short period and after that I felt extremely guilty, I felt extremely bad so actually I felt extremely demotivated. So, from my perspective, competing with other people is great only to a very small extent. If I try to motivate myself by looking at other people and trying to be better, better that they are, it's demotivating me, it's extremely demotivating me. But, as I said before, you need to listen to yourself because in case this is something that is motivating you, maybe it's good to find a good competitor, a person who can be your partner, who can give you a lot of feedback. That you are sort of balanced, that it's never a huge difference between your achievements.

I was also trying to change myself. I was trying to motivate myself by using thing things which are not, let's say, in accordance with my priorities, with my beliefs, with my nature and I failed miserably because when I had to control my behavior all the single second, and I had to control what I was speaking all the single second, it was a nightmare. After every single day I was exhausted, I didn't know why I am so exhausted. Then after a lot of talks with the good people, with my friends, and after self-analysis I came to the conclusion that I feel extremely exhausted and extremely demotivated if I act against my nature. So, it's very important to find an environment, to find a team where you can behave as yourself according to your nature.

Also setting goals which are not in accordance with your capabilities like this great cowboy who would love to shoot a lot of creatures but his kinds are just like that. It's also demotivating. It's great to set up a great goal, to go to the moon. For me this kind of goal would be to run the marathon, probably after 200 meters I would just lose, I would lose my lungs and all the other guts inside. I would just throw it away. And that would be the goal, if I set it for myself, I will be also totally demotivated. Half a year ago, it was the middle of the winter, but the winter in Cracow was like a very late autumn, nothing really winterish, so I decided "I'm going to start running again. Mhm." I set myself a goal that I'm going to run 10 kilometers the second day and you can imagine how I failed miserably and till this moment I haven't restarted running because it was just, just not my style.

Another thing that, which might be motivating or demotivating is when we do some experiment and the result is totally different than we expected, when we start using some tools and then something totally different happens. It's extremely demotivating, at least for me when I set myself a very strict goal and I

really want to achieve this particular result and any other result would be a failure. So, if I define it in the wrong way. Even if I achieve a success and I just don't discover, I don't feel it as a success, it's going to be, it's going to be bad.

Language, the way we communicate. That's something extremely powerful. We can easily motivate other persons or demotivate by using the wrong words or using the right one by using some kind of a trigger which will cause a lot of emotions in other person's style. By words we can build those kind of bridges between ourselves. In many cases those kind of bridges which are helpful, which are helping us to get from the bottom, bottom world, that might be extremely motivating.

Time. Time might also be extremely demotivating. Ok? if we set for the team, for ourselves a goal to be met in a very short time which is not enough it's going to be a disaster. Of course, we all know that deadlines are motivating because just before the deadline suddenly everyone is so full of energy, is starting a lot of work. We sometimes work over hours, which is completely insane and we look to a lot of things because there is this deadline. But, if we put the deadline absolutely crazy, in an insane way, which is going to be extremely ridiculous, it's going to be demotivating because from the very beginning we know that we will fail because there is not physical chance to meet the deadline.

Meetings. Meetings can be demotivating as well. In many cases, I'm not a huge fan of meetings. Whenever I'm on a meeting that's longer than 30 minutes I get lost, I lose my attention, I feel demotivated and I just want to escape which isn't always possible unfortunately. When I was talking with my team members they said they also are not huge fans of meetings. That's why I like the idea of no meeting culture, just sharing the information as it is without the need of meetings because meetings are stressful, might be demotivating especially without agenda, without action items. It's very tough.

Sharing ideas. So, motivating each other by inspiring each other. That's a great way of motivating ourselves and motivating other people, right? So, if you have some great idea, sometimes it is better to share it, to discuss it, to talk about it because you can inspire a lot of people. I didn't know that this kind of activity may have this kind of a power. So, whenever I discuss something with someone and he shares some ideas with me, wow, it's like a magic potion of positive energy, right? Because while motivated we're like a very, very. Do you remember this advertisement of Duracell, this rabbit that was running very fast on the right batteries? Motivation can be the same for us. It might be a great energy, energy for us whereas sometimes we feel that work is not the right place. So, we feel that, because of gazillion of devices we are still at work, we don't have time to recharge our batteries and we are getting less and less motivated whereas work should be something like this, it should be a magical machinery that is converting regular people into superheroes because of the motivation part, because of the inspirational part. And they believe that this is possible because, I don't know, maybe I was just lucky, but in many cases, the places I had the pleasure to work because of the people I was working with. It was places like this, right?

Gitte and Lilian are talking about changes. The change is not easy, the change can also be demotivating especially if we don't know the goal, if we don't understand it, if we don't feel secure, if there are so many obstacles. So, we need to be very, very careful in case of a change not to make it demotivating. Sometimes we are in the trap of motivation. So, we want to control the environment, we want to control our motivation beast to make sure that it goes in the right direction. As a regular beast, beasts are not easily tamed. If we try to control it, hm, it may just go back to sleep, that's one of the situations. It may try to prove us that it is much stronger that we are. So, sometimes it is better, at least in my case it was just better to leave it all alone so that it can go in a natural way.

So, one of the most surprising findings about motivation I found was that achieving success might be extremely demotivating because what's the next step? I achieve this goal, I put a lot of effort into it. So we were working on a great project. It was very tough, it was challenging, a lot of obstacles. We did it. Ok? And what's next? Then, just after achieving the success there is a huge danger that you will fall into

a demotivating area that is going to be very dangerous area. In some cases we even do everything not to achieve the goal just to not get to the state which is extremely difficult.

So, when I was doing my internal analysis about what motivates me, about what is demotivating me, how I can help other people to motivate them, how I can inspire them to be better, to be more happy, to be more smiled people, then I felt that's, wow, actually Agile, Agile manifesto. This is a very nice guidance how to do it. So, the rule about the individuals and interactions, from my perspective, that's the key. So, when the team is focused on individuals, when the team is focused on interactions, when the team is self organized, when it is independent, when there is this management by participation, when they can make the decisions, when they are in charge and they are managing the environment and they can make the project decisions, then it's getting better. I'm not saying that then the team is 100% motivated, we are like working 200% of the norms, and, you know, we are finishing the projects in half time that we are supposed to do it, but the environment is so much better, it's so much easier to inspire them. They feel self inspired because they can make choices, they feel in charge of the whole environment. So, actually if you are looking for an interesting inspiration for the motivation maybe it's worth to read the manifesto once again. Maybe it's worth to look at it once again and look at the people. I'm talking from a perspective of a very people focused person so it might not apply to you. But, still, I believe that in the manifesto you may find a lot of good motivation recipes.

So, for the very end, my advice because that was something that I was struggling with, when you're working on your motivation, when you're trying to motivate other people, don't lose things that are important for you. It's just a cupboard of things which were lost and found and sometimes when I was focusing too much on motivating myself I was losing the things that were the most important for me, in my life. So, I just had to stop. And I had to rethink it and then I've got the balance it was much better and I felt much more motivated. So, what I wish you, what I hope it will be take away for you from this half an hour is to listen to yourself. Don't look at the results of the research, don't look at the scientific book and try to do everything by the book because you are unique, you are different. There may be so many other things that are motivating you. It's goo d to look for the inspiration. It's good to discuss other people and their cases because maybe it's going to be inspiring for you. But don't feel bad because something is not working for you. And try to be egoistic, try to look at many things from your perspective because if you're going to be happy, if you're going to be motivated, then you can share it with other people.

So, thank you so much, especially huge thank for the volunteers. It was a pleasure.

Torbj□□rn Gyllebring: Sustainable Pace: lessons learned from running 100 miles

Torbjorn Gyllebring: It hurts up to a point and then it doesn't get any worse. Those are the words of Ann Parson after finishing an extremely ruling race. And these are also words that I sort of carry with me through many of the things that I find valuable and meaningful in my life. Running is one of them, standing on this stage, it hurts to some point but then it doesn't get any worse.

So this is a talk about sustainable pace, it's my lesson learnt from running 100 miles and this is a talk that haunts me. It's one of the first sessions I ever did. It's the one that has then sort of been haunting me because everyone that saw it the first time said ''That talk was really good.'' And I was like ''Thank you very much but I've been doing some other stuff after that.'' One thing to remember, one thing that I concluded, for its rehash is that running 100 miles, sorry, that's definitely not a sensible thing to do.

That's the first lesson. This is my lesson from jogging 100 miles! I'm not an elite athlete. I'm just a regular guy and this is approximately how this sport works. 90% of this game is mental, the rest is, the other half is physically. And just to get a feel of the room. Could everyone of you stand up if you've ever run, this should be approximately everyone. There's a few people still sitting. Have you never ever run? Stand up! Ok. So keep standing if your longest run is 1 km or more. Ok. So, if you've run further then 5 km keep on standing. 10 km? Half a marathon, 21 km, that's insane. Ok. So who's reached the peak of human potential, the marathon, 42 km? Your legs will fall off. More than 50 km? More than 50 miles, that's 80 km. I have a few people still standing in the back. I wonder what they did.

So, if you close your eyes and you imagine that tomorrow is race day and then you answer a few questions for me, in your eye ''What do you pack? What stuff do you bring with you? How did you prepare? What have you been doing up to this point? Once at the starting line, what's your strategy?'' This talk isn't a metaphor. This is a talk about running. Some people manage to make connections to projects out of it. Those are completely random and.. This is a running talk. Metaphors are brilliant. They let us discuss the things that we don't understand in terms of something we understand even less of. Keep that in mind.

So, I'm Tobbe and that's me trying to run. This talk will contain two unicorns. That's one of them. It will contain two mentions of SAFe.

Audience: Boooooo!

Tobbe: This is the first one. Some people already know what to do. If you want to e-mail me or reach out you'll find me on twitter as drunkcod.com or you can just send an e-mail to shout@drunkcod.com and it will end up in my inbox and I will eventually answer it.

So this is actually about extreme jogging. It all started out with me sort of thinking, well it correlated with me actually getting a promotion in a leadership role and that correlated with me starting to jog, to run. The thing is that I run when I was frustrated and that actually ended up being quite a lot. So I ran a lot.I got this idea that probably there's something about running. I could probably use two principles since I am an extreme programmer. First edition: protocol. I should be able to use that to figure out how to run. It should be easy enough. And there's a few different ways to make running hard and I am not deeply goof at either one of them.

One of them is going fast. Trying to go fast is, it hurts, it hurts a lot and some people are extremely, extremely good at it. If you want someone, if you want an example of someone who's really good at going fast take Usain Bolt. Or we can go far, we can go further and do something bigger and more grand

than anyone has ever done before. At that end of the spectrum you'll find people like Johnny Scorus that has the world record and the shortest time for running 1000 miles or running across America or doing a lot of insane things. So you can try to do things quickly or we can try to them over extended periods of time. Both of those bring with them their own challenges. I heard of someone wanting data. Being data driven is nice and this is actually data collected from a few of my projects over the span of a few years. And what we can see here is that the velocity in my running projects sort of tends to go down the bigger the project is. Somewhere we see that we seem to hit fatigue. Something is slowing us down. The dynamics change.

So, running 5K. The 5k's are a brilliant starting race but this is already long distance running. 5k is far. But what do you actually need to run 5k's? Well, you need to get up from your sofa, at first and then buy a pair of running shoes. And that's approximately it. And you can probably do it in clogs, too, if you want to. There's a lot to be said about shoes. These are a few of my favorites, but essentially any pair will do. You go out and you're running, that's all you need to do. You go out and you're running. You need to train for it but you don't actually need a lot of stuff. The dynamics are very easy. You show up, you lace up your shoes, you go running. It's an easy project.

So, now what happens when you sort of say "Well, that was easy. Let's go for 10k run." Well, you need to train a bit, of course. You need to add some distance. On top of your shoes, all of the sudden, you are not only running, you're sort of eating. You're eating funky stuff like this is fructose. You might start drinking water. Drinking water is fine at 5k if it is too hot but most of the time you don't need it. You start getting all this extra logistics just for the particular reason that you just went a little bit further.

And then it starts getting weird, the half-marathon. The half-marathon, 21 Kilometers. This is sort of, this is where insanity starts. This is where people are going "Oh, you should probably be careful about that. Will your knees actually carry through?" And as a good friend of mine said that actually got me stuck on running to begin with, he's like "Well, Tobbe, you need to understand that half-marathon is a completely different sport from the 10k." And I'm like "What?" I'll just be doing the same thing twice as much as I did the prior time. It can't be that bad, can it? Anyone ever heard that? Anyone ever thought it? I fought with the 5k's. That was easy. It was a hell of a running. And then 10k's, double. Sip some water, it's ok.

All of the sudden sports drinks start entering the picture. You go out for decently long runs and you sort of, you need all this crazy equipment. What's this? Socks? Socks all of the sudden start to matter a lot. I didn't think about my socks when I stood at the start of a 10k run. I'm like "Yes, I've got my shoes! I'm going that direction." Socks have actually become more important. More gear is required. We need to take more care. Now all of these sort of, it takes some extra preparation. And then, of course, the challenge of a lifetime, the marathon. The ultimate thing to do. It's this and then you die. Because that's how we actually came up with the bloody idea and of course marathoning is from the start a male sport. Because we are the only ones that were dumb enough to actually run ourselves to death. Anyhow, marathons. That's two halves." That's a different sport", my friend said. Oh, yes, I can see the reasoning.

So when you come up to the marathon all of the sudden you need to add, what's this, energy gels. Why would I need that? Well, you do or you might not. The thing is that there would be some arguments regarding this some of the stuff that I said. Well, if I was only going for a low carb diet I wouldn't need those bloody gels. It might be true, it might not. I found them very helpful but I'm caring extra stuff again. It's one more thing to think about, again. And the texture of the race is starting to change. Anyone ever hit the wall? So, this doesn't actually make the experience sort of.. This is more like a hit. Boom! It's horrible. It's called bunking or hitting the wall. Essentially you're cruising alone fine in the middle of your marathon and all of the sudden the wheels just fall off. And that happens for a very particular reason. It happens due to you being a system, a system in fine balance but with finite resources and constraints that you need to adapt to. So within your body you store about 200 calories worth, 200 calories worth of glycogen, blood sugar, most of it in your liver. That's enough to sustain you through the half marathon and it's

approximately enough to sustain you for 90 minutes of sustained, hard effort. And that's approximately the 30k mark in a marathon. And that is when this happens. So, essentially, once you've reached that point you are out of blood sugar and everything becomes hard because you're burning fat instead. And that's what the gels are there to help you with, just to top off that limited energy supply.

It hurts. It hurts bad, but if you want to see a battle field go and check a marathon at the 30k mark. And I was dumb enough to run a marathon and due to a bet ended up running one more. And after running two marathons I was thinking "Wow, I have got this running business down. It's good." And an old friend and classmate said, he'd always been running and said "What's your next challenge? How will you top the marathon?" "I don't know. I don't have a clue."And he said "Well, there's this thing called ultra racing or ultra running." The ultra starts where the marathon ends. So any distance beyond the marathon is, technically speaking, an ultra. And commonly ultra is to start with is like 50k. And what he did, he found an interesting race, an interesting challenge and I was actually called the Tobbe extreme challenge. TEC100. So it's a 100 mile run. That's close to 3..That's close to 4 marathons together. And he sort of got me to sign up. And I was like "We'll try running for a bit." We didn't have a clue on how to actually do this, but we tried.

And this is essentially the first thing you find if you're trying to figure out what the hell is this. Any idiot can run a marathon. It takes a special kind of idiot to run an ultra marathon. And I'm a very special kind of idiot or I'm a bit like a bumblebee but I didn't actually know that I couldn't do it. So I just went out and did. One of the key points that..If you are trained in an ultra you'll sort of figure a few things out. For example almost all of your trainings run, training runs are very much at, at a much higher pace than your regular training. You're racing slower than your training. That's complete opposite of everything else.

So, 50 miles. 50 miles is an interesting distance, close to 2 marathons. So if the wheel falls off at 30k's what the hell did you do for the remainder 50? That's a good question, isn't it? So, what you do is start eating and you slow down because you need to give yourself time to prepare and to actually, to sustain yourself. You can't get too far into debt. So if the wheels fall off, hitting the wall is you running into technical debt within your body. So, running 50k's you start drinking stuff like this. This isn't the best sports drink ever. It tastes horrible but you get used to it and then it's all Stockholm syndrome from there. The thing is that it contains both fat and proteins and complex carbohydrates. It's, it's actually, it's almost like Soylent. It's brilliant. And you're starting eating meatballs on the run. Training for an ultra marathon entails training to eat while running. That's a really, really strange experience.

The hundred mile. The hundred mile race. So I saw a few interesting things in that small group that had set out to tackle this race and one of them was that you need to be committed. If you're tackling something like this, if you're tackling even the half marathon, even the 10k, there's a really good predictor if you're going to make it or not. And it's simply this "If you think you are not going to finish, you won't finish." If your resolution isn't complete, if you haven't set your mind to it, you won't finish. When you invite doubt into your mind, that's when you lose the ultra. And all of the sudden you also need to carry a lot of more stuff. Running an ultra marathon, running 100 miles, the first time I did, I was on my feet running for 21 hours, 40 minutes straight. Think about that. I needed to keep on moving for almost a day and I also needed to carry a headlamp because, of course, you'll be running through the night. You don't stop. You need an emergency blanket. You need a ton of stuff like that.

The pleasant part about ultras, having run them one more time, is that, there were two more times, this particular race, is that you can do stuff like set a PR with 3 hours. SO this year I ran the same race in 16 hours, 25 minutes. That's approximately an hour per 10k 16 times in a row. The thing is that I have been running road races, and the particular 100 miles that I ran was quite flat. So there's, other ways you can change this style of running. You can go "Uh, hill races!" I realized I'm terribly crap at that. I can run 100 miles but show me a hill and I'll die. This is a good advice for the day "You can never run a hill too hard, you'll collapse way before you hurt it."

We can go running on trails instead. You can use nature, you can go out and discover. And that also adds flavors. It requires technique, but it is a completely different experience from running those 100 miles on asphalt or easy trails. Just go out in the forest and run and be connected to nature.

So, that's my story, that sort of..This is 100 miles, this is sort of how the races relate. This is how the dynamics change. These are ways to make it hard. But what did I learn except for the fact that you do not run 100 miles, you jog it?

The power of habit. The thing is that if you want to complete that race, no matter what the distance is, no matter how big your project is, you want to find leading indicators, you want to find the stuff that helps you achieve what you want to achieve. I need you to focus on do's. I need you to build good habits. I need you to learn the habits that build you up in order for you to actually be able to race. Habits is essentially… will you win it, will you lose it. It's very much the power of consistency over heroics. So most people when they sign up to a race, especially if they sign up to the half marathon or marathon, what they do is that they postpone training for a bit and postpone it a bit more and then it's race week. And then what do you do? You go out for this one monster run to prepare yourself. Anyone ever done that? Ok. I have done that a few times. It doesn't actually work.

The heroics, heroics will only set you back, will brake you down. You need to be consistent, you need to always be running. You don't need a whole lot of training. I train as much as about most people claim to train. I spend 45 hours a week exercising. That's not a whole lot. But I do it every week. Sticking with it, practicing. You need to always be practicing if you want to be, if you want to perform. That's the thing. You need to get into that habit. You need to get into a mind set of always, always training, always practicing. Train on training. So, the best mindset is essentially adopting one "I'll be training on training." Can I get 3 or 4 runs in this week? Don't actually..It doesn't matter if they're long or short, if they're hard or easy. You just want to build that habit of getting those runs in.

```
And what would cause anyone to not train? Why would anyone not want to exercise? So, th\
is is essentially true. I've got a good friend that said "Well, you know, training, tha\
t's something that you do between eating and injury." He was injured all the time and h\
e didn't actually progress. If you accumulate so much damage to your body or anything e\
lse that breaks down, you won't get to the starting line. And if you don't get to the s\
tart, you probably won't get to the finish. Training ain't racing. A lot of people sort\
 of conflict these two concepts. When you train, you train. When you race, you race. De\
livery and practice are different things. During training you can do a lot of stuff tha\
t you wouldn't be doing during the race. You want to test your racing capabilities. You\
 want to be close to performance as much as you can. But you also want to pay extra att\
ention and be kind to yourself. In the race? That's a different story.
```

Essentially, anything you need to do to deliver a sizable project like running 100 miles is you need to train, we talked about that, you need to eat, eat well and you need to sleep. Most people forget one of those. They forget to sleep. If you're not sleeping, you're not recovering. If you're not recovering, you're not improving. If you don't give yourself space to internalize and to grow, to recuperate, you won't be delivering, you won't be performing. And all the effort you put in to your training will be wasted. It doesn't matter how hard you train, unless you sleep. And don't forget, you need to eat, train, sleep. Repeat. Habit, consistency, these are the things that will build you up to reach your goals.

So, you learn by racing, but races aren't for learning. When you're out there, hunting your dreams, the last thing you want to do is start experimenting. You come up to an aid station and you say "Uh, new, shiny sports drink!" Don't take it. If you see anything at that aid station that's new and shiny, a new goody,

don't take it. Go with what you know. I think there's a lesson in that. You'll learn a lot, you'll learn a lot about what works, what doesn't work but do not experiment when you are trying to perform at your best.

Behaviors give results. That's the thing. You focus on the behaviors that are the requirement to get the new results you want. You focus on leading indicators. You focus on getting those run is, runs in. You focus on the paces. You focus on how your body is adjusting to training load. You get feedback so you get leading indicators in the moment. Get a pulse watch, get anything that gives you by feedback, so you can air on every single moment while you're training. Get feedback on how you're doing. The results, the outcomes are simply the result of consistently training. So focus on behaviors. Always, always, always focus on the behaviors. The results will come, they're still static but if you focus on the results, you'll start to go for heroics. And if you go for heroics, you'll probably, as someone said, you'll lose faith in your running. If you do like, was it Monika? I'm getting back to running, I'm going to run 10k. So I haven't run in half a year and you feel like shit. And you won't. You'll overextend yourself. Focus on the behaviors.

You should never fear moving slowly. Relentless forward progress. That's how you improve. It's not about getting everything done, it's not about improving every single aspect of your running life. It's about taking one step at a time. It's about one small improvement here, one small improvement there. Constant Kaizen based on feedback, consistency, habit.

And you need to know why you are racing. You need to know why you are doing this. You need to know your goal. Is it simply to finish? Is it simply to reach the finish line? Or is it to give it your all? Because this will, is in your strategy and your strategy needs to be consistent from the first step with your goal. If you just want to finish, take it slow. If you want to give it your all, well, then you will also burn out. Then you will at times fail. So which one is it? Both are valuable, but you need to know on the starting line why you are there. To race well is to race evenly. So the best results in a distance, or in a race will be had if you get close to neutral split. That is if you ran the first half at approximately the same pace as you ran the second half. That's sustainable pace for you. You need to figure out what the pace, what pace exactly to start at and then just to keep at it. You often say that it's not the fastest runner that wins, it's the one that slows down the least. Keeping at it. And this goes for, but in order to that you of course need to know what race you're running. And often we think that running uphill is a battle, is a struggle. We can see the effort it takes. But running downhill requires technique. It's where you can gain massively on everyone else. So learn proper technique for running downhill and figure out what downhills are.

Cadence. Cadence is one of the most important aspects. And this is a magic number, 180. That's approximately the number of steps you want to take every single minute. You want to drum it into your body so you naturally run at that cadence. SAFe.

Audience: Booooooo!

Tobbe: See, SAFe like programs or SAFe training programs. There's tons of programs out there, there's tons of framework, frameworks, that will sort of, how to structure your training. And they're all good and people earn a lot of money creating them and they're useful but they should be used as inspirations. You should take the parts that you like and discover the rest of them. It needs to be close to you. You need to listen to your body. You need to watch your results. You need to build it for you. Use them as inspiration but don't rely on them.

So this is my question to you : What is your next race? Do you know how far it is? Do you know what your goal is because if you don't it's not very likely that you'll be able to pace yourself. It's not likely that you'll be able to prepare, to figure out what gear you actually need. It's not likely that you'll finish or that you'll race well. So please, before going out for that next run, figure out what your purpose is. And if you want to read a very honest story about one man's struggle, read this book "Seeking Ultra". It's written by one of the most humble and honest guys I know. It's written by Paul and it's essentially, he's going from a couch potato to an ultra marathon in 6 months. It will give you tons of practical tips along the way. Read

it. It's brilliant. And go run.

One more thing. I promised you a unicorn. So always remember that "Last is just the slowest winner." Thank you very much.

Andrea Provaglio: The Expectation

Andrea: Hello everyone! How are you, guys? The conference is slowing down. You seem relaxed from here. It's good to see you again. It's good to see old friends. So, thank you, Paul, for the nice introduction. For those of you who don't know me, well, I work independent as a consultant. I work with organizations who want to change for some reason, so they call me in because they want some kind of change which is a very mysterious work sometimes. It could be a very large organizations sometimes such as the United Nations or very small companies. But, you know, they want this, they want this thing which is called change.

So, with change comes expectations and this is what I would like to talk about today. So, apparently, according to William Shakespeare this idea that expectation fails is not new to technology. He's saying that very often "expectation fails and most often where they most promise." So, what do we mean by expectation? So, let's try to define expectations at least in this context, of course. So I'm not trying to come up with a universal definition of an expectation. But to do this, I will need Pawel. Where is Pawel? Back there. Pawel, I need you. You can stay there. I have a question for you. Let's say you go to the bar and you ask for a beer, something you never do, right? Never do that. And they get you a very good beer, perfect temperature in the right glass. Would you be happy?

Pawel: Yes.

Andrea : You would. So, I want a beer, I get a beer, I'm happy. Good. But let's say that you ask for a beer and you get a cappuccino. Would you be happy?

Pawel: No.

Andrea: No, that's what I thought. Thank you, Pawel. So, sometimes it's very easy to understand what an expectation is. I want something, I get something, I'm happy or I'm not happy. Ok? Sometimes, although, is trickier, like in this case. Our product is supposed to be delivered by the end of May because our market window ends. After that date, the product won't be as valuable as it could be. Ok? So we deliver by May 29th and we're happy. The client is happy. The stake holders are happy. Users are happy. The team is happy. Or, we want that product delivery, product is delivered by May 29th except that one key feature is missing. Happy? Nope. Probably not, probably not. So, what happened there? Was it a communication problem, something that we forgot to talk about? Was some kind of assumption that was not clear enough? So, what happened? So, sometimes these expectations that fail are not so simple as "I want a beer, I get a beer, I'm happy". So, where is the complexity? I'm trying to come up with some kind of simple model for expectation. The idea is very simple. You make a decision, like I want a beer, there is an outcome, there is some kind of feedback. Ok. I wanted a beer, I get a beer, I'm happy. Or, maybe not. And then we take another decision based on that feedback like "Oh, that was good. Let me, Let's get another one." Ok? And then "That was very good. Let's get another one.", you know. Up to a point where it becomes hard to articulate the sentence "I want beer." And the loop ends. Ok?

That was an example, but the question is the outcome you cannot control. The feedback is objective. The question is how about the decision that you're making? How about the decision? Which are the rules by which you make decisions? Because the decisions you take will determine the outcome and will determine if your expectation failed or is met. In a project, in a software project we take hundreds, thousands of decisions throughout the lifetime of a project. So, understanding what happens when it comes to making decisions it's interesting to me.

It turns out that what we have been talking about was just what we call "the tip of the iceberg". Let me give you some theory first and then we'll do something more practical. So, if we talk about organizational

culture, Mr. Edgar Schein is suggesting that on the surface what we see are artifacts and an artifact could be the language we use in the teams or the decisions, of course, that we make, could be the physical layout, could be the processes. Basically everything that we can feel or observe, so those are artifacts. But if you look below the surface an organizational culture, you will find values and beliefs. Something like ideologies, goals, aspirations. Something we believe in. But if you look even deeper, and that's the most interesting part, you will find basic assumptions, which are something we take for granted. And of course we alter many of those. But, many of those, many of these basic assumptions, as you can read, are unconscious.

You can tell there's something strange in the complexity of an organization. What I mean is you can see that organizations are complex animals because just if you look at the artifacts, you will notice that some things are kind of out of joint. Like, for example, you will hare statement like top management or executive saying "We values team work here." At the same time you observe the reward system in the organization is clearly designed to promote competition among individuals. So, how can you meet this expectation of having good team work if you promote individual competition, for example? This is an expectation that is doomed to fail. Or, we want to do Agile and then you talk with line managers or project managers and it turns out that projects are expected to be with fixed time and scope. But we want to do Agile, yes? Interesting, right? Or you talk with the teams, technical teams and they tell you "Yeah, we would like to have some time, to have a sustainable pace like Tobbe was saying, but you know, we don't have time for that. So, we don't have time to self-improve." At the same time that team keeps on taking on more and more work, even though they could refuse. So, you see this kind of dissonances in the artifacts.

Now, another gentleman, Mr. Peter Senge introduced the concept of mental models. Basically, the decision rules I was talking about, remember the surface there, the outcome, decision, feedback, that part is just the part "I want a beer, I get a beer, I'm happy." But, what we're talking about it which are the rules by which we make decisions, we decide that we want another beer, for example. So, there are rules behind that and those rules most frequently are driven by mental models. So, what's a mental model? It's something that we need as humans to make sense of the world. It's absolutely subjective. It's basically a set of stories, impressions, assumptions that you have about the world. And using these you filter the complexity of the world and you try to simplify it and to live with it. So, mental models. Mental models are most of the time completely unconscious. But they could be brought to the surface.

And another gentleman, Chris Argyris, suggested that there is a way to learn about how we think which he called double loop learning. And Marcin has been talking wonderfully about learning yesterday. So, in double loop learning we don't just learn that if we order a beer and we get a beer, we are happy, so we can have another one. But we also learn, we get some feedback about, ok, what is the mental model that I'm using, that is active inside of me so that I do want another beer? So, this kind of understanding in an individual, in a team or in organization bringing to the surface mental models will most likely increase the probability that the results meet the expectation because at this point you are aware of what you want to expect which is different form expecting something.

So, if we put all together, and we also add a little spice which is called shared vision, ok, you see that the complexity of having expectations and creating eco-systems in which the probability of meeting expectations raises, it's not so simple. There is much more than meets the eye. That's what I'm trying to say. It's not just the surface. Again, generally, our concept of expectation is "I want this, I got it." And that's what expectation is. But there is much, much more.

So, what I would like to talk you about is the decision rules, the mental models, feedback, the double loop learning. Basically the idea is how can you replace a mental model in an organization or a team, or whatever, it's a social system, in such a way that benefits that system and probably in a way the system, the organization decides to do? So, how do you change that back? It's a metaphor.

I'd like to try an experiment with you, guys, and surprisingly it does not involve standing up. It requires

talking. Ok. It requires talking. But, or course, you are totally free to accept it or respectfully decline my proposal. So, if you don't like to talk to people around you, that's perfectly fine. But anyway, some people will talk and will be making some noise. So, at some point we'll need to bring the group together again because we have a limited amount of time. So, do you know the thing about someone raising his hand or her hand and you do the same at the same time? You finish your sentence and then you stop. You know it? Ok, let's try this. So, when you see me doing this, you stop talking and you raise your hand. Ok, make some noise, noise, noise, noise. Now. Bla, bla, bla, more, more. Thank you very much. That's what we are going to do to regroup because we don't have much time. So, what I would like to talk you about, what I would like you, guys, to talk about; the guy at the table there maybe you can share this with the other people around you. The people here in the front, do whatever you want. Find someone to talk to if you'd like to. Ok.

Remember I mentioned these dissonant artifacts like "we don't have time to improve but we are taking on more work." Ok? Has, is any of you familiar with this? Has it ever happen to you? Raise your hand. A team saying "we don't have team, we don't have time but keep taking on more work? Ok, this is not that team, ok? We're not talking about you, we're talking about some fictional team. But what I would like you to do is spend 3 minutes, and I will be timing this, to try to make hypothesis about what could be going on there. Why would a team say that they don't have time to improve and, in fact they don't, at the same time they take on more work? Ok? So, I will give you 3 minutes to think about this and share your ideas with the group, at your table or whatever and then we'll regroup. Ok? Go! Have fun.

Thank you. Thank you everyone, thank you very much. So, what did you find? Can I have some, someone would like to share? In your group what was the hypothesis? I need one. I'm sorry?

Audience: Inaudible.

Andrea: Pressure. Where does the pressure come from?

Audience: Inaudible.

Andrea: Mindset. So, deadline driven, deadline driven development. Ok, that's one possibility. Any other, somebody? Yes.

Audience: Inaudible.

Andrea: Ok, it's an excuse for not doing something. Ok. Thank you. Oh, Liz. I'm af, I'm sorry I can't hear you from

Liz: So, Olaf and I actually have quite a bit time because we're both independent. We realized that we want more work and when it's difficult to find work it makes us demotivated and we don't have the energy to improve the ways in which we go out and get work which is an interesting. There you go.

Andrea: Ok, thank you. Thank you. So, as you can see we have very different opinions. The reason is when you look at artifacts they are easy to spot but they are very hard to make sense of because there is so much below the surface. Could be an excuse, could be management, deadline driven development, bla,bla, bla. It could be many things.

Now, I'm going to ask you for the next step. And the next step is, I'm going to give you less than 2 minutes probably and I would like to do something, a little twist, a little twist. The question is "Which you think was your mental model which made you think that it could be an excuse or it could be deadline driven?" And I'm sure there many different opinions here. So the question is, basically I'm turning the question not to an hypothetical team, I'm turning the question to you, as an individual. Ok? And the question is " What do you think is your own mental model so that you thought that these guys are using that as an excuse or the environment is not supporting them and so on and so on? " I will give you one minute to think about this. Just one minute. You can do this in silence, you don't need to talk, if you don't want to. But think

about that. What was your own mental model, for you as a person? Any flash of light? Somebody wants to share? You don't have to, you can. Ok, that's fine. I think. Time is running out so we need to move on.

This was an example of what we call reflection and Olaf has been talking about reflection quite a lot yesterday, about self-reflection. The idea is that you review not only the artifacts, statements and behavior, but you're, but you also start asking questions about why am I interpreting reality in such a way. So, this calls for self-reflection. And it can be done by an individual or it can be done in a team. How do you do that in a team? By talking. But also visualizing things or having specific data. Now, the thing is that if you want to do. I don't know if you noticed the change that was in you when I asked you "Could you please think about someone else?" Someone who has a problem and let's try to analyze their problem. And then I asked you "Ok. What made you think what you thought?" Did you notice the difference? Did you notice how much defensive you became, how much reluctant you are about sharing your mental model, how much vulnerable you feel when it comes to talking about this is the way I perceive reality? Ok? Now, if you do this in a team, you can expect resistance and projection. Projection is when you have this kind of parallel conversations. So, somebody is talking about "Ok. We, why do you keep taking on more and more work?" and projection is "Oh, it's not us. It's the manger that keeps give us more work." Ok? "Why are you not self-improving?", "Oh, because we, the organization doesn't have time." Projection is when you try to deflect the question outside of you. It's a defense mechanism.

Gitte and Lilian have been talking quite a lot about how change can be scary. But if you think creates this kind of environment, and if you can use factual data that becomes much more easier. So, what I'm trying to say is, the little experiment that we did here in 5 minutes is bases on the same principles. Try to analyze something, come up with objective data or visualize something and then ask people to give their own interpretation not about what they are looking at but about why they interpret that in their way. And share that. That creates much more alignment. So, the first thing I was telling you, I've been telling you is, yes, you can bring to the surface mental models which in an organization is an extremely important thing to do because it creates more alignment. And if people are aligned, the level of unmet expectations will probably decrease.

The other thing you can do is not just to try to bring that to the surface using data, you can also try to provide alternatives. You can tell stories from the world, like these 2. Let's say that one mental model is that financial payoffs are predictable or should be predictable like budgets. Ok? We want to spend this and we want to get that in return. Ok? This kind of thing.

Or, things will not change. Ok, we'll make a plan and things will be what we expect to. Ok? I have a couple of examples that sometimes I use. Let's think for example about the music industry. The music industry has dramatically changed their business model because up to 10 years ago we used to go to music stores and buy physical records, which means those things needed to be recorded in professional studios. We needed to print the vinyl's, to print the album, ship this thing using trucks, and so on, and so on. And these are all costs, including the professional recording. But now, that good, this music that we used to buy has been, as we say, ephemeralized. It is intangible now. We don't need a physical medium to ship the music. We can just download it. So, not only we can buy music but there is an entire online music store. Previously, I used to go to the store to buy music, now it's the store that's coming to me whenever I want, wherever I am. It's here. I have a store here. I can pay with virtual money to buy a virtual good. There's a virtual sales assistant, there's a virtual catalogue that I can browse. The entire store is here because it's no longer physical. And that has dramatically changed the ratio between costs and profits, right? It's much cheaper to produce music now, much, much cheaper to produce music and to distribute music. The business model has changed.

And since it's an Agile and Lean conference, if you know about Lean and this Lean product development approach, probably you know there's a statement by Reinertsen and he's saying that "If we make much more money from success than we make from failure, then we can risk more basically." We don't need

to have this 50% certainty that we will get some kind of financial return which is exactly what the music industry, in general, the publishing industry, movies, TV, books, is doing. What they're doing is say "Ok. It costs so little now to produce something and distribute it that I may risk to produce 100 records in one year expecting returns on just 5 of them. If I have full success on only 5 of these records, I can risk to be a failure for 95% because the costs are very low and the profits are still high." It's a different business model, but this is an example not of a business model, but of a mental model. And you can bring this, you can talk about this with your management for example. You can say "Well, you know, these guys, the publishing industry they've changed their way to do business." "Hey, dude, we are a software company. Who is more intangible than us? Who is selling more intangible products than ourselves?" So, why should we learn from them? Why shouldn't we change our mental model and business model? So, what I'm saying is, you can give evidence to the mental models and discuss them, you know, using inquiries and so on and so on. You can visualize or you can give alternatives. You can take stories from the world and say "You know, some people are doing something different." Like in this case.

Uncertainty. In Geneva, the CERN, they are doing anti-matter experiments. In 2012 they needed to slow down their schedule because in Thailand rained a lot, because Thailand was flooded. And the reason is that Thailand is the largest producers, producer of hard disks on the planet. And the guys and CERN in Geneva they ran out of memory space. They didn't know where to store those peta bites that they get with every single experiment. So, if the world, if the society we live in is so interconnected maybe this idea, things will not change, things are predictable you know, maybe we can revise that.

Or, do you know these guys? The brain of mobile operators, ok? These are the founders of WhatsApp. A few years ago mobile operators invested billions in expanding their infrastructure because they were planning to make money from text messaging. And then, these guys show up. And then, boom. Who's text messaging? Who's using WhatsApp in this room? How many of you are using WhatsApp? Ok? Well, about 20%. But, you know, 20% less messaging revenues for a mobile operator, well, it's a big loss. So, it's very unpredictable.

So, what's the mental model behind that? This idea that you can predict things, you can stick to the plan. Maybe there are alternatives like these guys, Gojko, Chris Matts and Olav. I especially like Olav's picture here. They are providing alternatives to this idea that things can be predicted. You can use impact mapping if you want to run a set of fail safe experiments. Or you can use real options. I'm not going to explain what these are. What I'm saying in this context is you can bring these as examples to people as different mental models. It's not just a business model, it's the mindset that is different.

So, I think we're done. And what I've been trying to convey in these 30 minutes is that expectations are inherent in life as Shakespeare pointed out, are inherent in what we do especially because in a software project or in an organization we take hundreds and thousands of decisions in some places every day. So, if there is a decision, there must be an outcome. When the outcome doesn't match the expectation that was, you know, behind this decision, we are not happy. But the question is and that's the thing that's below the surface "How do we make decisions, especially in a social complex system like an organization?" Are these decisions aligned in the same mental models? Are these decisions, for example, reasonable or realistic? Kasia was talking this morning about say no, learn to say no. If an expectation is totally unrealistic say no and provide alternatives as she was saying.

So, what you can do is you can develop your sensibility, your sensitivity to artifacts. Start to observe when artifacts do not match. Ok? And start asking questions then again using objective data, using self-reflection, inquiry. And the other good thing is, make sure that everybody knows which is the, which are the rules by which we make decisions. We could say for example "You know, I strongly believe we have surfaced that in our team, we are really scared about job security. And that's the reason why we take on more and more work and we don't improve. But we have been able to surface that." So, at that point your decisions become more sensible and probably your expectations will be met. I hope so, but that's my wish

to you.

So, I'm done. Thank you very much for your attention. That's me, my contact information. I'd like you to get it to stay in touch. You can find me on LinkedIn, you can find me on Twitter, you can find me on Slideshare if you'd like the slides. These slides will get applauded tomorrow night as I get home. Ok? So, thank you very much.

Darci Ducher: Expertise is Not Enough

Darci Dutcher: Thank you for that introduction. I have to say I don't know where Russ is in the room but I will comment that even as an extrovert, standing in front of 200 people, a little bit intimidating. So, bear with me. I'm used to doing this in quite smaller venues because user experience generally is kind of a side topic at Agile conferences, that at Agile conference you act a bit as a side topic. So, it's quite exciting to be sort of in main stage today.

So, what I'm going to do, is much like other speakers today, I'm going to make you stand up. If you think that you are good at your job, stand up. Ok, if you think you are good at any part of your job, stand up. Ok. Most people are standing. I'm a little bit worried about the ones who aren't because hopefully there's something at your job that you're good at. We can address that after the session. The next question that I have for you "If you think that being an expert and having the knowledge that you have right now is enough to continue to be good at your job, stay standing." Sorry, did I make that confusing? Ok. So, this is interesting. So, most people think that with the knowledge. The rest of you can sit down, sorry. This is interesting. So, most people, ups, most people in the room think that the knowledge that they have today and the expertise that they have is not enough to do their job well in the future.

So, I'm going to confess something to you. I'm fairly good at my job. I've been doing it for a number of years and not I'm going to tell you for how many years I've been doing it but it's a number of years that I've been doing my job. And I'm pretty good at it. I've designed airplane cockpits, I've designed healthcare software, I've designed software to run nuclear power plants. Designing life in that software, I've been there, done that. And I'm actually, as far as I know, no one had died because of something that I have built. So I'd like to think that I'm fairly good at my job. Then my confession is that is not enough. That's not enough for me to actually go into my work every day and do what I do because what I'm, what I do is about the customers that we have at my company and making the world a better place for the people who use the software that I build, or that I design. And being an expert and the knowledge that I have right now and today is not enough for me to do that. And that's what I'm going to talk about.

Luckily for me some of the other speakers have brushed over, not brushed over, have mentioned some of these topics already. So it should be a bit of familiarity with some of what we've talked about. I'm pointing the wrong way because it's backwards for me.

So, just quick introduction on Darci. That's my Twitter id. That looks like a sock, it was meant to look like California, which is where I'm from originally. The reason I mentioned that is because I get very excited about what I do and sharing knowledge with other people and trying to understand their experiences so that I can improve what I do. Being from California I often I get quite excited and that means I talk really fast. So if am talking too fast, put two hands in the air and I'll slow down for you.

It's the only slide that's not hand drawn. You can tell I'm not actually a visual designer, I'm a user experienced designer. The one thing I'm not going to do, I'm not going to sell snake oil today. I am not going to give you a solution that suddenly makes your life better. I'm not going to tell you how to do anything differently. I'm going to offer you some options. I'm going to tell how changing my point of view and my mindset made my life at work better. So it's not snake oil, it's my personal story and my experiences. And hopefully someone in the audience will recognize something here and will be able to make their life and their work a little bit better because of my experiences. In other words, hopefully, you can avoid making the mistakes that I've made.

The premise of what I'm going to talk about is experimentation and testing. That's a little test tube in case you couldn't tell from my drawings. It's better than expertise on its own. The combination together

is even better that being an expert by itself is not enough to be good at our jobs. I'm going to talk a lot about user experience because that's what I know. However, most of this applies to almost anything that we do. Simply being an expert is not enough if you can actually test it and experiment. So, some of this fits into what Marcin was saying earlier around "Learning isn't very useful until you put it into practice." Being an expert isn't very useful until you can prove it in an experiment or a test. Ah, too many. Ok.

So, what do I mean by experimentation? So, my view as experienced user designer experimentation includes all of these different types of testing: A/B testing, interviewing customers, usability testing, formal usability testing, informal guerilla testing, unmoderated testing, testing paper prototypes, the Lean concept of get out of the building. So it's talking to people, it's understanding what they need and what they do. The world that I designed software for 5 years ago isn't the same world that I design for today. And if I stick with what I know, and stick to the same thing I've always tried and always done, I'm not going to be very good at my job and I'm not going to be able to address the needs of the customer who I build software for.

So, I talk about all the testing. One of the things that we do at my company and I've done it at a couple of other companies, as well, is we've implemented something called 5 on Friday which means that every Friday rain or shine, we have 5 customers come into our office to do some testing for us. We don't always know what we are going to test. We don't always have something to test, but we always have a valuable conversation with them. When they come in sometime we test live software. Sometimes we test our competitors' software. Sometimes we test paper prototypes. Sometimes we put sketches in front of them that were drawn 5 minutes before they turned up in the office. And sometimes we sit down with them and we have a conversation to say what is it you need, what gap are we not filling right now. And all of that provides us with valuable information that we would have never gotten if we hadn't talked to the people. And by making it a ritual, it's an expected part of every week. Every Friday 5 people come in. It's really started to become a part of the development cycle. So, the developers and the Q&A, the product owner and the business analyst look forward to having these sessions because they know they are going to learn something that it's valuable to what we're doing. Maybe it changes our road map. Maybe it just simply validates what we already have. Maybe it raises questions that we need to do further research into. But either way we end up with additional information that we didn't have. Not everybody can do 5 on Friday testing. If it's something that you can fit into your schedule, it's a really valuable tool because it involves everyone on the team in that testing and it's not just the users experience designer going off and coming back with " I have done research and here is the answer." That's not very helpful to anyone else in the team. Most of the reason that I apply like Agile is that it's about collaboration and it's about the team together. And if I as a designer have to go off into a dark room or run away into a laboratory, and do some testing, and I come back, it's a lonely experience for me and I don't get to involve the rest of my team.

So, you using approaches like guerilla testing or the 5 on Friday means you can involve the entire team. And hopefully since I'm guessing most of you aren't designers, and that actually sounds like something you'd want to do, you'd want to see real people using the software that you built. So, often too easy to sit in a room with your computer, with your team and Agile and have stuff on the walls and it's exciting and we forget the people who are actually at the other end of the system. And they have to use this and sometimes it's a system that they want to use. Sometimes it's a system that they have to use. And sometimes it's a system that it's life or death, that causes a life or death situation for them if it has to do with healthcare, if it has to do with transportation, if it has to do with lots of things. So, actually sitting in a room and never seeing customers means that you are missing information that you could have and would make you better at your job because you could empathize with what these people do.

I'm pointing in the right way? So, you can tell I'm not a visual designer. This is a hippopotamus. One of the reasons that I think this culture of experimentation is important is because it helps you fight the hippo.

How many people have heard of hippo? Ok. Not very many. So, I'll explain this to you because it's easy. Hippo stands for highest paid person's opinion. And often trying to fight a hippo, usually a stampeding hippo, isn't really something you want to do. The same thing is true if you are in a meeting and the CEO says "I think it should be blue." It's an opinion, it's coming from somebody who's highly paid, well respected, but probably not qualified to be making that decision or that recommendation. So, using an experimentation culture give you, gives everyone method for fighting the hippo. We're not even fighting the hippo. It's more about befriending the hippo.

So, I'll tell you a story that happened to me a couple of weeks ago. Someone in our company, will call him Dave, totally not his real name, Dave came to me and said "Darci, I want to try this thing. I've seen it on a similar site to ours, it's not our competitor. It's sin omething that I think is really valuable. I want you to do this for me now." I, my head I was thinking "Well, before I get a solid map for the next months, and we don't have extra time to do this stuff, and actually this part of the website it already works really well. We tested it. It's one of our highest converting products. Why do you want to change this?" And instead what I did in that moment as I swallowed thought and I said "That's really interesting. Tell me more. Tell me what it is that you want to see and how you think this is going to benefit our customers and our company." And I sat down with him and he's not a designer and he knows he's not a designer and he didn't want to do the designs. He wanted to see someone in the company turn his idea into reality. And I sat down with him. I got the information from him and I took it away to my team. I have a team of 7 designers that I work with and we sat down and we worked out some ways to, some options of what we can do with this design and we made some changes to it but it was still fairly true to his original idea. And because we also have web developers on my team, I skimmed a little bit and snuck it in. It wasn't really in the road map but, you know, it was kind of important to the company so we snuck it in. We got it done and we put it live for an A/B test and it doubled the conversion of that particular product.

And I could have taken that as a bit of a problem, but it wasn't my idea. As the sole user experienced designer in my company I should be the one creating the wire frames, I should be the one deciding the use of journey. And instead I thought of it as a win-win situation. He got his idea live, we sold more of that particular product which is quite important to us because it's one of our most popular. And I befriended the hippo. So, the next time when Dave, who's really not a hippo, you know, it's a good point. The next time Dave has an idea, he feels comfortable coming to me because he knows that I will support him when and how I can. It also means I don't have the sole pressure of being the only person who can do this design or that can up with these ideas. So, I now have a source of 400 people in my company who can come up with ideas and we can test them, we can put them into place under certain restrictions. We still need to A/B test or validate the things that are going to break the site. But I have a wealth of information and ideas that I didn't have before and it is because no one had actually tried to make friends with the hippo. They'd fought the hippo and you almost always lose when you fight with a hippo. As you can imagine. Although that one looks quite cute most of them don't look quite so friendly.

I won't make you stand up for this one, I'll just ask a question. How many people at work often feel pressure to always have the right answer? Ok, that's actually pretty confronting. I've asked this question before and got a lot of people who said yes. One of the things about testing and experimentation is a means that I don't have to be right all the time, I don't have to have the one right answer. I can fail, I can be wrong, I can come in and say "Actually I don't know what the right answer is." But here's some options. Lets' try both of those and see what happens. So it takes a lot of the pressure off to be right all the time. And obviously, you know, most of us work in Agile environments and we believe in failing fast and learning from our mistakes. The reality that I've experienced though is a lot companies really struggle with that point especially if you're working in something, in a situation where making a mistake can cause the company money. So, as much as everyone says "Yes, yes. Fail fast. Learn from the mistakes." It's ok not to be right. For me taking an approach of experimentation more than expertise has taken the personal pressure out of me and I don't feel I have to have the right answer on the tip of my tongue every

time someone asks me a question. It gives me the opportunity to test things, to learn things, to try new things, and to sometimes actually say "I don't know. Let's try a few things and find what works best." And for me personally as a bit of a perfectionist, that has massively improved my work situation and all I did was change my mind about how I did things and how I approached design.

Has anyone ever heard of hero design? A few people. Ok. So, what hero design is, is something very common in the US design community. it's one designer going into a dark room, running away, working their magic. No one knows what they do in that room, right? And they come out and they have the answer. And it's like the Ten Commandments, you come off the mountain and they are inscribed in stone. And that's what you are stuck with. And here in design often is the idea of one person in isolation away from their team without all the information, without all the options that are available to them and without all the expertise and experience of the rest of the team.

A culture of experimentation in testing means you don't, you can get rid of the hero designer and you make design a team sport. When I sat down with Dave, the hippo, I wasn't designing on my own, I was contributing my experience and so was he. And so were the members of the team. So what we were able to do rather than having the hero designer who runs out and says "I have the answer!" we work on the answer together. And for me, one of the reasons I really dislike hero design is to me it feels like going backwards into the Waterfall world. Someone goes off into a room with the requirements, they do their work and this is the answer and they throw it to the development team and they're off to work on another project. They're the superheroes. They try to solve all the problems. And that's not very motivating for them, it's not very motivating for the team they are working with. To be fair it's fairly lonely existence for user experience designer to do that.

If you think about user experience design most of it is about understanding people and trying to work with people and know how to make software better for them. So, being isolated from their teams is actually not very useful for them. It can cause real problems for the people, for the designers because they are then isolated and working in a vacuum and that goes against most of what Agile claims to be about. What I truly believe it is about.

So, I recently found this quote. Look at that slide, the picture is a little bit blurry. Sorry. "Best practices are someone else's solutions to their problems, not necessarily the right solution to ours." That's from the Poppendiecks and their Lean book. So, the thing I like about this is often in user experience design and many other fields that I've worked with , there's considered the best practice that people should follow. The best practice is often someone else telling you how to do your job and what you should do. In my many years of working in software and working in design, I have never solved the same problem twice. It might be a similar problem. It might be designer log in. How complicated can log in be? It's always a different audience, a different year which means there are different devices available, there are different expectations from our customers, the requirements for log in are different. Some log in can be new? Not super secure. Some logs in need to be secure because it's sensitive data they're protecting. I've never solved the same problem twice.

So, this idea that there are best practices that we can rely on to give us the answer is troubling to me because that best practice is a solution to different problems form someone else. Maybe it gives a starting point, maybe it's a useful place to begin from and then do testing on how does that apply to your own situation? But actually a best practice probably doesn't get you the answer for your specific situation that you're trying to solve. So, relying on best practices within design and other fields is quite dangerous. In a culture of experimentation and testing where everything you do is tested, everything you do is validated, or verified. You can actually start to learn what best practice is for your own situation and understand where that might change in the next situation that you find yourself in.

So, we're nearly done. I'm going to leave you today with 2 challenges. The first challenge is to adopt the phrase "Let's try that." rather than saying no when someone gives you a daft idea. Try it. Talk to them

even if you don't end up implementing it. Have the conversation with them because making someone in your company feel heard and feel valued will go a long way to making your life easier and making things easier for your team. So, adopt the phrase "Let's try that." and after you tried that, test it. Not just assume it's right. Test it and compare it to what you already have so that you have learnt something new from that situation and possibly befriended while you're at it.

The second challenge: use your expertise but test and learn every single day. So, this is about not getting stuck in a rut, not relying on the information and knowledge that you currently have because that information and knowledge becomes outdated incredibly quickly in a fast moving world. So, test what you do whether it's user experience, whether it's process with your team, whether it's a new architecture, whether it's a new way of doing Q&A. It doesn't matter what it is, try something new.

And I have to say after spending a day and a half here at ACE! my brain is bursting with ideas for what to do. I'm not actually sure which one I want to try first and how I'm going to take this back to my team. I haven't worked that out yet, but what I do know that I want to do is take these ideas that I've learnt and listening to Martin around "learning isn't real until you put it into practice". Take the things you've heard over the last couple of days. Test them and learn something from it. And then share that with your team. I would believe that any team in this room where there are people here could benefit if the team made a mission to learn something new about their customers every single day. We can improve our software, we can improve our teams and by then, by doing all of those we can make our personal world better than it is today.

Thank you. That's it.

Tammer Saleh: Teaching the Elephant to Dance

Tammer Saleh: All right, thanks Paul. And in the interest of your sanity I am going to try to keep this as short as I can. So, again, my name is Tammer Saleh and my presentation for you today is "Teaching an Elephant to Dance". It's about how the company I work for, Pivotal, used agility to transform a product we inherited called Cloud Foundry. So, hope this remote works. First of all, everybody, please stand up. All right. Is everybody up? Ok, go and sit back down again. I actually don't have a question for you. I just felt left out. Everybody else was getting you guys to stand up. So, thank you very much!

So, first we're going to talk about what Cloud Foundry is. We are going to talk about some of the accomplishment that we've, well, accomplished. We are going to talk about Pivotal as a company and its history, the challenges we faced, the solutions we've put into place. And then we are going to talk a little about Aagility and why it is important.

Again, my name is Tammer Saleh and I've been all over the place. I have a very wide amount of experience. I can, I self identify as a developer. I've been either in operations or engineering for the majority of my life. Now I find myself shackled. They no longer let me take code. I just tell people what to do. In the past I've written a book. I was VP of engineering, I was founder for a consulting company. Right now I'm in charge of part of the product for Cloud Foundry over in London. I'm also American so we brag a lot.

First of all, I think because of the venue and the people do not talk and also because it is Europe, some of you might not know what Cloud Foundry is. So, can I get a show of hands? How many people know what Cloud Foundry is? How many people know a platform as a service is? Ok, that's good. At least a third of you.

So, for those of you who don't know what a platform as a service, basically, it's a step up from what you think of an infrastructure in a cloud, so AWS, VSphere, OpenStack, Google compute Engine. Those are all infrastructures. Basically you ask for servers and I give you servers. But that was a lot of work on top of that to actually get your application deployed and available to your customers. So, our product, much like other ones like Heroku, Open Shift I think it's called, our product is a platform. Basically it's all the infrastructure and planning that you need so you can easily deploy your application as fast as possible. Ours in particular is open-source. It's designed to work on all the different infrastructures. It's designed for large installations. So, think enterprise installations and it comes packaged with a set of big data applications. So it's pretty ambitious for a platform. But from my point of view as a developer, the reason I get excited about platforms as services is that it enables agility in a devops way. As a developer I no longer have to worry about throwing my coat over to the operation side of the team. They give me a tool where I can deploy my code as often or as soon as I want. That's the key to agility.

Ok, let's talk a bit about the accomplishments that Pivotal was able to achieve with the Cloud Foundry platform. A year ago we were in a really bad place. The Cloud Foundry team was basically stagnant. They were doing yearly production deployments of the platform, once a year. They had no on premise product which was the main goal of the business, it was to make money from an on-premise product. They never shipped one. Everything was about architecture but they made no real progress on individual features. They didn't have an engineering discipline and they had basically zero testing culture. So, of course, this is a successful story.

Where are we now? In a much better place. Now we do weekly production deployments which for my taste it's a little bit slow but it's still much better than over a year. We launched our own prem products

in exactly the amount of time we expected to. We launch updates to that every quarter. There's been a consistent increase in quality, a measurable one. We have a huge testing infrastructure. I'll go into little bit more, more importantly, in fact the most important thing is the engineers are happy again. They see progress, they feel like they're accomplishing something. They're in a well-oiled organization.

So, let's talk about some history. How many of you heard of Pivotal Labs? Raise your hands, please. Ok, that's what I thought. It's mostly an American company. We are going to talk about Pivotal Labs, the transition into Pivotal and talk about EMC and VM where and how they contributed in the meantime.

So, Pivotal Labs was founded in 1989 by a friend of mine, Rob Mee and his partner, Sherry. It was, you know, originally it was just Rob. It grew developer by developer until it hit about 200 people. He used to say "We're not a machine that makes software, we're a machine that makes software-making machines." It's an awkward quote, but it's true. Pivotal has many offices and each one is designed around the same principles, almost cookie cutout and they're effective almost immediately. We have offices, the main one is in San Francisco. We have Palo Alto, LA, Boulder, Denver, Chicago, Toronto, Boston, NYC, and the one that I work in which is, which is why is the biggest one on the screen, in London.

We have all the big name clients that you'd recognize. I switch that screen because I didn't talk to the lawyers first. But probably if you know Pivotal Labs, you probably know them from Pivotal tracker, an Agile work-tracking tool for the developers. How many people have heard of Pivotal tracker? Ok.

So, again, it started as Pivotal Labs, a consulting company. In 2012, sorry, in 2012 they were bought by EMC. EMC at that time still owned a controlling stake in VMware. What EMC did was, they said "Ok. Pivotal Labs is doing a great job in one of our projects. We can't let that escape. So at this point we are just going to buy them, consume them into our organization." But they were smart enough to realize that if they did that, like any other acquisition, they would basically crush and turn everything good that's about the culture that they'd acquired. So, instead they did something really smart. They, excuse the pun, they pivoted Pivotal Labs out. So now, Pivotal Labs or Pivotal which is the company they named it, is its own identity, still partially owned by EMC and VMware and it includes Pivotal Labs. And what they also did is they handed Pivotal all these other next generation kind of Agile technologies which included Cloud Foundry, but also things like Reddit, Green Plumb, sprints for those Java developers and Pivotal HD. So, that's the history of Pivotal and how we ended up inheriting this code base and a lot of the developing culture that comes along with it. So, of course, whenever you inherit something like that there is going to be a lot of big challenges to face.

The most obvious challenge is the amount of technical debt that was sitting there in this code base that we inherited. This is one of my favorite photos. I don't know where this is, but people actually live there, so it reminds me of most of the software that I've worked in.

The other thing was, the culture was, well it was not healthy. There were a lot of cowboy coders. One of the things I love about this conference, especially this year is the motto of the conference "We don't need another hero." I feel that every day. Every day I feel like I inherited something. It was clearly written by one person at 3 a.m. with a bunch of gilt and now we have to fix it.

We also inherited a culture of architects. Does everybody here know XKCD comics? Raise your hands if you know XKCD because it's the best thing in the whole God damn world. Ok, great. If you don't, look it up, so funny. This is my favorite comic about how to be an architect. Can you all read that? So, he's asking for some salt, we waits a bit, he said "Where's the salt?". The other guy says "I'm devising a way of getting you any condiments in an arbitrary manner."

So, a culture of architects, people who want to big upfront design, people who want to make sure they've got it all right before they actually get anything out. And that's a big problem with Cloud Foundry and I want to make it clear, though. I don't want to blame the cowboys or the architects for this. They are just developers doing the best they can. They were told "This is the way you build software." You can't

possibly start a project as big a platform as a service without knowing exactly what you were going to be building in the future. It's bullshit, but it's understandable. The problem is really with the management structure that was directing them, telling them "This is the way things have to be done."

Another challenge that we inherited is that all of this is open source and personally I find it very challenging to work with the open source community. They're a wonderful thing but it's a burden as well as a benefit. It's difficult to get the same level of quality out of, again, individual developers working all over the world. So we had to devise a system for including the open source community. And then, of course, in general, we inherited remote developers in various different countries and so getting them to drink the cool aid, getting them to understand the Pivotal way was very difficult.

So, now that I've talked about the challenges, let's talk about some of the solutions. So, I think the biggest take away is that Agile practices are often fractal, they are often recursive, self- referential. So, for example, let's talk about pairing. How many people here have participated in pairing? How many people know what pairing is? Ok. I kind of expected more, honestly. Pair programming is when you have two developers sitting at the same computer, 2 developers working on the same problem with one computer. It sounds silly. You actually get more than 200% productivity and the developers are happier. The reason that it works is that, there are many reasons, but basically the cracks of it is that writing software is not part-part, debugging software is the hard part. That's where you spend your weeks. It's, I wrote it once„ that it took me day, I spent the next 5 days trying to find that one bug that was from a typo way back when. When you have 2 developers sitting at the same computer, the bug rate drops tremendously.

But, it's also about knowledge share. So, you don't just have one pair of developer sitting together all the time. You are constantly rotating developers. So, every morning during the stand-up the developers decide who's going to be switching in which pair. Now you have teams of these pairs, again, we don't want to keep knowledge silo in each one of these teams so on a regular basis we rotate the developers in each team to make sure that the developers are in a run time team, understand what's going in the operation team of in the front end team. You want this knowledge transfer to happen all over the company. But, this is just one office. So, great, we have knowledge transfer in one office. But again often Agile is fractal, recursive, self-referential. The same pattern can often be exploited into larger and larger arenas so we rotate developers between different offices, between Los Angeles and London, New York and San Francisco. Again it's all about that recursive nature. So, when you're looking at a problem like this, you don't always have to have a new idea how to do Agile. You don't have to come up with a whole new framework. You can just apply the same ideas at a bigger scale and it works.

Similarly, you don't need the entire company bought in at once. All you need is any representation from any level of the company. You need someone to advocate it at a leadership level, at the product and engineering sites. Once you do that, you plant these seeds of converts and they become evangelists. And again, things like pairing build this up so you get developers at every level and product people at every level and people in the leadership team saying "No, no, you, guys. This really works. We're not telling you have to do it, yet. But we're playing with it and you'll see that we are getting results." Similarly once you've planted those seeds that are evangelists you want to nurture this, promote them from within so eventually they become leadership. One of the things that I love so much about Pivotal, especially when is Pivotal Labs, is every pivot, which is what we call Pivotal employees. Every pivot was at one point a developer, you know, the founder was an engineer. He worked on client projects before he built the company up. Even the office manager and the lawyer, which I thought iwas hilarious, I met the lawyer and we were shaking hands and he seemed a little odd, or whatever. And he talked to someone else and he's like "Oh, I paired with him a long time ago. " So everybody comes from the same ground of understanding why Agile is important from , I guess we like to say it's in their blood.

Another important aspect is the strong boundaries between products and engineering. I think this is one of the things that makes the office I work in so incredibly healthy. Most engineering organizations I've

been to either the product side of the house has no teeth, right? How many are product people? PM's or project managers? Ok, often you find yourself unempowered. You know the right thing to do, but the developers just aren't listening. They are off doing whatever features they want to work on, whatever bug they think it's important. And that's unhealthy.

On the other side of the spectrum, though, in the places the developers feel as though they have no power, they're told by their product people "You've got to get this done. You have to get this done now."Here's a deadline, what can you ship? Right? Clearly I think everybody in this room understands how unhealthy that is. Pivotal is one of the few places where they figure out the lines, the boundaries, the lines of demarcation so well that neither side of the house feels superior or feels that they're under the heel. It create this culture of respect. It empowers the PM's but we're not allowed to say how things get done or when. We say what order but we don't say how long each feature can take. The developers tell us that and they get to focus on their craft and they get to focus on just writing good software.

The real challenge though is the old guard, how do we convert them? Well, the key is understanding that developers want to work in successful teams of smart people so if you are executing on this correctly, they start to see results, right? You pair program with them , you put them in small teams with constant rotation so that they see a lot of different things going on. You give them autonomy so that they can decide how their software gets built and you show them measure results. It makes them happy. Now, it makes most of them happy.

Sometimes those big, upfront architects, those cowboys, sometimes they join the organization because that's what they want to do. And, you know, they would just be happier somewhere else. It's ok to draw a hard line especially in a situation like this. It's ok to lose developers if they are going to be happier somewhere else. You want to keep the ones who are going to enjoy your culture because they are going to become those advocates.

One thing that I think it's really important to understand is that Agile isn't just about process, right? We are going to lose track off that when we care so much about the process. It's about results. And we take small steps so we can change direction quickly. That's the key of Agile, right? We test drive our codes so we can faster in the future. We don't architect a future proof so we don't create more than we need, that's waste. Sorry , we don't work nights or weekends so we can perform in our intellectual peaks. We don't get exhausted. We believe in sustainable culture. We pair programmers who make fewer mistakes so we don't waste time and so we can teach our peers. Again Agile isn't about process. You have to understand the problem before you try to come up with a solution.

Real quick I am going to show you a video that these guys made. This is recruiting video, but I'm showing it more because is the best taste of what it's like to work in the office every day. I worked with these people, some of them I paired with, doing code and they have this exact attitude all the time. I'm pretty sure they just hired a film crew to come in and sit and just talk. And they talked and they edited it and made it this, real fast.

Man 1: I am pivot.

Woman: I am pivot.

Man 2: I am pivot.

Man 3: I am pivot.

People: We are pivots.

Man 4: We are together all day to solve problems.

Man 5: We interrupt each other, a lot mostly.

Man 4: No, we are not.

Woman: We are in the business of writing code that will last.

Man 2: We engineers have a lot to say in what happens with the products.

Man 3: We don't introduce a lot of bugs along the way.

Woman: That's more about up here, than like down here.

Man 6: Pair programming is all about exchange of ideas.

Woman: like having 2 minds for one problem.

Man 6: 2 keyboards, 2 mice and constant discussions.

Man 4: It's actually fun to make software with another person.

Man 2 :Couldn't imagine not pairing.

Man 3:We get to learn and the clients get to benefit.

Man 6: Every day is a flood of social interaction and learning from fellow pivots.

Man 5: I'm learning so much every day.

Man 3: It's really collaborative.

Man 4: He makes a big deal about at the office.

Man 1: We are in this world that headshots with 5 points.

Woman: We have cater breakfast every day.

Man 4: Food all the time, drinks all the time

Woman: Fresh fruit and smoothies.

Man 7: The ping pong and the breakfast. They kind of tied to this bigger culture.

Man 5: I walk around all these people I would like to work with.

Man 3: People are really friendly here.

Man 1: It's a wonderful learning opportunity. It just feels that it has no balance."

So I didn't want that to sound so much like a plug. I wanted to show the smiling faces, the culture. It makes me happy to go at the office every day. But more importantly that's how you convert the old guard. You have people who are that happy about doing what they do and the old guard are going to come along with it. They want to be that happy, too. Let me say, the process does work and the reason that we do have process to make sure there's an even playing field. If you want to go fast you go alone. If you want to go far, you go together. So, everybody is playing by the same rules. We do gradual indoctrination.

Pivotal labs excels in converting teams. It's what we always did as a consulting company. We bring in developers and convert your engineering team. So, we just apply the same principles to Cloud Foundry. We have stand-ups, IPM's, retrospectives, inceptions. Someone said that one of the people that presented about retrospectives and how , you know, how boring and painful they can be. And I think that's probably right. One tip is to always have alcohol at a retro. That makes all the difference, beer.

We pair program. That's the actual developers over there. Testing is super important to us. We have a full suite now of unit, acceptance, integration tests. We have this massive continuum integration pipeline that expands multiple infrastructures: AWS, VSphere, OpenStack. So, we invested in that. We spent time saying "Ok, where's one of the problems with this code base? It has no testing infrastructure." So we were flying blind.

So, in summary, we have to support from all levels, gradually converting the old guards. We focus on results of a process. Everyone plays by the same rules so it's fair and importantly it means even elephants can be taught to dance.

I don't think there'll be time for questions, but I'll be having coffee over there so if anybody has any questions, just come and find me. And, of course, I must say we are hiring developers and product people.

Thank you.