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.