DevOpsDaysDC 2019

Resilience Engineering: The What and How

https://devopsdays.org/events/2019-washington-dc/program/john-allspaw/

Video Recording - https://youtu.be/9f4-Z8Tasa8
Slides - https://speakerdeck.com/jallspaw/resilience-engineering-how-and-what

John Allspaw

John: Hey everybody!
I'm really excited. Oh, look at that. That looks awesome. This slide came out really well. So I'm really excited. And there's a lot to talk about. The topic is -- so a bit of disclosure. I'm here to challenge you this morning on some sort of preconceived notions, as I sort of talk about resilience engineering and resilience. The idea here is to provide fuel, like Nathen and Holly mentioned, to provide fuel for your conversations later. I'll talk about this in a little bit. Resilience engineering is a field. Is it possible that somebody could do a 40-minute talk on a field? Of science. I mean, I don't know if I were you I would believe me. If I could do that. So again, this is going to be fuel for discussion. So I'm going to do a little bit of mixture of snippets and perhaps some sort of controversial thinking challenge your paradigms. Here are some things about me, things that I've written and places that I've worked. Funny thing, so I gave this talk -- there's a not great talk, it was a great talk as far as a duet talk was concerned with my friend Paul in 2009, and at the time it was really weird. It wasn't clear to us. Paul and I thought we're going to talk about what we do, like what our perspective is at Flickr around this idea that maybe it's possible that developers and operations engineers could work collaboratively to support each other. And like help further each other's, you know, distance towards goals, and like cope with constraints and that sort of thing. It was pretty wild. And really not used to the software industry wasn't really used to. In fact there were a lot of people that came up to us after that talk and said I think you're a lunatic. Like you all are bananas. And deploying multiple times a day, you all should at least one person said: You should both be fired. We thought that's how radical it was. But it was an exciting time. Six months later, Patrick Dubois held the first DevOpsDays at Ghent. There were a handful of things that he enjoyed. So I just want to throw that out there. The topic of resilience engineering, I believe, is similar, in that there's

something happening, and I'm going to explain that. But right now there's something that's happening. Resilience engineering is a concept -- resilience as a concept, resilience engineering as a practice, as a field, as a community is still almost entirely unknown. And I could give probably conference talks all day every day and it would still be unknown for a long time. But there are glimmers of hope and some progress being made. So thank you for listening to me.

This doesn't really matter.

I -- oh. I'm part of a consortium, we're finishing up the second cycle, an academic and industry consortium called the snafu catchers and if anyone wants to talk to me later about your company being part of the third cycle, I would love to talk about that and here's the company that I work for at the moment. Okay. So here's my goal. If you walk away from this talk with any answers, that's going to be purely coincidental. My intention here is to help you ask better questions and lay out some threads that you can pull on elsewhere afterwards, maybe in Open Spaces to ask sort of better questions about resilience and resilience engineering. So here is -- here's the deal. Summary slide. All over simplified but we'll revisit this later. Resilience engineering is a very relatively recent field, aiming to create and sustain conditions where resilience can manifest productively. We'll talk about resilience later. Resilience is something that a system, your organization not a software, it's something that a system does, not what a system has. It's not a property, it's not a state. Resilience is, in another way of putting it, sustained adaptive capacity, sometimes referred to as continuous to unforeseen situations. These are vocabulary words to set the stage, and it's mostly important to set it aside from other paradigms that you're used to.

And last thing I'm going to end with, to say that it's our world of software, the software industry at large has some unique opportunities that other domains who have been involved with resilience engineering, don't have. But there are -- I wouldn't say significant, but certainly some real challenges that face to make some progress on that.

So resilience engineering. Going to talk about the field, community, and then I'm going to talk about what this practice looks like. I'm not going to talk about what that practice looks like very much. Mostly because it's sort of a stage setting. In order to talk about what it means to engineer resilience, I have to really sort of explore a little bit with you what I mean by that. It is an overloaded term, much like DevOps, much like Agile, much like a lot of terms. First I want to wipe away a bunch of either preconceived notions or potential points of confusion. So let me tell you what resilience engineering is not, first. It's not SRE. It's not DevOps. It's not

something invented by a particular company. It's not chaos engineering. Although it's becoming closely related to, in a number of different ways, if we have time we can talk about that, to chaos engineering. It's not automation either. So I want to set aside -- remember, there's resilience and there's resilience engineering. I want to sort of make a sort of distinction there. In that vein, resilience is not redundancy, it's not robustness, it's not high availability, it's not fault tolerance, it's actually literally -- it's nothing about software or hardware. Resilience in the frame of resilience engineering, which emerged sometime in the 2000s, early 2000s, is not a synonym for these things. It's something different. Something beyond a second order, higher order concept here. So it's a field and a community. Multidisciplinary. Largely, it came from a field known as -- the origins came from a field known as cognitive systems engineering which you can think of a punk rock slash splinter of what's now known as traditional human factors, early you 2000s, largely inspired cognitive engineers to think about this as they investigated and sort of explored a couple of NASA incidents at the time. Eight symposia with this community, I'm wrong, it's about 15 years, here, and here's some covers of books to prove to you that this has been a thing. Two weeks ago, I was in Sweden for the eighth resilience engineering symposium. If you have the opportunity to go to the west coast of Sweden during the summer, it's awesome. You should totally do it. Resilience engineering is a community. And it's largely made up of practitioners and researchers from these domains. This is what I mean by multidisciplinary. Resilience engineering, and the topic of resilience in technical systems and sociotechnical systems, has its roots in biology and ecology, but is largely studied in organizations from these perspectives. And working in these domains. So what you'll see up here is a list of domains that are traditionally known as safety critical domains. A lot of either fixing people, moving people, providing power for people, or -- I mean I guess killing people. There is a view that says human factors, the general so much of human factors came from the military, not just the U.S. military, but military. And that is for the better or the worse, working out, why somebody who was supposed to be killed didn't or why somebody was killed who shouldn't have, was. That got dark, sorry. Let's backtrack.

So you note here, I'm going to add software engineering, but this is only recent. This is very recent.

And it's still -- like I said, there are still baby steps being made introducing the world of software into this sort of community. Some people, I'm hoping that maybe a handful of people recognize -- are any of these names people that you recognize? Raise your hand. Great. Next year it would be great if everybody worked out -- yeah, I've seen those names and I've read those.

You certainly will. I'm going to expand on this. These are what I would say the OGs, sort of the heavies. This bottom row here, Nora Jones works at slack, Casey Jones running a start-up. There's me. These are people who went further in their career to get a degree in human factors and system safety of which resilience engineering is inextricably linked. Before I get too far in this, I want to mention this URL is maintained on a GitHub account by Lauren Huckstein who's at Netflix. It's not an explain or an introduction to the topic in various ways of exploring both personalities who have done work in this area and the topics and concepts.

So, resilience. Try to get some shared understanding here. Resilience could be described as pro active activities aimed at being prepared to be unprepared. This is different what we're used to. In software we're used to preventative design. We want to write our code, architect our systems, our infrastructure, all of the stuff that goes into the code and supporting the code as it's running, to take into account scenarios that are untoward or unwanted and be able to handle them gracefully. Graceful degradation is a view. And a lot of the thrust behind tolerance and systems are in that same vein of preventative design. The difference is that resilience engineering is, and resilience manifests in scenarios that are unforeseen. That were not and -- that were not imagined as being possible in the preventative design stages. This means that you can't justify it economically. It sounds paradoxical, but you are doing it. Resilience exists in the world whether we know how it works or not. Sustaining the potential for future adaptive action when conditions change, a simpler way of describing it, something a system does, not what it has. Perhaps a simplistic analogy, raise your hand if you've heard of chaos engineering? Perhaps a simplistic way of describing resilience is not what results from doing chaos experiments, it's about funding and supporting the teams that develop and perform those experiments. You see? So this unforeseen, unanticipated, these are hallmarks of complex system behavior and resilience is aimed at setting conditions and scenarios up so that these can be handled. But it's paradoxical. How do you prepare to be unprepared? This is what the community has been wrestling from a research perspective and all those other domains for about 15, 20 years. I love this quote. Things that have never happened before happen all the time. Of course for anybody who's in charge of or responsible for production systems understands of course this is exactly. Incidents in our world are effectively surprises. So in surprises lay the seeds that we could explore, try to discover, tease apart, pull apart in particular ways to find what resilience might look like in the wild. So attempts to make another analogy here. So robustness. I'm going to use an example of vehicle suspension. Shock absorbers. Designed for vertical disturbances that a vehicle might experience. Within a constraint. It's not

horizontal, it's vertical. In the case of struts it's a little horizontal. But in the case of shock absorbers, largely for vertical disturbances, but it's within a range of, a defined range of position and impact. And force of impact. Which means it's designed foreseen. This is the operating envelope, robustness. Robustness is what shock absorbers bring to the vehicle. Redundancy, the presence of a spare tire, designed for a situation where a tire in production is blown out or otherwise needs replacing. It helps and you're willing to give up a decent amount of space in your trunk in the vehicle to carry it around with you. You're burning fuel because it's extra weight, but you believe that it is necessary and so you are incurring a cost, you're sacrificing some goals in order to bring it, but in the end it's still redundancy. Both of these are preparations for vehicle-specific situations. None of them help with congested traffic or a shutdown of a particular route you need to take or the ill health of a driver. Other challenges. Resilience could be seen in this way as the capacity of finding ways of getting to your destination and what you might need for that. For example, when I was in Sweden, having cash in local currency, having some amount of fluency in the local language, access or an ability to get to rail and bus schedules, an ability -- cell service in order to maybe postpone my appointment or having someone else stand in for a period. That's this higher level that resilience describes. It is not robustness, it is not redundancy, it's this higher level. See how squishy it is? It's a bit hard to get your mind wrapped around it. David woods has said is resilience is a verb. If you're using nouns, state, property, something like that. It's not likely that you're talking about resilience. Resilience described by verbs.

So, sustained adaptive capacity, otherwise people have called it continuous adaptability. It means it's poised. It doesn't mean the adaptation is happening but you have the ability to adapt. Continuous deployment has been seen, described as a manifestation of adaptive capacity of the potential to adapt. In that you're investing a good deal to deploy, if you need to, when you need to. What you need to. Does it mean that you're deploying all the time. Deploying all the time would require all of these conditions be set up, but building that takes a lot of effort. Raise your hand if you would say that you're part of an organization that practices continuous deployment? Okay. For those with your raised hands, the world you lived in before continuous deployment cost less money to deploy than it did. Incurred a bunch of risk and a bunch of over things, but it was very different. So this is what we mean by investing. And especially investing in something that you don't necessarily have good, you know, defensible economic justification for. So can it be found in the wild? The answer is yes. And then how. We look at incidents. Incidents are briefly incidents provide a number of different productive

opportunities. If we know how to look into them. If we know what to look for, and what to do with what we find.

So let me start with -- we're going to talk a little about incidents and how we might be able to do that. So let's start with this. All incidents can be worse. Disagreement? Great. You're my people.

The question then is, if all incidents can be worse, then what are the things that people are doing to prevent it from getting worse? Which generally speaking, we don't give a lot of attention to. We give a lot of attention to the things we do to fix it, maybe a handful of other details. But there are a number of different things that we're doing. In fact, even all the time, you're doing things, to prevent incidents. Those generally don't get a lot of attention. When you do have an incident, it conveniently provides attention for you to sort of explore. But that is the -- that is the general MO. If you look at what people are doing, which normally would be seen as unremarkable or maybe described as oh, that's just good engineering or oh, that's just expertise, you know. She just knows what to do. She's good under pressure. You know. Sort of those sort of fuzzy sort of descriptions. So how can we find this? Well, first we would have to find incidents that have a high degree of surprise, number one. If there is no surprise, then tempo, time pressure, risk of consequences, is not very high. And that's not great for a number of different reasons. To find sources of resilience. We also want to look for incidents whose consequences were not severe. This is paradoxical. You would think well, the bigger the incident the more attention. That's actually a lie. That's actually the opposite. The more severe, more visible an incident is, the greater the attention and clamor for answers quickly. A quick wrong answer, a quick simplified, over simplified answer, is always preferable, organizationally, politically, to developing an explanation for an event, than a more defensible detailed, less simple set of descriptions. Unfortunately.

So we look closely at details of these sort of medium sized events that have high surprise about what people were doing. How people work together. Diagnostic activities, therapeutic activities. The costs of coordination between different teams or different expertise. How do hypotheses evolve and change over time. And once you can find these elements, these qualities, you want to protect them and actually bring a lot of attention to them to support them. An example is Sylvia. Everybody works with a Sylvia and everybody knows if this weird esoteric database over here that just generally runs like if it's having issues, if you've been working at this company for a while, you call Sylvia. That's the answer. That's the run book. Step one call Sylvia and Sylvia just knows. It's fluid whereas the rest of the team might be freaking out. All of

a sudden when Sylvia goes on vacation, lots of people notice. In that way, you would say that this esoteric domain expertise that Sylvia has is a strength. You should be supporting Sylvia in much better, explicit ways. So what is indication, what does novelty look like? So here's some real snippets from real communications during real incidents. Indications that people are surprised is actually quite easy. You don't have to have a degree in qualitative research and data analysis. You know what this looks like. When you find contrasting mental models, people trying to make sense out of what's happening, asking each other questions, taking information, observations that others are giving them and fleshing out the gaps or the mysteries that they have to make sense of what's happening, is happening almost fluidly, especially for teams who have been working together for a long time. In some cases slacker ISE transcripts give a unique window, especially with remote teams. Quite often when you have people who are co-located and they're doing a video conference or just incident transcripts with really successful, really progressive teams, these communications sound a little like this. Hey did you -- yeah. What -- no. That's not today. Yeah, but, no. I checked that. I don't think -- do you think? All right yeah. Can you? I got it. Okay cool. What did you find? No. No. Shit. Not a lot from research perspective to go on there and you have to double down on your interviewing. Anyway, why wouldn't you look at incidents with severe consequences? I'll say it this way. Scrutiny from stakeholders with face-saving agenda tend to block deep inquire. These meeting severe incidents, the costs of getting details, the worst-case scenario is you get engineers talking to you, look, I don't know why -- it wasn't that big of a deal. Exactly. It wasn't that big of a deal. That is why we need to know more details about it. Because it wasn't a big deal. Which means that there is stuff that you did. What are stuff that you know? The thing about resilience engineering research and finding resilience in the wild, especially in software, relies a great deal on tacit knowledge elicitation. Experts are not necessarily expert about what makes them an expert. They don't know what they know. In fact some cases muscle memory is the only other explanation -- ah, it just didn't feel right. Something was weird. There are techniques, the NTSB trains them all the time in their investigators. There are techniques to unearth this tacit knowledge. So Goldilocks incidents are the ideal.

Initiative. I'm going to link to a couple of papers at the end. There are a number of essential characteristics of resilience that bear out across multiple domain case studies and research in the field. Initiative is one of them. It's only one of many, but it's a bit of -- this is sort of a whet your appetite. Bear with me, sort of a, oh, academic-sounding description here. Initiative is the ability of a unit, an adaptive unit, a team, for example, to adapt when the plan no longer fits the

situation as soon from that unit's perspective. The willingness, even the audacity to adapt planned activity to work around impasses or to seize opportunities in order to better meet the goals better intent behind the plan and when taking the initiative the unit begins to adapt on its own without asking for and waiting for explicit authorization or tasking from other units. This is initiative in the manifestation or the expression of initiative is something you would look for to find sources of resilience. Not all incidents have sources of resilience. This is a element that if you can identify is there. I'll give you a case where it wasn't. And what I would say is then a case of brittleness. Absence of initiative.

2010, Knight Capital trading algorithmic hedge fund trading firm. Raise your hand if you've heard of this incident. All right. Then great. Let's just surf through the summary. New changes were deployed to participate in new market. Unexpected algorithmic mechanisms, basically set unbounded automated trading activities. The team rolled back the changes and the situation got much worse, seven times worse. And at the time the team did not believe it had authority to halt the system and they were relatively nervous to go to management, because they didn't have a good understanding, they didn't want to go to management to ask about halting, because they didn't feel comfortable or confident that they could explain what they knew about what was happening. $440 million loss in 20 minutes. Interestingly, a handful of years later, the New York stock exchange experienced some leading indicators that something wasn't quite right. And they escalated -- they immediately got all of the authority they needed and they halted all trading on the New York stock exchange and they took it down for four hours and it worked out. It didn't -- it didn't experience the type of financial loss that this had. But of course, when you do something like that, everybody was really pissed off they took it down for four hours. So you're kind of screwed if you don't, and you're screwed if you do.

I want to talk a little bit like let's bring it from this abstract to your world. So in responding to an incident, a handful of questions to get you sort of thinking around these terms. Do you have access to contact details for everyone in your organization? If you go with the idea that it's -- you won't be able to know ahead of time who you might need to speak with, at what time when you -- about what. Can you think about any actions that you absolutely need permission to take, or are there scenarios where you are granted explicit authority to take decisions as the shepherds and escorts of production systems, to make those sacrifice decisions, to halt your system, for example. Are there -- do you have the ability to flip all feature flags at your disposal is another way of thinking about it. What repercussions exist for violating procedures? Or, ready? Relaxing compliance.

The last question I want you to think about which is can you anticipate what neighboring or related teams need in the future, that you have, that they may need, that you have, expertise, resources, that sort of thing, and can donate to them these resources before they need it, even if it sacrifices some of your local goals. If so, this is a manifestation expression of resilience. Adaptive behavior that can be borrowed, donated, and in a reciprocal relationship, when it's necessary. This brings us to -- we're almost done here. Brings us to this question which is can resilience be engineered? Given everything I've told you about resilience, can it be engineered. I'm going to be honest here. This is the answer: Maybe.

There are more over the last couple of years, there are more signs, glimmers of hope that have emerged, that I can't -- that I don't have enough time to talk about in other domains. Mostly those domains are in intelligence analysis and there are papers, you will see them in the resilience papers, in emergency room units in hospitals, in trauma centers, and there are a couple of cases or at least one case in software that's being worked out at the moment. And sort of being explored and written about.

So here are the challenges. And here's -- this is the controversial bit. What are the challenges with the DevOps community sort of embracing some of this stuff? Well, first is inertia towards a status quo and over simplification. This is 40 minutes and I haven't scratched the surface. This stuff is pretty abstract and I don't worry about that, because this is also the crowd that is making some pretty good progress with understanding distributive systems and if you can understand a distributive systems paper, then Jesus, you can understand a resilience engineering paper. I would say as a whole in software we don't have a great ability, almost an inability, chronic inability to learn from other domains. Why? Because unfortunately silicon valley ruined it for everybody who's not in silicon valley which is we're always full of ourselves we invented everything and everything else invented elsewhere is no good. I'm only half sarcastic. The last thing is techno fetishization. Wait can I automate some of what you're talking about, what tool do you use, what do you think about observability? This is about cognitive work, it's not about tools. And so these are going to be, if we start from these principles and then build tools to help us, after we understand what we're building for, then we've got a shot. So the status quo beliefs really quickly a tyranny of metrics and shallow data. Raise your hand if there's a person in senior leadership that you have to make a graph about a butch of data about incidents. Sometimes bar graphs. Yeah. There's another slide about that at the end. Under investment in real incident analysis expertise. What passes for effective and productive post mortems right now is hello world. Come talk to me later and I'll expand on that.

Probably longer than you would want to listen to me. Over simplified models such as one size fits all. I'm going to leave you with this last little bit before I summarize, inconvenient shallow data. In order to understand where resilience manifests in the wild, which is necessary in order to have any notion of being able to engineer it, we have to see what it looks like and see how it manifests in context, then we need to be able to look deeper into incidents. What prevents us from looking deeper into incidents is the notion that we can describe sufficiently what an incident has to tell us with this really sort of skeletal, shallow data. Meantime 2X numbers are negotiated. These are not objective. They are negotiated and also they're averages. All incident data is reactive. It's scoped to the events that you have your focus on events. You don't have -- where is your denominator? Your denominator is all the events that could have happened but didn't. This is the paradox of resilience engineering. To look for and accentuate and enhance all the things that normally prevents issues from happening. Not an absence of incidents, but the presence of capabilities that prevent incidents normally.

Trending frequency and mean time to blah blah blah tells us nothing about learning, tells us nothing about prevention, expertise or proactive capacity. It does, apparently, have value from a marketing stand point, or at least from ITL certification stand point. But it doesn't tell you anything. If you show me a graph of your incidents and mean time too things going down, it's not evidence of learning, it's evidence that people have worked out how to make a graph look good. Bottom line revisited.

It's a field, it's a community. It's something a system has, something a system does, not what it has. It's not a property. You do not reach a resilient state and then you stay and maintain that. It is sustained adaptive capacity to unforeseen. Keyword, unforeseen. We've got challenges here. If you want to talk more about this, I'm on Twitter. And here are a handful of things that's helped fuel some of the notions here. My hope is that you're sufficiently intrigued that I'm either -- there's something here, this guy has something here, or I'm absolutely full of shit.

Regardless, both of those outcomes are great for me. So thank you.

[Applause]