One Knight in Product - E179 - Dave Farley

Thu, Jun 22, 2023 5:34PM 50:57

SUMMARY KEYWORDS

continuous delivery, software, work, organisations, build, people, good, teams, product, talk, developers, ideas, software development, organise, users, engineering, bit, companies, technical, executable specification

SPEAKERS

Dave Farley, Jason Knight

Jason Knight 00:00

Hello, and welcome to the show. I'm your host, Jason Knight. And on each episode of this podcast, I'll be having inspiring conversations with passionate people in and around the wonderful world of Product Management. If that sounds like your kind of party, why not snag yourself an invite and join me and some of the finest thought leaders and practitioners in the world on https://www.oneknightinproduct.com, where you can sign up to the newsletter, subscribe on your favourite podcast app or follow the podcast on social media, and a guarantee you never miss another episode again. On tonight's episode, we ask ourselves what is continuous delivery? And why is it the best way to build quality products and build them fast? is continuous delivery something product managers can help with? And how can we sell the benefits of Best in Class software development practices to non technical leadership who see it all as just geekery? And can we prove it makes any difference to our product success at all? For answers to all these questions and much more, please join us on One Knight in Product.

Jason Knight 01:04

So my guest tonight is Dave Farley Dave's a legendary software development guru, consultant, trainer, YouTuber and author, who says has been having fun with computers for nearly 40 years. Me too, Dave. But if you're anything like me, you'd have started out with 10 PRINT "DAVE IS GREAT" 20 GOTO 10. Dave's an aeroplane enthusiast who likes to compete in aerobatics competitions, but he's also dive bombing and training his sights on poor software development practices, telling us all that we need to get good at continuous delivery. No, no, no, not your Amazon Prime habit. But making sure all of our products are always ready to release and we automate away all the unpleasant bits. Hi, Dave, how are you tonight?

D

Dave Farley 01:36

Oh, great. Thank you. That's some introduction. And mostly correct.

Jason Knight 01:43

Mostly!? I'm interested in what wasn't correct. Was it the first program that you wrote?

D

Dave Farley 01:47

No, that was pretty close.

Jason Knight 01:49

So were you a ZX Spectrum guy to start with then? Certainly, I was.

D

Dave Farley 01:52

That was my second computer. Third computer. So I started with ZX81, then a PET for a little while with 6502 processor in a little tiny nine inch screen. And then I graduated to the Spectrum.

Jason Knight 02:11

Oh, there you go. And the rest is history. Well, you're talking to a Your Sinclair Star Letter winner here. So you know, we're in the presence of greatness between us. But anyway, so let's get down to business. First things first, people might know you from your book, or quite possibly from your YouTube channel. But you're also currently working as a consultant. So what are you working on day to day with your consultancy, Continuous Delivery Ltd?

D

Dave Farley 02:31

So my consultancy is largely about helping people to adopt Continuous Delivery particularly... but what I think of just better software engineering practices, on the whole, mostly these days, partly through the success of my books, and my YouTube channel. I work for large companies who I can't name we're talking about afterwards. But lots of big organisations, usually, and often with interestingly challenging problems. One of the companies that I worked with a few years ago, for example, was Siemens healthcare, building medical devices, and communicating with them in the cloud. So helping them to adopt some of the practices and bring some of the engineering stuff that I talked about, to those kinds of, interestingly, complicated sorts.

Jason Knight 03:20

Well, I was gonna say it does sound like big companies like that are probably almost like the poster children for the types of problems that you call out and your books and with your blogs, and with your YouTubes as well like these legacy products that have been going for a long

time, maybe not even called products. They're just these things that exist, and they've got this accumulated organisational and technical debt over the course of 10-20 years or whatever. Do you tend to walk into quite a lot of those types of companies?

D Dave Farley 03:44

I walk into a lot of those kinds of companies, but they're not the only ones. I mean, continuous delivery, is, I think this is not an extravagant plan, continuous delivery state of the art for software development. It's what the best organisations at software development in the world do. And so it works, whether you're building literally spacecraft, medical devices, world class, you know, web companies, banks, cars, almost anything you can think of, if you want to build software, the way to do it better and faster, is through continuous delivery. So I mean, relatively high demand as a consultant, and I'm over time, you know, I've become probably slightly more bullish in the way that I talk about it.

 Jason Knight 04:31

Well, we'll talk about the bullishness in a minute, but want to talk about the aforementioned book, which I've got, it's obviously really good read gets very technical towards some bits of it. But I think just as a general piece of writing, it definitely talks about a lot of issues I've seen a lot of times in my career. You've obviously got the YouTube channel where you put a lot of clips up talking about some of the themes in the book and how to get the best out of software development. But there's a lot of people out there these days as you know, cranking out content. So how do you, I guess, decide I want to talk about keep fresh and make sure that you being super relevant, like, Do you have a specific content plan or just whatever comes to mind?

D Dave Farley 05:07

It's mostly what comes to mind. So I, the YouTube channel, in particular, which is a lot of what I work on these days, because it's it's like a priming pump for real things that I do. Yeah, but the YouTube channel, in particular is a bit of a, you know, a regular target. You know, once a week, I'm releasing something new. So I'm always working to a deadline. And so we've released, we're coming up on 300 videos, or something so far on that channel. So I have some themes, I have some topics that I talk a lot about, one of my big things is, is automated testing and driving development from automated testing as part of the continuous delivery process. But also, I'm really interested in software design and, and designing complicated design in complicated systems, in particular. And so I talked about those sorts of things. And largely, I just kind of, I'm always sort of, on the lookout for ideas. And I've also spent quite a lot of my time, my career working as in sort of technical leadership in teams and organisations. So I'm going to be interested in the human aspect of how we help organisations and people to work together to successfully build great software. So I've got a fairly broad spreaders topics that I tend to cover and talk about. And so I might do things well, I haven't talked about anything about teams for a little while. So I'm looking for, I think, on a on a theme of teams, or I haven't done a TDD thing for a while, and so on. And so and so I usually lucky, I get to come up with something usually.

Jason Knight 06:49

Ah, continuous delivery to your YouTube channel as well. So you're living the dream. But, speaking about content, I remember speaking to our mutual friend, Allen Holub, I know you spoke to on your channel, I've had him on the podcast before we had a great old time. But he's obviously got a bit of a reputation for speaking his mind and being very outspoken kind of absolutist and kind of bullish. And it's fair to say that some people are out there reacting well to that. And some people are out there saying, Well, I'm never gonna listen to this guy again. And it does feel that your approach is a little bit more, let's say mellow. It's not that you don't believe in many of the same things that he believes in, or that he doesn't believe in the same things that you believe in. But you seem, maybe it's just you know, personalities, but you just seem to be a little bit more relaxed when you're talking about it. So how do you balance being kind of opinionated and wanting people to work well, and giving those best practices to people without being too much like that, but not going too far the other way and like turning people off, you are being accused of being unrealistic?

D Dave Farley 07:52

Well, first of all, all of those things happen to me as well. But also, before I start there, let me just say that Allen's a lovely guy, and ...

Jason Knight 08:04

Oh, I love Allen!

D Dave Farley 08:05

Everything he says, he always says, with the best of intentions of trying to help people that I'm doing that too. So nothing that I'm going to say now is any criticism of Alan, he and I are very close in terms of our views on how software development should be practised and the problems of it. So I, I used to work for computer manufacturers writing bits of operating systems and bios and stuff like that, I ended my career when I was full time writing software building, low latency, very ultra high performance trading systems of various kinds. And so this goes kind of hardcore tech. So I am naturally I'm a geek, I'm a deeply technical person. And one of my aspects of Geekdom is that I am very strongly engineering focused. And so part of that for me, and part of my work is based on trying to take a very scientifically rational approach to addressing and understanding the ways in which you work and so on. So for example, I promote ideas like test driven development, which we've talked about continuous delivery, trunk based development, continuous integration, specific approaches to automated testing, have very strong views. Like for example, if you can't determine the reliability of your software, at least once per day, you're not practising continuous delivery. But I can kind of define the reasons why all of those are true, why I believe all of those to be more than just my opinion. All of those things are very, very strongly correlated with success. And I can point to the evidence, and that's the way that I go about things. So I've started to think about what I'm doing these days as in part being a little bit one of the science communicators, Brian Fox, or David Attenborough, somebody like that. I'm not on this scale, of course, but I'm trying to communicate complex ideas in simple ways, but without dumbing them down. I'm not trying to create sound bites or clickbait or anything like that. I'm trying to give you a reasoned

description of why I believe what I'm saying is correct. And so I come at it from that way, I'm not a person who tends to, I will call on expertise in other areas, but I'm not, I'm usually telling you what I think rather than telling you what somebody else is thinking, if you see what I mean.

Jason Knight 10:28

I do, but now I'm starting to think of you, you know, walking along a beach BBC logo, talking in a slow fashion about test driven development without the pizza turtles by doing stuff. This is gonna be an amazing TV show. And I'm gonna be strongly lobbying the Director General of the BBC to get this made.

D

Dave Farley 10:47

Well, funnily enough...!

Jason Knight 10:50

That's the scoop. But you just said that you're obviously really optimising for the technical side, like, that's your audience, you're deeply technical person yourself, you're a geek yourself, as you say. But at the same time, a lot of the enabling conditions for some of this stuff in the companies, the messy companies that people work for, also come from non tech people. So it's not always just about winning the hearts and minds of the developers, but also of the people that they work with, and their leaders and the executive leadership of the organisation. So do you recommend that those people read your book, watch your YouTube videos, go on your courses, or they need to go and find someone else that is maybe a bit more in the middle still believes the same things as you. But at the same time, your stuff is so relentlessly engineering focused that they wouldn't maybe bond with it so much.

D

Dave Farley 11:38

So I aim very firmly to try and make sure that my stuff is accessible. It does have technical content, my stuff. And that appeals to one segment of my audience, but but the large part of my audience is usually people that are experienced in software delivery, one way or another, I have a lot of people that tell me that they're product owners or project managers or technical leaders, CTOs, those sorts of people listening to my stuff, as well as the hands on people. So I'm not just cranking out, you know, saying this is our Java works or something. What I'm talking about is the socio technical practices of what is fundamentally an engineering discipline. This is a way of organising human beings collectively to solve technical problems. My favourite description of continuous delivery, is working in a way so that our software is always in and consistently in a releasable state. That has an impact on everybody, in every role, even remotely connected with software development, it has an impact on the way in which companies organise themselves as a way in which we structure the teams that we work on and divide at work. It has something about the way in which we released the change into products, the way in which we develop products, and think about them in terms of their impact on users, and how we collect information about how they impact on users. And when. And so one of the world's leading continuous delivery companies, for example, is Tesla. They ...

Jason Knight 13:15

Oh.. Elon!

D

Dave Farley 13:16

Yeah. Whatever you think of Elon Musk and Tesla are remarkably good at being able to target what their users want, and keep their products up to date live in production. That's a continuous delivery company, at multiple dimensions, that simplistic level, but also deeply in terms of the way factories work, and the cars are built and that are designed and all of those things. So ideally, this is a much broader topic than only the technicalities.

Jason Knight 13:45

No, absolutely. And I think one of the most interesting things that you say, I think in your book is that everybody is responsible for the delivery process, which I think really resonates certainly with me working with all the cross functional people that I work with, and trying to make sure that everyone's on the same page, and that we're all going in the same direction. So I genuinely genuinely agree. But if I'm not Elon Musk, but just some other random, maybe fairly tech, savvy person, business leader, someone who didn't get a spectrum in the 80s, and like what you said, sounds good, kind of, but I don't really understand what the actual concrete benefits of, for example, software was being in a releasable state. Like that sounds technical to me, how would you describe the concrete benefits to my business of that being true?

D

Dave Farley 14:35

But the concrete benefits are easy to express, because we've got the data. So So there's been a survey that's right now run by Google. So the company that started this app was bought out by Google. It's the group within Google's called DORA, which is DevOps research and assessment group. And they conduct something called the State of DevOps Report. And DevOps is I think of a component of continuous delivery. DevOps people think it's the other way around, but they're wrong. So so what what that study says, so over 10s of 1000s of surveys of projects over many years, it says things like, if you practice continuous delivery, your company makes more money. If you practice continuous delivery, your your organisation, the people that work for your organisations identify more strongly with the organisation. If you practice Continuous Delivery you built, you literally build better software, higher quality, fewer bugs, more maintainable, faster, into production, teams with a high score on the metrics, and behind that, that study spend 44% more of their time, on adding new features to their product than teams with low scores. These are commercial impacts, this is nothing at all to do. It's not nothing to do. This is a consequence of some of the technical behaviours, but it's not because of the technical behaviours. And this isn't some kind of technical nerdy impurity, this is this is engineering in the true sense of the word engineering in other disciplines is the stuff that works. If it doesn't work in another discipline, you change it until it does. That's what's going on. Now, with software. What we're describing here is an engineering practice that rolls out some of the dumb things that we've done in the past, and leaves the field open for doing the better things.

Jason Knight 16:32

Oh, I Well, let's talk about some of the dumb things that maybe people were doing in the past, and I guess maybe some people are still doing and now because they don't know any better, or they haven't read your book or watch your channel. What are some of the worst hallmarks of a non continuous process like things that are holding engineering teams back, stopping them delivering products and realising business value? By the presumably manual processes that are doing like, what are some of those processes?

D

Dave Farley 16:57

I think there are two fundamental ones that I would call out. One of them I've kind of referred to in the past is kind of the trillion dollar mistake. And it's completely understandable. But the trillion dollar mistake is confusing what it is that we do building software, we have manufacturing, oh, no relationship whatsoever. Ours is not a cookie cutter process. One of the unique properties of software, digital assets anyway, one of the unique properties is that the product is a sequence of bytes. however big or complex, that sequence of bytes is in terms of its operation as a system, once we've got a sequence of bytes, we can reproduce that sequence of bytes with zero defects for essentially zero cost. That means we never have a production problem. That means we never have a manufacturing problem. So our problem is distinctly different from that. Our problem is always because of the economics of that, the impact of that our problem is always to be creating something new, because otherwise we'd be being doing something stupid. If we're not producing something new in our context, then why on earth are we just taking and cloning for zero cost what we had before? Yep, so we're always doing something new. That means that we must optimise for learning. That means that every aspect of our strategy as an organisation, in carrying out continuous delivery is about learning. I've been interested this week in the launch of Apple's new virtual reality, Reality One headset.

Jason Knight 18:34

The three and a half thousand dollar VR thing.

D

Dave Farley 18:37

Yeah, I'm a geek, I'm interested, I think this is going to be some sort of fascinating thing. I want one of these things, to be able to have these big displays in front of me and all that kind of stuff. There are going to be parts of that that are wrong, there are going to be parts of that product that don't land with the users. There's a whole bunch of things. They don't know what's going to work at this point. Apple are, if not the... it changes, but they're probably the most successful company on the planet at the moment financially. They don't know what works, as Steve Jobs used to used to you used to say, how do people know what they want until we tell them but also, you know, we've got to learn. So you know, the iPhone, as it came out wasn't the iPhone that we have today, things evolve over time. This is deeply about what modern engineering in general is about, but profoundly what software is about. Software is soft, it's in the name, the clues in the name. So we need to be able to we need to be able to work in ways

that we can change it to evolve our products and morph them into the things that our customers really want to do that, to optimise for that kind of learning. We've got to think in terms of designing products that are changeable of gathering information back from the users to understand what what what lands with the users so that we can learn from that and adapt to it and build the software in a way in ways that allow us to do that easily. And all of these things are part of But So fundamentally, that's it. So the second thing that we've done in terms of this, which is kind of related to the first thing, it's kind of the, the wrongheaded, but understandable kind of production line thinking is decomposing the act of delivering software into a series of silos, a sequence of silos doesn't work. In order to be able to optimise for this kind of learning, we've got to be working in ways where we've got access to all of this deep flow of information, the people that are built the reality, one headset will be deeply intrigued, there'll be watching very deeply what people are saying about it, because that's the learning from it, not just the product owner or whoever else, but the developers, the technicians, the testing, all of those people are engaged and interested in what how their product is landing, what it means. And each of them will be taking different learnings from that. And we need to structure to facilitate that kind of thing. We've also learned that, you know, over the years, I'm a big fan of the team topologies book, which talks about ways of structuring organisations into smaller units of software doesn't scale very well, it's a fantastic book, if any of your audience haven't read it, they should go out and buy it immediately. Because it's one of the most ...

Jason Knight 21:15

Yeah, I should get Matt on really, shouldn't I?

D

Dave Farley 21:16

You should get Matt on, or Manuel. But they are... it's a fantastic book. And it gives us a language to express ideas that have been around for a little while. But small teams are dramatically more important than big teams. And that means we've got to organise everything that we do, to be able to have the small autonomous teams that are responsible for their own work, largely not handing it over to silos of people in different parts of the organisation. And so building up walls of bureaucracy that slow everything down. One of the other deeply profound findings from the state of DevOps report is that there is no trade off between speed and quality. The old story of the Iron Triangle is false. It's not true...

Jason Knight 22:03

Oh controversial, this is going to break all the project managers' hearts!.

D

Dave Farley 22:07

Yes, but I'm sorry, there aren't. And we have the data, we have the data now that says that it's true. So in order to be able to build high quality systems, you must go fast, by which I mean, you must make change in small steps, small, safer steps, evaluate the steps get feedback from

those steps quickly. And in order to build quickly, you must build high quality because you can't afford to go back and spend all your time fixing the mistakes of last week, this week, if you want to build a high quality system, if you want to build a system efficiently.

Jason Knight 22:41

Yep. And obviously, there's a lot to unpack there that maybe people can just let all that knowledge sort of settle on their shoulders. But I want to touch on the silos for a minute there. Because that's something that I think is really important. And obviously, talking about cross functional teams. But one of the things that are one of your videos that really resonated with me a while back was this idea of the role of quality assurance and testing within continuous delivery or just agile product and project management in general. Yeah, and obviously, that's one of the areas where product managers and the rest of the organisation and obviously QA teams themselves outside that tends to be one of the common handover points between the engineers and those teams, because a lot of stuff gets built, then it gets tested, or then it gives you a T ID or QA T ID or whatever it gets whichever t it gets done to. But in your mind, then if we're not having these silos, and we're not having these status codes, like what is the specific role of QA, and indeed, manual testing in a continuous delivery world?

D

Dave Farley 23:41

Well, first, first of all, that the idea of, of manual testing is never regression testing. In my view, I think we use automated tests for that. Yep. Humans are not good at being repeatable and reliable. So we want to automate all of the regression testing. So that leaves us with a need for the kinds of things that human beings are good at, we're probably never going to write an automated test that says, Does this blue button have blue text on it? Like human beings in this spot, that kind of mistake immediately, that kind of sensory thing that kind of impacting the kind of does this paint mental models in my head that allow me to extrapolate and move on to the next useful things I want to do with the software? Those things are not not easy to automate? And so humans are great at that. But the key idea in terms of the, you know, the role of QA, I think, again, the mistake that the siloing mistake, is where you put it in the development process. If we're going to build high quality systems, you don't get that by waiting until you're finished and they're looking at it and say, Now, that's crap. You get that by building the quality into the system from the first place, Deming that the kind of father of lean manufacturing... the grandfather rarely of lean manufacturing said, you don't inspect quality into a product you build it in. So what we want is that we want to organise our work in a way. So that we are building with high quality. If you imagine making something physical, then the equivalent of that is making sure that you measure things before you build them and or measure things as you're building them, and then fitting the pieces together and seeing whether they work. And we can do the same thing with software. So that's what test driven development is about. It's the equivalent of being able to do that building little jigs in which we could try out our ideas, and, and so on. So there's a role of quality bringing that up front into the development process. So one of the ideas that I promote to my clients and through my channels is the idea of acceptance test driven development. So we've got, we start off the development process early in the development process anyway, early in the kickoff, we create an executable specification that describes the behaviour that we want from the system doesn't say how the system works, but it says what the system is meant to do. And so that's a great role for QA people, because testing professionals think about these things differently, they will

come up with interesting, new ideas, it's also a good idea. Often it has a good impact on the kind of product design, because they're thinking in terms of the utility of the system, you understand the belief system, and all of those kinds of things. And so we want them engaged early in the development process before we built the stuff, ideally, before and during, while we're building this stuff, so they can be doing the they can help us get these specifications into better shape at the start. And then they can help us to to do that more qualitative, exploratory testing, as the software is evolving and being developed, as new features are being added not only when they're finished, leaving it until the software is notionally finished, is throwing it over the wall. And it's too late. As a developer, it's no good you coming to me, after I finished saying that I didn't do a good enough job, I want you to tell me that I'm not doing a good enough job while I'm not doing a good enough job. And then I can correct it.

Jason Knight 27:14

Yeah, that's really interesting, and I think speaks a lot to the idea that, I mean, this is something that people talk about in product circles all the time, like the handoffs or the relationships between product and QA teams and engineering teams, and like when people should be involved in the process. And I'm a big fan of getting everyone involved way up front. Yes. So that everyone knows what they're doing from the start, rather than someone trying to craft some perfect little document that says everything, and then send it over there gets developed, and then the QA team see after that, and then they test it, it just, it just all seems kind of a bit. Like there's the sound how that's supposed to work!

D

Dave Farley 27:49

Indeed and it doesn't really I don't think you've read. I've been building software in virtually every way that you can imagine our guests during the course of my career. And I've worked on lots of teams that didn't work very well, I've worked on lots of on what the software projects that didn't, it didn't work very well. And you know, I think I spotted some patterns. And all I've also been fortunate to work on some software that other people have called great and on teams that other people have called great. And it seems to me that you know, what makes those teams great is not that they're populated with geniuses, but that they are so much more collaborative than normal. There's some lovely data from years and years ago, I remember reading back in the 90s. Some time, there was some research done into the real Rockstar, high performing developers, the 10x. Developers, everybody loads. And they did this research over hundreds of projects. And they were trying to identify what it was that these 10x Rockstar developers had, and what the only thing that they could point to that was different. And that they shared in common, these great performers was that they talked to other people more often.

Jason Knight 29:12

Oh... developers don't like that sort of thing. Half the time, if you've worked with the same developers I have, which I'm sure you have at some point.

D

Dave Farley 29:17

Yeah, but what the data says and I don't mean to be rude to anybody. But what the data says is that those aren't the great developers, the great developers talk to people all the time.

Jason Knight 29:26

So like those ones, I'd be listening to this. But it's not just the product and the QA silo. There's another silo or maybe another kind of stage gate or even a waterfall, if we should even use that term. That does exist in some firms, especially b2b businesses, setting and maybe to larger enterprises as well. This is the kind of the idea that from the commercial side of the business that they've kind of got their own waterfall that they want you to slipstream into because maybe there's some press releases or marketing rhythm that they want to get into or their trade shows or maybe there's even fairly conservative customers that don't want the system to change all the time, because it's some business critical back office system, and they kind of just want it to stay the same isn't a way to competently make continuous delivery work with the rest of the quote unquote, business?

D

Dave Farley 30:14

Well, yes, it's still the best way of working, it is the best way of working. So it will do the best job. And whether you're targeting a particular date, or a particular set of features, it will do the best job of either one, what's impossible, what's irrational. And it doesn't matter what the process is. But what's irrational is to try and fix both time and scope. And people have known this for years, you know, the project management books have been saying this for four years, you can't fix time and scope. And then everybody, nearly all organisations go and try and do it. Which is, which is, which is just irrational. So continuous delivery is working. So our software is always in a releasable state. So I can guarantee you that we're going to hit the release date, I can't tell you what I'll be in by that release date. But we'll hit the release date! Or... continuous delivery is the most efficient way of building software. So we can build you whatever it is that you want, faster than other ways of working can. But I can't tell you when you'll get it. So which one do you want?

Jason Knight 31:24

But the follow up there, I guess is whether the fact that you're able to release your software continuously means that you have to release your software continuously.

D

Dave Farley 31:34

Now, it's a good question. And a confusing one. And the confusion is, is down to me and people like me, because the terminology is confusing. So the way that I describe it is that continuous delivery is about working so our software is always releasable. We don't have to exercise the release, we don't have to do that that's business contextual. That's a decision for the business about what makes sense. Now there are lots of good reasons why it's a good idea to release more frequently. But you don't have to be practising continuous delivery. There's another CD, which is continuous deployment, which is a subset of continuous delivery, which is, essentially what we do is that we just automate the decision to release. In continuous delivery, we usually

build something called a deployment pipeline, which is an automated validation that our changes are good, fast enough, secure enough, whatever it takes to make determine that they're releasable. So we use a deployment pipeline to determine that our changes are ready for release. And then if all of it if our pipeline says it's good in continuous deployment, we just automate it, we just push the change out into production. And that's fantastic. But you don't have to do that.

Jason Knight 32:42

What about things I feature flags? Is that something you advocate? So for example, making sure you do deploy everything, but you kind of have some method to turn stuff on and off, maybe either globally or just for certain beta customers or something like that. Is that something that you support?

D

Dave Farley 32:56

Yes. So one of the more challenging things that is a consequence of the way in which we work. So part of my definition for continuous delivery, is that whatever the scale of your software, you can determine the release stability of your software at least once per day. Now, if you think about a conventional development, and you've got a developer that goes and lock themselves away in a basement with their headphones, working on a branch somewhere for a few weeks, we don't know that our software is releasable. If we include their software, we don't know whether their software is going to clash with the the other developer in a different basement software. We don't know whether it's going to break everything. So the way in which we evaluate that is through continuous integration, we make progress in tiny little changes that don't all necessarily add up to complete features. And we evaluate those changes multiple times throughout the day, getting really fast, fine grained feedback, as our software evolves and and gets more and more functional. That means in a continuous delivery world, the consequence of that is that as software's releasable. So if the deployment pipeline says that's all good, it might go out into production. We don't know that it will, but it might. So yes, now we need techniques like feature flags to be able to turn off the stuff that says, actually, that bit is not ready for users yet. It's working. It's tested, we know it's safe, but it's not yet ready for users. And it turns out this this sounds, this sounds sketchy. This sounds like a lower quality solution. But the reality is opposite. The data says that this is the safer way to make changes. And then if you think about the reason why is interesting, it's all about these small steps. If I make a big complicated change, then there could be really nasty things hiding in that big complicated change. If I make a tiny little change, that's just one or two lines of code. I'm going to be pretty confident that I know what's in there, you find a test that code and evaluate those and all my tests pass, I'm fairly confident that it's safe. And even if it wasn't, it'd be really easy to back it out again. So it turns out that working in these tiny steps, even for safety critical systems, is a safer way of working.

Jason Knight 35:18

What I'm so old, but I'm assuming that some of the people you go into or maybe some of the people you engage with out there with your content, or just some people that you bump into a cocktail parties, or all these other things that consultants of your success go to some people

out there, they're gonna be like, sorry, Dave, that doesn't sound any good to me. I don't think we could do that here. So what are some of the most common objections you hear from people when you're promoting continuous delivery, but they just don't think it's for them?

D Dave Farley 35:43

There's, so my co author on Continuous Delivery was Jez Humble. And he had he had a great conference presentation a few years ago, called something like it couldn't work here. And he came to the conclusion that all of the objections, boiled down to three different categories are software's crap, our organization's crap, or our people are crap. And if any of those is true, then you fix it. I have that conversation all the time. You're quite right. And I'm belittling it. But I just wanted to point people in the direction of a good YouTube presentation...

 Jason Knight 36:21

Yeah, I'll find it and put it in the show notes.

D Dave Farley 36:23

Yeah, it's really good. It's funny. Jez is a great talker. But I think it's a reasonable summary. So the data is on my side, I think I can give a reason for the way that these things work. And I think that's coherent. And if people can show, you know, show me holes in my reasoning, I'm going to be grateful to them, because I'm a scientific rationalist. And that's, that's the way I learned. So, you know, I'm interested in these sorts of conversations. It used to be that I wouldn't be as bombastic as I have been today, talking about things like safety critical systems, because I didn't have the evidence. I could talk about finance systems, I could talk about commercial systems. And those are things I couldn't direct safety, Chris was putting now I've seen it work. Now we've got the evidence, I can literally point you to companies that build space rockets that do continuous delivery for space rockets, I can point you to Tesla, that build cars, they can make a change in software that changes the production line in under three hours. You know, there'll be churning out different cars in three hours. That's the kind of change that we're talking about here. The US military are applying continuous delivery to software in fighter jets and your other defence systems. So this way, we can No, I don't think that there's any category that you like, there's bound to be some kind of corner case that I can't think of. But there's no such a broad category. It works for embedded systems, it works for finance systems, it works for regulated systems, medical systems, automotive systems, games, anything that you can think of, you can kind of point to an example somewhere of somebody that works this way. And so there's not that kind of the barrier. So then there's the argument about, you know, your people not been able to do this or your software been in the wrong shape. And yes, if you've got a an old legacy system that wasn't designed to work this way. That's a challenge. It's more difficult than starting from a blank sheet of paper. But there are well trodden paths to alleviate it's horrible, hard, grungy work to tread that path. It's not an easy move. But it's completely doable. And the vast majority of organisations that end up as practising Continuous Delivery organisations started from there, not with a blank sheet of paper. And then there's the then there's the problem, but oh, yeah, well, this is great for Tesla, and Google and Amazon, but they're all geniuses and, and my team is not geniuses. That's not true, either. So you know, all of the data or all of the evidence is that it's much more about the

way in which people work than who turns up to do the work. In fact, Google did research across 180 teams, I've repeated three years to try and identify what made the really high performing teams. And I think, well, I'm pretty damn sure that what they were looking for was all this combination of skills, and this, this level of expertise, and all that kind of stuff. And it was none of that. The number one correlation for the highest performing teams was the degree to which the people that worked on the teams trusted one another, and worked together and the way in which they organise their work. So this works for beginners. We've got examples of that. It works for experts, you've got examples of that. And we've got in virtually every version of difficulty that you can think of the downside of continuous delivery, is really what I mentioned earlier, it's not an easy change to make. This is challenging. This challenge is nearly every preconception that most organisations and most people have about software development at some level. And that's not an easy thing to deal with. If I could wave a magic wand and get over that problem, I'd be a very wealthy man, I think Because because nearly nearly everybody wants to work this way, but it's just hard to get there. But it's possible in my experience.

Jason Knight 40:08

But where does much of the pushback come from then? Like, is it coming from the developers themselves? The engineering teams, maybe the CTOs or the VPs of engineering? Or are they kind of well up for it, but it's the commercial side, the leadership teams, the product side that are holding them back and not giving them time to do it? Or is it kind of both?

D

Dave Farley 40:28

I think the simplest answer is that it's both, you did get different pushback from different groups of people. There was a good Harvard Business Review article a couple of years ago, I'm going to paraphrase it terribly, because I can't remember off the top of my head. But it said something along the lines of high end, agile development has proven itself, the barrier is not demonstration that it can work. It's not evidence or the adaption to particular promises. It's the behaviour of executives in organisations. It's the traditional thinking in leadership in organisations. And I think there's, there's definitely a strong part of that. That's true. I think that many people, I think that many people are in the trillion dollar mistake, they're thinking of this as a manufacturing process. And they tried to apply, essentially, the techniques of a building a production line to something that a production line and doesn't work with production lines, developers push back for different readers, if you're a developer in a certain kind of organisation. And the requirements aren't really requirements, the requirements are just instructions for you to code as a solution in a certain way. And you're not responsible for the quality of the software, that's a fairly easy life, you get paid reasonably well, to translate what is what's essentially code written in English or another human language into code written into a programming language. That's a fairly easy life. It's not software development, it's just coding. But software developments more complicated than that. So there's some pushback. So developers don't really want to do their own testing, sometimes, if they've not been used to that they don't really want to take responsibility for the design of the software, because they used to be told those things, and they don't want to take responsibility to the quality of it as well that somebody else is looking after them. Those are the kinds of barriers and getting people to adopt these sorts of behaviours is difficult and challenging. But I don't know any

organisation. Or really, I don't know, any individual that's gone through this transformation, I suppose, into this new way of working, that would willingly go back and work in the old way, once they've experienced it. It is like the, you know, the walls falling from your eyes.

Jason Knight 42:44

All right, so let's assume then, I mean, again, this is podcasts primarily listened to by people building products, either product managers, some developers, some designers, startup founders, all of those people, many of whom might be listening to this thinking, What can I do to help developers get into this mindset? Or help buy them space to we can't even fix our generic tech debt? Yeah, let alone the entire way that we deliver software? Yeah. What can product managers do to help to advocate for some of these practices, or try to help to embed them within organisations and maybe join in with the persuading?

D

Dave Farley 43:22

I think that there are a few things, as you pointed out, I'm a consultant. And one of the ways...

Jason Knight 43:29

They sould just hire you!

D

Dave Farley 43:31

No, no, I wasn't, I wasn't, I'm very expensive, and my time's very...

Jason Knight 43:37

They can hire me, it'll be fine.

D

Dave Farley 43:41

But I, you know, I kind of come up with these mental models of what it is that's going on. And one of my mental, you know, I've got this kind of model of an ideal organisation. And then what I'm usually trying to do is I'm trying to say, so how can we move you in the direction of this idea? That's kind of partly the game that I play? One of the attributes of this ideal organisation is I think that in an ideal organisation, that every individual would be responsible for their own work would be able to see the consequences of their decisions, good or bad, and learn from them. I think that's a reasonably good thought experiment for what that would be like. So what would that mean for a product owner? So So one of the big mistakes I think that that's, that seems to be very, very, very common in the kinds of organisations that I consulting, at least, is that the requirements process has been denatured to the point where it's no longer really a requirements process. It's a design process. So requirements are put this button on the screen here, add this column to the database, make that faster. These aren't requirements.

Requirements are things that users want of the system. So people like product owners should start to really, really focusing on trying to describe behaviours that users want from the system, you could think of software development as a translation process. And our aim is to organise it into a series of small translation steps, each step to be reasonable, a reasonably small step from the previous one. So the starting point is nearly always somebody's kind of vague wish of what they what they wish the software that would do that it doesn't do now. And then the next step after that is to capture that in the form of some form of user story, we'd like to tell a little story about a user using the system, from the perspective of that user, say nothing at all about how the system should work. Nothing, it's not even that there are buttons on the screen of all fields to enter when we're thinking solely. I'm a user, I'd like to, I'd like to order a book. And I'd like to get the book back, you know. So then, the next little translation step is to come up with an example that if that example existed, demonstrated that this wish that's captured in our user story would have been met. But the need that was described had been met. And then we're getting towards the technical end. Now, if we could translate those examples into executable specifications. That said, I observed or, you know, I, here's all the setup of this scenario, I observe that this behaviour exists in the system. And still, that executable specification doesn't say how the software works. And the job of the software development team is to take that specification and translate that into working software, it's their job to come up with a solution, it's not somebody else upstreams, to tell them what the answer is, somebody else the job of the upstream process, and people is to define the need, and to express that as clearly and concisely as we can. So one of the things that product owners could do is start censoring themselves and not allow ourselves to say what the answer is not allowing themselves to define what the solution to the problem is, that can that can have the conversation. But in the written materials in the in the artefacts that we create in the communications that we establish, their job is to focus on the customer need, and represent that through the process, in my view. The other one is to think about, you know, how could we learn that? So a product owners job is going well, if they're coming up with ideas that users like? So you want feedback from the real world? On those ideas? So thinking about ways in which we could evaluate those? What's the point of this feature? Is it to recruit more users? Is it to narrow the margins on our sales or whatever else? It is, I don't know. But those sorts of ideas so that we've got some kind of way of measuring success. And using the kind of Site Reliability Engineering sorts of techniques of stating what that experiment, you know, HCH features and experiment and state what the goals of the experimenter, as part of the requirements process would be nice. And then there's the, you know, how do we learn from that? Well, success is due the features that we're specifying in this way, help teams to develop high quality software more quickly. And if they don't, if they're getting in the way, then there's a problem. And that problem needs addressing.

Jason Knight 48:27

No, absolutely. Well, again, a lot there to think about. And, you know, personally, from my perspective, I'm really, really keen and we touched on it before to get everyone involved, as soon as possible, really drive home, that kind of collaborative attitude, and not just we talked about it not pass the ball back and forth. Or, in some cases, like a dog, a bag of dog poop, is getting passed back and forward and no one quite wants to hold of it. I've definitely been in situations before where every time you wanted to release software, it was almost like someone myself a stink grenade in the office, everyone just wanted to be there when when the software was ready to be released. But in any case, I think it's definitely some inspiration for product teams, and hopefully, their cross functional colleagues and maybe even their leaders to think a little bit differently and maybe start to think about how they might make everyone's lives a

little bit easier. But where can people find you, Dave? After this, if they want to find out more about your course your content, talk about continuous delivery, or see if they can tap you up for any flying lessons?

D Dave Farley 49:26

Yeah, so the easiest way to find me is probably go to YouTube and search for continuous delivery. And you'll find my channel. I'm also a regular on Twitter, still mastered and as well but Twitter is @davefarley77.

 Jason Knight 49:42

Holding on by your fingernails!

D Dave Farley 49:45

Yeah, still there at the moment. And there are there are lots of other ways that you can find me from there. My courses... they're getting great feedback and they teach teams and organisations to do the kinds of things that we've been talking about. So that's really what they're about. If you're interested in that, go to https://courses.cd.training/

 Jason Knight 50:05

There you go. Well, I made sure to link that all into the show notes. And hopefully I'll have a continuous stream of people hadn't straight in a direction. Thank you. Well, that's been a fantastic chat. So obviously, really appreciate you spending some of your valuable time sharing some of your pearls of wisdom, and hopefully helping us to inspire people around the world to think a little bit differently. Hopefully, we can stay in touch but yeah, so now thanks for taking the time.

D Dave Farley 50:27

It's my pleasure. Thank you.

 Jason Knight 50:30

As always, thanks for listening. I hope you found the episode inspiring and insightful. If you did again, I can only encourage you to hop over to https://www.oneknightinproduct.com. Check out some of my other fantastic guests. Sign up to the mailing list, subscribe on your favourite podcast app and make sure you share your friends so you and they can never miss another episode again. I'll be back soon with another inspiring guest but as for now, thanks and good night.