AI and Software Engineering: From Strategy to Execution

Full Transcripts

1
00:00:05.910 --> 00:00:08.940
David Mantica: I repeat myself multiple times.

2
00:00:09.100 --> 00:00:21.990
David Mantica: But that's… I am a meanie, but that's just because of people coming in. Anyways, if you want to zoom in and tell us where you are, where you chat in, where you tell… tell us where you're zooming in from, that helps you get a sense of chat.

3
00:00:22.130 --> 00:00:26.330
David Mantica: Also, we had some adjustments to the agenda due to illness.

4
00:00:27.160 --> 00:00:37.490
David Mantica: Illness is no fun, and one of the speakers was definitely not feeling well. So, he had to pull out, so we added somebody new. Cape Cod! There we go!

5
00:00:37.690 --> 00:00:40.720
David Mantica: Beautiful place. Seattle, wow, there we go.

6
00:00:41.410 --> 00:00:43.140
David Mantica: Seattle!

7
00:00:45.150 --> 00:00:46.310
David Mantica: Love it!

8
00:00:49.520 --> 00:00:52.709
David Mantica: Oh, it's very… yeah, it is earlier there, that's for sure.

9
00:00:54.910 --> 00:01:02.819
David Mantica: Portugal! Now we're getting… now we're having some fun here, too! Seattle all the way down to Portugal, very cool! Oh, wait, up, sorry, to the left.

10
00:01:03.350 --> 00:01:06.469
David Mantica: To the west, east, well, at that point, it's either way.

11
00:01:06.870 --> 00:01:12.439
David Mantica: Ottawa, right, great. Westminster, Colorado, thank you all.

12

00:01:12.550 --> 00:01:13.960
David Mantica: Rockin' and rollin'.

13
00:01:15.730 --> 00:01:24.330
David Mantica: We'll get started here 2 to 3 minutes after the top of the hour. Laura sent the agenda. Laura, you might want to send it a couple more times, because people who are new don't see it.

14
00:01:30.680 --> 00:01:35.739
David Mantica: Oh, is that the maple leaf? Very cool. I'm gonna put a number 2 next to the maple leaf, that's neat.

15
00:01:40.840 --> 00:01:46.590
David Mantica: Oh, but then we can have some trash talk about, Olympic soccer, Olympic hockey.

16
00:01:46.890 --> 00:01:48.500
David Mantica: But nerve! Cancel play.

17
00:01:48.500 --> 00:01:49.190
Sean Miller: Fair enough.

18
00:01:49.490 --> 00:01:52.230
David Mantica: Or not.

19
00:01:52.520 --> 00:01:57.799
David Mantica: Hey, we still have… we still have a couple… we both still have a couple more games to get through. You guys had it tough yesterday.

20
00:01:58.270 --> 00:01:59.430
David Mantica: So did we.

21
00:02:01.760 --> 00:02:04.269
David Mantica: Lot of NHL players out there, man.

22
00:02:04.520 --> 00:02:06.439
David Mantica: A lot from different countries now.

23
00:02:08.419 --> 00:02:11.139
David Mantica: Yep. They're definitely taking it serious, too.

24
00:02:14.220 --> 00:02:17.570

David Mantica: to see if Sidney Crosby can play or not, that's gonna be interesting.

25
00:02:17.870 --> 00:02:35.869
David Mantica: Well, thank you all for joining us. This is AI and Software Development, Software Engineering. We have a phenomenal agenda for you with some changes. Laura's been chatting in the URL so you can look at the speaker bios. You can look at the updated agenda. We added a new speaker.

26
00:02:35.870 --> 00:02:45.749
David Mantica: But if you want, if you could Zoom… you can chat in the Zoom, or zoom in the chat, and tell us where you're coming from, where you're zooming in from, we'd greatly appreciate that. We'd love to know.

27
00:02:46.280 --> 00:02:50.519
David Mantica: Because we're very, very nosy here at AI for Software Engineering.

28
00:02:53.710 --> 00:02:55.240
David Mantica: Alright!

29
00:02:56.520 --> 00:03:03.630
David Mantica: Oh, we got an AI… AI agent in! Okay, we got one note-taker, see how many more note-takers did we get? Mr. Wessel.

30
00:03:03.630 --> 00:03:06.049
Scott Bird: With permission, feel free to tell me.

31
00:03:06.050 --> 00:03:08.139
David Mantica: Aw, we're gonna kick that crap out, baby.

32
00:03:08.860 --> 00:03:10.100
David Mantica: I'm just kidding.

33
00:03:11.030 --> 00:03:17.880
David Mantica: Hey, man. Mr. Wessel, you're a TV guy. Have you watched Station Eleven? Has anybody ever watched Station Eleven on.

34
00:03:17.880 --> 00:03:20.949
Tom Wessel: I have, yeah, that's actually… I really enjoyed it.

35
00:03:20.950 --> 00:03:23.430
David Mantica: Holy cow, I loved it!

36

00:03:23.910 --> 00:03:33.519
David Mantica: I watched it without my wife, per our conversations, because I know she would like it. But I… I mean, that's… I… I… it was amazing, they did such a great job with it.

37
00:03:33.690 --> 00:03:37.259
Tom Wessel: Oh yeah, definitely. I haven't read the book yet, but yeah, great show.

38
00:03:37.400 --> 00:03:45.100
David Mantica: I read the… I read… I did some research, I read the different… read about the differences between the movie and the book. There are some significant differences. Kind of cool.

39
00:03:46.300 --> 00:03:53.330
David Mantica: Alright, so Station Eleven, HBO, great numbers, if you're looking for something to watch, 10 episodes!

40
00:03:54.080 --> 00:03:57.540
David Mantica: Pretty wild stuff, post-apocalyptic. Anyways…

41
00:03:57.760 --> 00:04:07.959
David Mantica: We're at the top of the hour, but remember what I say? Two to three minutes after the top of the hour, we'll get going here. We'll get the kickoff started with our conference director, Mr. Jespers.

42
00:04:07.970 --> 00:04:13.020
Joseph Hudson Jr: As well as, just some basic introductions, and then we'll jump right into…

43
00:04:13.300 --> 00:04:16.180
David Mantica: Mr. Elliott's presentation…

44
00:04:16.660 --> 00:04:36.460
David Mantica: But again, as you're coming in early, let us know where you're coming in from, so you can tell us what you're hanging your hat. We got Portugal, we got India, we got Ireland, we got Ottawa, we got Massachusetts, Colorado, Virginia, a lot of folks from… oh, we got Pakistan? Fantastic! Love that, love it. -Oh, we have New Zealand!

45
00:04:36.910 --> 00:04:41.379
David Mantica: You guys… the presenters better do a good job, because poor Andy is up…

46
00:04:41.610 --> 00:04:46.020

David Mantica: Crazy time right now, so… it might actually… oh, he's my old boss.

47
00:04:46.170 --> 00:04:49.410
David Mantica: -Oh, Corey, don't say anything bad. Kenya!

48
00:04:49.810 --> 00:04:53.389
David Mantica: All right, folks, we have Kenya. This… now we're rockin'. Now we're rockin'.

49
00:04:53.670 --> 00:05:07.749
David Mantica: Alright, Firefly is up, some of these Fireflies working for us as well. A couple more minutes, then we'll get going. We have a very good sign-up number, so… and I just want to make sure folks get in. We had a very, very good sign-up number.

50
00:05:08.000 --> 00:05:11.209
David Mantica: But I know afternoons can go crazy.

51
00:05:11.330 --> 00:05:22.199
David Mantica: And as you know, you get all the recordings, you get the slides, transcript, so if you miss anything or have to go in and out, that's going to be available to you after the session.

52
00:05:22.460 --> 00:05:24.529
David Mantica: Alright, one more minute!

53
00:05:24.730 --> 00:05:27.030
David Mantica: One more minute, then we'll get kicked off.

54
00:05:27.280 --> 00:05:29.560
David Mantica: Mr. Jespers, are you ready to kick off?

55
00:05:29.900 --> 00:05:31.000
David Gijsbers: Absolutely.

56
00:05:31.520 --> 00:05:33.319
David Mantica: Alright, tell you when.

57
00:06:00.670 --> 00:06:01.520
David Mantica: Alright.

58
00:06:03.660 --> 00:06:05.410

David Mantica: Alright, there it is!

59
00:06:05.560 --> 00:06:10.390
David Mantica: Dave! Dave! Kick us off, my friend! Let's do it!

60
00:06:10.600 --> 00:06:20.019
David Gijsbers: All right, thank you very much, Mr. Manteca. Hi, I'm David Gisberos. I'm, Director of Solutions Delivery for Interpros. We are…

61
00:06:20.370 --> 00:06:29.750
David Gijsbers: We have an AI and software engineering practice. We offer advisory services, training services, and staffing services for companies that are adopting

62
00:06:29.840 --> 00:06:43.709
David Gijsbers: AI into their software engineering practices. Our first speaker today is Steve Elliott. Steve is with, Deltwork, so without further ado, I'd like to hand it over to Steve.

63
00:06:49.150 --> 00:06:50.689
Steven Elliott: Here, thank you.

64
00:06:50.830 --> 00:06:52.820
Steven Elliott: Everyone see my screen okay?

65
00:06:53.790 --> 00:06:57.350
Steven Elliott: Excellent. Okay, great. Yeah, hi everyone,

66

Decisiveness Is a System, Not a Trait

00:06:57.690 --> 00:07:09.029
Steven Elliott: Thanks for, thanks for having me. Yeah, so I want to kick this off and just maybe start with a simple question, just to get everybody thinking here. So, think about the last decision that took

67
00:07:09.820 --> 00:07:21.870
Steven Elliott: too long in an organization you were working in. So, doesn't have to be a bad decision, just one that was slow. A decision, you know, that took weeks when it probably could have took days, or a decision that required

68
00:07:22.150 --> 00:07:26.879
Steven Elliott: 3 meetings when you could have done it in a single meeting, or a decision that needed

69
00:07:26.910 --> 00:07:40.909
Steven Elliott: sign off from people who aren't really even accountable for that decision. So be… be thinking about that, and kind of, as you think about it, kind of… let me tell you what I've observed, you know, leading companies, consulting with senior leaders, you know, working in larger companies.

70
00:07:40.910 --> 00:07:47.260
Steven Elliott: I was a leader at Atlassian, got to see a fast-moving company trying to make big decisions cross-functionally.

71
00:07:47.260 --> 00:07:52.120
Steven Elliott: And kind of the, you know, one of the aha moments there for me is, like, 9 times out of 10,

72
00:07:52.240 --> 00:07:59.640
Steven Elliott: Slow decisions are not a people problem, and they're not even a culture problem. They're really mostly a systems problem.

73
00:07:59.860 --> 00:08:10.699
Steven Elliott: And so what's really interesting about that, in kind of the… what we're facing, from a technology perspective going forward, in a world where we want to give agents more autonomy to make decisions, for example.

74
00:08:10.810 --> 00:08:26.190
Steven Elliott: This concept of how we make decisions in an organizational context is more important than ever, and I think it's going to become more and more of a differentiator for who's the winners and losers, you know, at the end of the day. So I want to… that's what I want to talk to you about today, so…

75
00:08:26.190 --> 00:08:37.990
Steven Elliott: You know, another really interesting thing to think about is, like, most bad outcomes are not because we made bad decisions. You know, in people's personal lives, you could say that for sure, but, like, in organizations.

76
00:08:38.030 --> 00:08:47.270
Steven Elliott: It's usually not because we made a bad decision, it's usually because we didn't commit fully to a good decision, for example, or, you know.

77
00:08:47.270 --> 00:08:58.909
Steven Elliott: I think about all the times I sat in rooms with brilliant people, deeply experienced, deeply committed, like, all the right things in place, and you still… you kind of watch decisions just disappear into the ether, or into the meeting minutes.

78

00:08:58.910 --> 00:09:14.180
Steven Elliott: And so, you know, we had the data, we had the analysis, but… but we weren't really sure whose call it was, or, you know, we did know whose call it was, but that person was waiting on 3 other people for more information. And so, you know, you see those signals in an organization where, you know.

79
00:09:14.180 --> 00:09:28.269
Steven Elliott: we keep asking for more data, for more data, or everyone agrees in the meeting, but then nothing really happens after that, or our decisions keep getting escalated and, like, float up to whoever's willing to take the heat for that decision, right? And so…

80
00:09:28.480 --> 00:09:30.830
Steven Elliott: You know, when you, when you,

81
00:09:31.920 --> 00:09:36.269
Steven Elliott: when you think about that, you know, I also think about this myth, you know, I think about, like.

82
00:09:36.300 --> 00:09:52.660
Steven Elliott: business leaders, like, people we look up to that we think are really decisive leaders, and a good decisive leader is great, right? But I also think we've been sold a bit of a myth that, like, if we just get that leader, then everything else in the company is going to work. And, you know, we're talking large companies at scale, and so…

83
00:09:52.660 --> 00:09:57.830
Steven Elliott: you know, we kind of want this one leader who comes in and just makes it all happen, but kind of what I found just…

84
00:09:57.830 --> 00:10:04.679
Steven Elliott: Just about every time when you pull back the cover on decision-making, you know, the fastest, most decisive organizations,

85
00:10:04.980 --> 00:10:21.129
Steven Elliott: They… it's nice to have the bold individual, and it's probably necessary to have a leader who can make decisions, of course, and leaders who can make decisions, but they've got systems that drive clarity, they've got decision rights, they've got clear processes, they've got accountability, and so, you know, kind of… kind of…

86
00:10:21.230 --> 00:10:39.529
Steven Elliott: you know, I think when I was younger, I used to think people were just born with decisiveness as a trait, and I think in an organizational context, you need a system. It's more of a system, and if the system's broke, it's going to affect everything in the organization in a major way. You're gonna feel it, and so…

87
00:10:39.530 --> 00:10:55.009
Steven Elliott: Let's… let's just quickly talk a little bit about how these systems break, because I think if we can talk about how they break, and then how to start to fix them, it'll be a really interesting parlay into working with agents, and how they make decisions in the organization, so…

88
00:10:55.010 --> 00:11:05.359
Steven Elliott: And I guarantee when we talk about some of these, how these systems break, some of this will probably feel familiar to you in at least one organization you've worked in, probably many or most that you've worked in.

89
00:11:05.360 --> 00:11:20.039
Steven Elliott: But in the Decisive Company, what I tried to do was, like, break down every decision failure I've seen into three categories. And when you look at the failures, just think of them as, like, holes in a leaky bucket, right? So the first one is blurry writes, it's an obvious one, like.

90
00:11:20.040 --> 00:11:37.749
Steven Elliott: you know, a company, maybe they have a racy or a Daisy or some way of making decisions to know who's accountable, but it's in Notion or Confluence, and it's kind of getting dust on it, right? So when people don't know whose call it is, they hedge, right? They CC everyone, they wait. So that's one thing, is Blurry writes.

91
00:11:37.780 --> 00:11:42.309
Steven Elliott: The second one is a… it's just a broken process, and the issue here is, like.

92
00:11:42.500 --> 00:11:52.029
Steven Elliott: it's not that you don't have the data, you're drowning in data. The problem is, the data doesn't flow into the decisions at the right time. So, we have dashboards, we have decks.

93
00:11:52.200 --> 00:12:02.989
Steven Elliott: But maybe no one's reading the decks at the right time, or it's all fragmented, so that's another part of the process that's just broken, is the data flow. And then the third one is the execution gap, and this one…

94
00:12:02.990 --> 00:12:13.700
Steven Elliott: This one's definitely a killer when you see it, but, you know, a decision gets made, but it never truly becomes actionable. So, maybe one person's tracking it, but it's not visible to everybody else.

95
00:12:13.700 --> 00:12:29.239

Steven Elliott: Or maybe we have the outcome, but we're not tracking the metrics to understand if we're moving the needle on that outcome. You see this all the time, and so, you know, the memory of what we've done in the past and what we've learned is just not there, and it creates execution gaps, and so…

96
00:12:29.240 --> 00:12:45.379
Steven Elliott: you know, if… something just to consider is if, you know, if I was thinking about how to improve decisiveness in an organization, just fixing one of these holes in the bucket can help a ton, and if you can fix all three, it's a good way to be on your way to being a more decisive organization, so…

97
00:12:45.380 --> 00:12:56.499
Steven Elliott: This is a really expansive topic, so, like, to put it in a 20-minute talk is a little tricky, so I'm just going to share a few concepts that I really love, that kind of have helped me and kind of helped shape my thinking around

98
00:12:56.500 --> 00:13:11.949
Steven Elliott: decisive companies. So, you know, Chip and Dan Heath wrote this great book about, decisiveness, and, like, their central insight was, our brains are wired to fail at decisions, which kind of took me back a bit, but, like, you know, they were just going through the science of, like, you know, we get narrow, we get attached.

99
00:13:12.040 --> 00:13:18.040
Steven Elliott: We get emotional at the wrong moments, and so they wrote this framework, the RAP framework.

100
00:13:18.050 --> 00:13:33.219
Steven Elliott: And it's a… it's a good corrective. It's a good way to correct some of these… some of these inherent behaviors and some of the psychology, but the point I… the point I wanted to highlight here in an organizational context is, like, if you look at all these steps, they're not mindsets, and they're not

101
00:13:33.220 --> 00:13:43.619
Steven Elliott: you know, your… it's not your character, your qual… you know, it's not who you are, it's… it's things in this… this RAT framework are things that can be designed. They can be interventions that are designed into your process.

102
00:13:43.620 --> 00:13:57.810
Steven Elliott: So, you know, widen your options, reality test, attain distance, set trip wires, like, think about those, like, each one of those is something you can build into how your organization operates, right? And that's really helpful because there's a lot of psychology going on with

103
00:13:57.850 --> 00:14:08.480

Steven Elliott: with decision makings, and what's cool about these frameworks is you can just… or these concepts is you can build them into your system. And then, you know, I really like Annie Duke's writing as well. She's…

104
00:14:08.790 --> 00:14:26.379
Steven Elliott: She was a championship poker player before she came… became a decision scientist, so that makes her extra interesting to me, but, like, the central… the central kind of theme in her work is… is something that often makes leaders pretty uncomfortable, because she talks about how we confuse decisions… good decisions, with good outcomes.

105
00:14:26.380 --> 00:14:38.649
Steven Elliott: So, I think she calls that resulting, and it's something I see all the time. Like, a good way to think about it is, if you punish a team for a bad outcome, or, you know, you're like, why did we have this bad outcome?

106
00:14:38.950 --> 00:14:57.259
Steven Elliott: But it came from a good decision. You've kind of taught them to be risk-averse forever, so if they made a decision, and it was a 80-20 decision, like, and it wasn't that risky, and we took the… maybe we hit the 20% probability, it didn't work out, but it was a good decision-making process, that's not really a failed decision.

107
00:14:57.260 --> 00:15:10.610
Steven Elliott: The outcome didn't work out, but that wasn't a failure. We actually made a good decision, and so I have to be really careful about separating the outcome and the decision itself. And the other thing she talks about that's really good is probabilistic thinking. So…

108
00:15:10.610 --> 00:15:27.500
Steven Elliott: it's, you know, when you think, like, I'm gonna go pitch my idea to the leadership team about what we should do next for a product or a service. She talks about, like, you really need to bring the percentages, right? You know, we're 70% confident that this will work, and that just changes the conversation.

109
00:15:27.580 --> 00:15:38.270
Steven Elliott: Completely. It invites debate, it separates the decision from the outcome, from how you're presenting it, and so, you know, what strikes me the most about the things I'm sharing is, like.

110
00:15:38.360 --> 00:15:53.979
Steven Elliott: this only works if your organization has systems built for it, but you can build systems for these things, and it makes a huge difference. And then, probably one of the most popular ones when people talk about decision science, a lot of it comes from Kaneman. He's, you know, fundamental to a lot of this stuff, tons of writing.

111
00:15:53.980 --> 00:16:05.600
Steven Elliott: That's been really influential, but his side is, like, you need to have different operating modes. He talks about System 1, which is fast, intuitive, automatic, and System 2 is slow, deliberate, analytical.

112
00:16:05.650 --> 00:16:18.849
Steven Elliott: And, you know, his point is, like, neither is wrong, both are necessary. But the failure you see over and over in an organization is, like, applying the wrong system to… to an opportunity or a decision we need to make. And so…

113
00:16:19.060 --> 00:16:33.020
Steven Elliott: All the time, you see, like, either we're putting way too much rigor on something where we should just decide it now, go do a test, figure out if it works, because it's completely reversible, or we go the other route, and we, like, make a fast, gut-level call

114
00:16:33.020 --> 00:16:46.479
Steven Elliott: on something that really did require more structured deliberation. And so, what strikes me is, like, in organizations, I see a lot of organizations, we know this, but, like, think about when you make decisions, how often do you see them being slotted into different systems.

115
00:16:46.480 --> 00:16:56.150
Steven Elliott: And saying, hey, this decision's gonna be made this way, and this decision's gonna be made this way, and, like, we're deliberately thinking about it. You also hear about, like, the door… like, Amazon talked a lot about one-way, two-way doors.

116
00:16:56.150 --> 00:17:09.259
Steven Elliott: Right? You can't… if you go through that door, you can't reverse it, so you treat decisions differently. And so, you do see organizations talking about it, but when I think about, you know, boardrooms I've been in, and, like, product meetings where we're making decisions, you don't see it…

117
00:17:09.260 --> 00:17:21.400
Steven Elliott: you don't see the decisions being framed up to where we have different flows. And so, I think that's another thing that can be very much systematized into an organization. And so, kind of what I love about those three examples, like, Keith is like.

118
00:17:21.440 --> 00:17:28.180
Steven Elliott: he says, design systems better. You know, Duke says, you know, Cal, you know.

119

00:17:28.390 --> 00:17:46.469
Steven Elliott: calibrate better. And then Kahneman's talking about matching the modes to the moment. And so, you know, in the Decisive Company, what I was trying to do was, like, fuse these into a practical framework where, you know, we've got decision rights, so everyone knows whose call it is, we've got decision process, so the right information reaches the right people at the right time.

120
00:17:46.470 --> 00:17:57.889
Steven Elliott: We've got decision execution, so everyone has an owner, or every decision has an owner and a tracker and organizational memory. So, again, just kind of the tip of the iceberg on thinking about

121
00:17:57.890 --> 00:18:06.659
Steven Elliott: how decisions are made in organizations, and, like, how we can be more decisive. And if, you know, if I go just… I'm gonna go into these really quick on those three concepts I just mentioned, but, like.

122
00:18:06.760 --> 00:18:22.689
Steven Elliott: you know, think about… think about decision rights. This is… this is one that's really important when we start getting into a world of agents running around making decisions. Like, who can make the decision? Like, even if we're doing human in the loop with agents, who owns the decision? It's really important, and a lot of organizations

123
00:18:22.690 --> 00:18:29.069
Steven Elliott: We just don't have that systematized and digitized to know who makes what types of decisions. Right.

124
00:18:29.070 --> 00:18:29.600
keith.boswell: Bye.

125
00:18:30.350 --> 00:18:43.880
Steven Elliott: And so, yeah, so there's, you know, I've worked in organizations where, like, there's 3 different people who all thought they owned the same decision, or an organization where nobody knew who owned it. It happens. And so, you know, and those are both equally bad and frustrating, and so…

126
00:18:43.880 --> 00:18:51.649
Steven Elliott: Decision rights have to be explicit, written down, agreed upon. We want that, you know, they have to be tiered because not every decision needs the same architecture.

127
00:18:51.650 --> 00:19:01.190
Steven Elliott: And so, you know, just… those are some important things to think about when we start thinking about 5 years from now, where we want to give agents more autonomy to help us make better decisions.

128
00:19:01.190 --> 00:19:19.929
Steven Elliott: And I'm not saying they're gonna go make the actual decision on every strategic thing. They should be helping inform it. They should be helping us learn from decisions. There's a lot of cool applications here. But I think, you know, decision rights is a pretty important one when you start to think about that world. And then the second one is the process, and so, you know.

129
00:19:20.700 --> 00:19:26.130
Steven Elliott: A decision process architecture gets you from that critical piece of, like.

130
00:19:26.190 --> 00:19:42.929
Steven Elliott: we need to decide something to… we've made a decision. And sometimes it's fuzzy in there, like, did we actually decide? Where is that… does everybody know where that decision was made, and is that, like… you know, one thing I saw in organizations, like, we would have a decision come up, and there'd be, like, 5 pages

131
00:19:42.930 --> 00:19:53.260
Steven Elliott: out in Confluence or Notion or SharePoint, and I wasn't really sure which one actually happened. I'd have to go talk to people to figure out what we actually decided and what's going on.

132
00:19:53.260 --> 00:19:59.990
Steven Elliott: And so, just for a good decision, just basics, it makes sense, but, like, frame the context. What are we actually trying to solve?

133
00:20:00.010 --> 00:20:16.550
Steven Elliott: you know, surface options. We do this wrong a lot, where we just have two options, A or B. Usually need three or more. There's a bunch of psychology behind that one. You define the criteria in advance. This is one we always miss, where it's like, what does success look like before all the politics kick in?

134
00:20:16.550 --> 00:20:22.100
Steven Elliott: And then you make the decision, and then, you know, the fifth one is my pet peeve.

135
00:20:22.100 --> 00:20:33.570
Steven Elliott: It's… we decided something, and we communicate the decision sometimes, but, like, the reasoning behind the decision and why we did this is so important, because by the time it filters down to, like.

136
00:20:33.570 --> 00:20:44.769
Steven Elliott: teams who are doing work, and they're like, hey, we're shifting directions again. Hey, we're pulling this budget. So often, like, the reasoning for

why we did it doesn't get down to that level, and they're just frustrated. They feel like you're just…

137
00:20:44.770 --> 00:21:02.409
Steven Elliott: you're just, you know, you're moving the goalpost on them constantly, and so that why piece is really important for organizational decision making. So, and then, you know, your standard ANDI patterns to notice if you're good at this are like, do we have decision debt building up? Are there a lot of decisions we need to make that we keep punting on?

138
00:21:02.410 --> 00:21:16.030
Steven Elliott: You know, do we have contact starvation? Do we have fandom decisions? I won't go into all that, but, like, the phantom decision one is, like, this universal thing you see in companies where we have a meeting, and heads nod, and then nothing happens. So, you know.

139
00:21:16.030 --> 00:21:30.489
Steven Elliott: But that's probably a good signal, is if you design a process for making decisions, if those fandom decisions, become really hard to do, or almost impossible to do, then you've got a good system. And then this last piece is execution. So the third component is

140
00:21:30.490 --> 00:21:31.460
Steven Elliott: you know.

141
00:21:31.460 --> 00:21:50.700
Steven Elliott: where most frameworks stop. We just… we say, we made the decision, we're done, but the execution piece is where all the learning happens, and so, you know, you can have crystal clear decision rights, this beautiful decision process, and if you don't loop it through execution, we lose so much of the important things around getting better and faster as an organization.

142
00:21:50.700 --> 00:22:09.360
Steven Elliott: Because, you know, obviously decisions aren't going to go execute themselves, they need owner's timelines, but, like, tying that to the process so that we build organizational memory and we learn from decisions is really important. Really, really helpful for, like, getting better and building those learning loops. And, you know, so…

143
00:22:09.530 --> 00:22:13.200
Steven Elliott: really, really kind of what… the way I frame that up is, like, if… when you build…

144
00:22:13.700 --> 00:22:32.030
Steven Elliott: memory into your system. If you can build a way to have organizational memory on the decisions, you won't just make better decisions a day.

It's really interesting what you can do in the future to make better predictions and progressively make better decisions in the future. And that's… that's really what you're after. So, kind of all together, you know.

145
00:22:32.320 --> 00:22:49.590
Steven Elliott: Decision rights make it explicit, decision process gives you the architecture to move from context to choice, in a clean way, and then the execution gives every decision an owner, timeline, success signal, but really what that's all about is… is building memory and learning. And so, when these are all aligned, like.

146
00:22:49.830 --> 00:23:04.219
Steven Elliott: what I've seen is, like, the momentum accelerates in the organization dramatically, and, like, decision velocity improves, and in the end, you get the ability to move fast without losing clarity, which is… which is really what everybody wants in the organization, so…

147
00:23:04.560 --> 00:23:23.950
Steven Elliott: So here's… here's where it gets kind of interesting. So… so how do you do this with agents? So, how does this… how does this scale? Let's say we've got decision rights, and we've got a process, and we're… we're better at execution and tracking that through the loops and learning loops. Like, how does this work in a world where we've got large scale, we've got humans working with agents, and so, kind of.

148
00:23:23.950 --> 00:23:36.240
Steven Elliott: when I think about the decision-making piece, and these pieces are coming together in an interesting way out there, but like, you know, in a fast-evolving area where, you know, agents need context to do their jobs in a large organization.

149
00:23:36.240 --> 00:23:44.039
Steven Elliott: If they can… if they can make decisions, or help us make decisions, in the decision system, it's a very powerful notion, and so…

150
00:23:44.130 --> 00:23:49.660
Steven Elliott: I think what's interesting about today is, like, we've always wanted to improve our decision

151
00:23:49.700 --> 00:24:07.500
Steven Elliott: systems in a company, our decision intelligence, but I think we're going to be forced… if we want to be good with AI and agents, we're going to be forced to focus on this problem now, which is kind of exciting. It's good for the humans, too. Like, humans want the clarity, humans want to be the, you know, for decisions to flow through the organization better as well. It frustrates

152
00:24:07.500 --> 00:24:23.070

Steven Elliott: it frustrates all of us, but for agents, we need context and memory. And so, kind of the… kind of the pieces you have here, if you think about it, like, whatever your decision system is, somewhere in there, you know, we need this… we need this ontology of, like, how do we think we operate? And then we need, like, a knowledge graph

153
00:24:23.070 --> 00:24:38.070
Steven Elliott: and memory to say, here's how we're actually operating. And those two things together are really powerful, that… the duality between, here's how we think we work, that's our ontology, and here's what's actually happening day-to-day, that's, you know, through a graph.

154
00:24:38.130 --> 00:24:52.429
Steven Elliott: And then, you know, if we can take that graph and extend it with MCP in a concept text engine, now agents can access that information and go get their context for, like, oh, I understand how, you know, people work, time, money, dollars, decisions flow through the organization.

155
00:24:52.430 --> 00:25:00.290
Steven Elliott: And you can just think about giving an agent context, and you know, some of this is a little more in the future. I know a lot of orgs aren't doing this fully, but

156
00:25:00.320 --> 00:25:15.190
Steven Elliott: in the future, they're going to need that context, and we humans need that context, so it's a really cool, a lot of concepts coming together that are really interesting, at least to me. And so, you know, when we started DogWork, we were kind of thinking about the question like this, like, if we were going to build

157
00:25:15.330 --> 00:25:28.060
Steven Elliott: a really decisive company into software, into a platform, what would it look like? And it definitely wasn't like, don't build another project management tool or an OKR tracker. It has to be something that, like, changes how decisions flow

158
00:25:28.060 --> 00:25:36.319
Steven Elliott: through an organization. So, what we kind of came up with was, like, the strategy to execution bridge is always key. Like, we still have to connect the dots.

159
00:25:36.340 --> 00:25:51.680
Steven Elliott: What's cool is, like, we have a lot better ways and technology of doing that, and agents can go fill in a ton of gaps around context and data, and do, you know, a lot of it comes down to, like, getting rid of data entry, and getting rid of timesheets, and getting rid of some of the stuff that we were just doing

160
00:25:51.680 --> 00:26:06.580

Steven Elliott: to try to connect the dots that we don't have to do anymore. But we need those dots connected, and we need it connected, you know, now, not later. So when we're making the decision, we have a good… a good frame for the organization. And then the memory piece is key. We need, you know, AI needs

161
00:26:06.580 --> 00:26:11.439
Steven Elliott: to get smarter every time it makes a decision, so do we. Agents need to surface

162
00:26:11.650 --> 00:26:30.070
Steven Elliott: things and have intelligence so that they can do cool things like surface risk before it's visible to us, right? And then what that up… if we have those systems, that previous slide I showed you, it really does open up these opportunities for better forward-looking operational intelligence, right? So most platforms we have, most…

163
00:26:30.070 --> 00:26:33.860
Steven Elliott: Most software we have to help us drive the data for making decisions.

164
00:26:33.860 --> 00:26:48.769
Steven Elliott: It's telling us what happened before, but what leaders are really interested in is trying to figure out what's going to happen next, and so putting that all together creates this really interesting opportunity to advance how we make decisions quite a bit. And so, you know.

165
00:26:49.020 --> 00:27:05.840
Steven Elliott: just to sum it all up simply, like, we need to focus on the data, the operating model, the decision systems for decisiveness, and what we get from that is improved context, better organizational memory, and better learning from prior decisions. So,

166
00:27:06.070 --> 00:27:24.089
Steven Elliott: Yeah, so that's kind of how to frame it up. So, just a couple things to leave you with, looking at the clock. Yeah, I think I'm good. Okay, so, just to make this concrete, because I didn't want to just make this all theory, so if you were thinking about this, like, you know, where to start, like, if you're in an organization where there aren't a lot of these things I've talked about in place for making decisions.

167
00:27:24.090 --> 00:27:28.019
Steven Elliott: kind of one thing I did in an organization that I saw work really well.

168
00:27:28.020 --> 00:27:37.259

Steven Elliott: kind of the first month, we were just like, we're just gonna audit decisions. We're just gonna, like, say where recurring decisions happen, where are we seeing fandom decisions, like, just having some conversation about

169
00:27:37.600 --> 00:27:55.850
Steven Elliott: what's the lay of the land? And then the second month, we basically put in a tiering system I was telling you about. We did one-way door, two-way door decisions, just simple. I like a three-tiered system, which is… which is, you know, Tier 1 can go really fast, Tier 2 is medium rigor, Tier 3 is, like, we need executive leadership buy-in.

170
00:27:55.850 --> 00:28:07.369
Steven Elliott: But whatever you put in, like, putting in some sort of way where we don't treat every decision the same way, because that's a killer. And then the third month, we started, like, actually capturing decisions and their outcomes over time.

171
00:28:07.370 --> 00:28:20.450
Steven Elliott: And we started setting… this took more than a month, but then we started setting up reviews after the decisions were made, and it's, you know, it's not a strategy review, it's not a performance review, it's a little different, it's a decision review. So, like, what did we decide? What happened?

172
00:28:20.480 --> 00:28:27.370
Steven Elliott: What should we have done differently? That was very powerful, and it didn't take a ton of effort, and we started, you know.

173
00:28:27.370 --> 00:28:41.410
Steven Elliott: I think after we ran that meeting 3 or 4 times, I started to feel, and I think a lot of people started to feel like we're actually learning here. Like, we're gonna get better at making decisions, so it was a pretty cool moment, but it's not that expensive of an experiment to,

174
00:28:41.410 --> 00:28:45.310
Steven Elliott: to run. And then just this last slide here, just, you know.

175
00:28:45.560 --> 00:29:04.309
Steven Elliott: leave you with a question that I often think about, like, if I wasn't in the room, so if you're a leader, and you're part of the decision-making process, if you're not there, and the decision had to be made without you, would your organization know how to make it? So you're going to, you know, you're going to Tahiti for, you know, 6 weeks.

176
00:29:04.420 --> 00:29:06.580
Steven Elliott: And, you know.

177

00:29:06.950 --> 00:29:23.540
Steven Elliott: just think about that, like, would they make the right call? Would they know how to make the call? Would they know whose call it is? Would they have the context they need? Would they know what the process was, right? And if you… a lot of times when you talk to leaders, they'll say, no, I don't think that… I think they would need for me to come back and weigh in.

178
00:29:23.540 --> 00:29:37.990
Steven Elliott: And so, you know, that's just a good signal to think about. Like, obviously there's some decisions that need to… like, if you have that, you know, that tiered system, obviously, Tier 3 always needs to go through the leader, so if you're on vacation, you're going to get called back. But the Tier 1 and Tier 2,

179
00:29:38.060 --> 00:29:53.340
Steven Elliott: Should be, if they're well aligned with the organizational's objectives, and we know how to run experiments and make decisions, we can start to, we can start to assign those out to agents and people, and have a system that really flows and doesn't bottleneck.

180
00:29:53.340 --> 00:30:01.200
Steven Elliott: on, on decision-making, you know, with good velocity. So… so that's it for me today, and,

181
00:30:01.200 --> 00:30:09.450
Steven Elliott: you know, I'd say, just, if you remember anything, I would say just, just my main point with this talk was just, you know, decisiveness is design.

182
00:30:09.490 --> 00:30:23.159
Steven Elliott: And that's what's kind of cool about it, like, it's… it's, you know, it doesn't just happen, but it's a design thing, so we can design better decisiveness into our decision-making process, into the systems we use to make the decisions.

183
00:30:23.160 --> 00:30:37.100
Steven Elliott: And so, you know, for me, you know, you can obviously see what's coming with AI, like, the companies that'll win over the next few years are gonna be the ones, you know, not just with the smartest people or the most agents, it's gonna be the ones that figure out how to align those things, like, align smart people with agents.

184
00:30:37.100 --> 00:30:46.729
Steven Elliott: in a decision-making process that's aligned and accountable, you get that all working together, and that's going to be a really powerful stew. So, thank you very much. Any questions?

185
00:30:47.170 --> 00:30:49.379
Steven Elliott: I don't know if we're taking questions.

186
00:30:50.100 --> 00:30:56.870
David Mantica: Absolutely would love some questions, but got some great comments in chat, the whole premise behind the book.

187
00:30:57.180 --> 00:31:06.660
David Mantica: the Abershaus book, your team should be able to operate without you, because your overarching intent is to understand, and the teams are empowered, right? That's pretty cool. Corey made that comment.

188
00:31:06.780 --> 00:31:12.950
David Mantica: Love the foundational element of this, Steven, this is awesome. I mean, if you're going to go into Gentex.

189
00:31:13.110 --> 00:31:17.420
David Mantica: I mean, some great foundational elements here to build off of.

190
00:31:17.540 --> 00:31:21.929
David Mantica: But any questions, thoughts? You can open up your mic if you want to as well.

191
00:31:27.860 --> 00:31:33.470
Corey King: Well, I'd be remiss if I didn't say, hey Steve, Corey King from your old Jira line team. Great to see you.

192
00:31:33.470 --> 00:31:34.529
Steven Elliott: Good to see you.

193
00:31:37.100 --> 00:31:41.099
Jess Wolfe: I might be remiss not to say the same thing. Jess Wolf from your old Jira line team.

194
00:31:41.100 --> 00:31:44.850
Corey King: Hey, Jess! So good to see you, too!

195
00:31:46.150 --> 00:31:48.300
Steven Elliott: Alright, great. Thanks, everybody. Appreciate it.

196
00:31:48.630 --> 00:31:49.510
David Gijsbers: Thanks, Steve.

197
00:31:49.880 --> 00:31:50.550

David Mantica: Yeah.

198
00:31:52.460 --> 00:31:56.760
David Gijsbers: So, so, when we were designing the,

199
00:31:57.220 --> 00:32:08.979
David Gijsbers: order or the flow of… of, how we were going to have the topics in this conference, you know, we wanted to start at the strategic level, so I hope that,

200
00:32:09.090 --> 00:32:13.569
David Gijsbers: Steve's presentation helped us to understand how we can create a competitive advantage

201
00:32:13.700 --> 00:32:25.290
David Gijsbers: through creating decisive systems. Next, we wanted to bring it into the real world and take a look at, some of the ways that people have been actually implementing

202
00:32:25.520 --> 00:32:37.880
David Gijsbers: AI and software engineering, and I reached out to my good friend Jess Wolf, and she was more than happy to jump in and gave us a great case study. So, why don't we hand it over to Jess for the next presentation?

203
00:32:40.200 --> 00:32:41.610
David Gijsbers: You're on mute, Jess.

204
00:32:43.730 --> 00:32:45.390
Jess Wolfe: Can you hear me now?

205
00:32:46.230 --> 00:32:47.360
David Gijsbers: Yes, you sound great.

The Unsexy Truth About AI Success: Leadership, Hygiene, and Human Foundations

206
00:32:47.620 --> 00:33:03.770
Jess Wolfe: Awesome. Welcome, everyone! So, I'm Jess, I'm here with John, and we are here to talk about the unsexy foundations, leadership, hygiene, and the human element that actually determines if AI succeeds.

207
00:33:03.770 --> 00:33:15.170
Jess Wolfe: or fails in an engineering organization. Now, Steve really gave us a good foundation for how do you think about these things, and the context, and all of that.

208

00:33:15.750 --> 00:33:34.059
Jess Wolfe: And what I wanted to do, when Dave introduced me, or asked me to do this talk, I thought a little bit about what am I seeing amongst my customers at Swarmia? And one customer really stood out, John Harbinson from PrescriberPoint. Steve mentioned.

209
00:33:34.060 --> 00:33:50.189
Jess Wolfe: You know, the future is that there's organizations who have this context in place for the agents. John is a great example of an organization that has that. They have not coded, or developers have not written code in over a year, I want to say. John will tell his story.

210
00:33:50.190 --> 00:33:51.879
John Harbison: Getting about that point, yeah.

211
00:33:51.880 --> 00:34:07.989
Jess Wolfe: Yes. And it's all agent-driven. So today, we're not here to talk about the latest LLM, or how to write better prompts. We're here to talk about the unsexy foundations that really determine whether AI can make your team 10x better, or 10x more chaotic.

212
00:34:08.179 --> 00:34:10.560
Jess Wolfe: And this all started

213
00:34:10.870 --> 00:34:18.769
Jess Wolfe: with a specific question. John asked, how do you measure developer productivity now that AI is in the picture?

214
00:34:18.940 --> 00:34:19.960
Jess Wolfe: So…

215
00:34:19.980 --> 00:34:43.829
Jess Wolfe: like I said, John is doing something very different here, and one of the things that we realized when we came into it was we were looking at traditional metrics, you know, your Scrum, your Agile metrics, and story points weren't going to solve this for us. John realized that before we could measure the new world of agents creating code, we had to understand what AI was actually doing in our process.

216
00:34:47.199 --> 00:34:47.980
Jess Wolfe: John.

217
00:34:47.989 --> 00:35:06.139
John Harbison: Yeah. So, when we started looking at this, we started with AI and getting our developers integrated with AI very early on. You know, we started with

the web experience with ChatGPT and them using it as questions, bringing it into their IDEs, and

218
00:35:06.179 --> 00:35:22.279
John Harbison: moving forward, we started giving them, you know, when Claude Code came out and things like that, giving them access to that, and then moving to this unattended world we'll talk about. But one of the things that we noticed very early on in this process, and Swarmia helped us get there when we started looking at our measures, is

219
00:35:22.279 --> 00:35:27.039
John Harbison: You know, the AI acceleration is a huge catalyst and an engine.

220
00:35:27.039 --> 00:35:38.459
John Harbison: And if it… the old garbage in, garbage out, or the Captain America analogy from the movies, it's like, you know, if you start with something that's bad and chaotic on the inside, you're gonna get that 10x worse on the outside.

221
00:35:38.459 --> 00:35:47.059
John Harbison: Same thing when you start with something, it's a strong foundation, a well-formed process, AI will help build upon that, and you'll have an amazing outcome.

222
00:35:47.099 --> 00:35:49.489
John Harbison: And that's something that people who are

223
00:35:49.579 --> 00:36:02.729
John Harbison: transitioning into the AI world and really accelerating their development engines, they're noticing is like, oh, I, you know, my Scrum process, or my project management process, or things like that are not working right.

224
00:36:02.729 --> 00:36:10.739
John Harbison: So, you need to really be able to get in and understand foundationally how you're running your organization and how you're running your team.

225
00:36:10.769 --> 00:36:16.149
John Harbison: And how they're being measured and working. So, we move on, Jess?

226
00:36:17.649 --> 00:36:18.589
John Harbison: Thank you.

227
00:36:19.249 --> 00:36:27.929

John Harbison: So, you know, I kind of looked at this, you know, and we were trying to find an analogy, and this was the one, the one that kind of came to mind to me, is like, when I was…

228
00:36:28.059 --> 00:36:31.029
John Harbison: Running the engineering org and monitoring them.

229
00:36:31.089 --> 00:36:49.239
John Harbison: before we had all this AI throughput, you know, it's kind of like driving a Camry, right? You look at your tire pressures, or your gas, and things like that, but most of it is, like, you don't see anything until an alert's going off, right? It's a very simplistic, streamlined experience. When you're running at the velocity of AI,

230
00:36:49.239 --> 00:37:00.869
John Harbison: you need to have a lot more data, and that doesn't necessarily mean that you have to consume that data. This is where tools and AI can help you with that data feed.

231
00:37:01.159 --> 00:37:06.759
John Harbison: So, let's look like what it looks like in terms of metrics. Jess, you want to pick up there?

232
00:37:07.390 --> 00:37:21.919
Jess Wolfe: Absolutely. So, this table represents the shift we saw with PrescriberPoint, moving from Camry on that traditional side to IndyCar, if you will, on the AI side.

233
00:37:22.430 --> 00:37:38.129
Jess Wolfe: when… what really happened was moving from Camry metrics, like sprint burndowns and story points, to IndyCar metrics, like real-time PR review, latency, and investment balance, it really helped them take an AI-first world book, and

234
00:37:38.160 --> 00:37:43.670
Jess Wolfe: it helped us realize some things, right? That we don't necessarily care about story points anymore.

235
00:37:43.690 --> 00:38:06.049
Jess Wolfe: We don't care about lines of code anymore. We do care about rework. We care if AI is pushing 3,000 lines of code, for sure, but we need to understand the context around that. Was it a big deployment, or was it that's happening in DevOps, or was it something that is in engineering, and should we split that? So really looking at the context of these metrics is really how we're making the shift.

236
00:38:06.050 --> 00:38:09.850

Jess Wolfe: But one thing I want to Point out here is that

237
00:38:10.760 --> 00:38:30.519
Jess Wolfe: Unlike a car, what we're seeing here is the metrics on the right that are really helping us drive AI forward, and I would say even engineering excellence forward without AI right now, are eventually going to move to the left. There will be faster and new things, and we always need to be in more of a learning environment.

238
00:38:30.620 --> 00:38:31.970
Jess Wolfe: This shifts.

239
00:38:32.120 --> 00:38:37.359
Jess Wolfe: In measurement is only possible if you change the way your team is structured.

240
00:38:38.250 --> 00:38:57.320
John Harbison: Yeah, so this is foundationally one of the big changes we had to look at. You know, traditionally, we had a product department that had the voice of the business, they had the strategy, they had the idea, we had a design group that was helping realize and build the specifications, and we had the engineers who were there for execution.

241
00:38:57.440 --> 00:39:13.259
John Harbison: And what we realized is the engineers were missing a huge piece of context. They weren't involved in the design, or they were brought in late. They weren't involved in the strategy, or if they were, it was like an engineering leadership that was brought into the strategy.

242
00:39:13.480 --> 00:39:28.609
John Harbison: So, we created and changed our workflow so that now, when we're doing our product ideation, when we're doing our envisioning, when we're doing design, we set all three stakeholders down at the same time. It's usually the engineer who's doing the work.

243
00:39:28.610 --> 00:39:36.650
John Harbison: And is going to be in charge of that task is the one at the table, not their boss or me, doing that and having to read out to them.

244
00:39:36.840 --> 00:39:48.620
John Harbison: And they're actually contributing. They're not just there listening. They're also there participating in the discussion. They've got domain knowledge, and they're adding to the strategy session.

245
00:39:48.830 --> 00:40:01.820

John Harbison: So this has also changed the way that we train and enrich our engineers as well. We're not just evaluating them on their technical aspects, we're also evaluating on their business and their design aspects as well.

246
00:40:02.010 --> 00:40:16.450
John Harbison: And then this also allowed us to, improve our results with AI, because now, when the engineer is working with the AI agent, they have the complete picture, or the full context in their head.

247
00:40:16.520 --> 00:40:22.429
John Harbison: So they can now talk and collaborate with the AI agent without just having an engineering perspective.

248
00:40:23.180 --> 00:40:28.700
John Harbison: But we also don't let the agents run wild. We have different… no, you were…

249
00:40:30.360 --> 00:40:49.749
John Harbison: So we have multiple ways we've enabled our engineering department to work, and this is constantly changing as new technology comes out and things become available to us. So, we have… Model 1 is like, you know, we have Cursor or another IDE that has an AI capability built into it, and it's a very interactive session.

250
00:40:49.750 --> 00:40:55.269
John Harbison: Looking at the code and working with the engineer. We have the semi-unattended models.

251
00:40:55.650 --> 00:41:03.050
John Harbison: So this is more along the lines of, like, running clawed code in a terminal, or codecs, or any of the other tools that are out there.

252
00:41:03.140 --> 00:41:19.960
John Harbison: And then we have the ability to do a fully hands-off approach, where the AI is given a ticket, or given a story, or given a piece of context, and said, go do this work and come back to me when you have something to show me, or you need a human in the loop.

253
00:41:20.040 --> 00:41:32.189
John Harbison: Because you have a blocking problem. And we're not restrictive with the way that we work. We allow the engineer to apply whichever model makes sense for the work that they're doing.

254
00:41:32.190 --> 00:41:42.139
John Harbison: So if they're more like, this is a Model 3 kind of activity, or this is a Model 2 activity, they have the autonomy to pick that and act accordingly.

255
00:41:42.880 --> 00:41:54.859
John Harbison: And then, you know, I think that's, yeah, that's the right one. And then, you know, this is what we call our secret sauce, so kind of moving into, what that's changed for us is.

256
00:41:55.030 --> 00:42:09.040
John Harbison: the engineers are using AI to write the code. Like, even doing as much as, like, even basic stuff like setting up ENV files are being done through the agent. Not always the best use of tokens, but hey, you know, I love what they're doing.

257
00:42:09.340 --> 00:42:28.560
John Harbison: They're becoming UAT experts as well, because they have the context in their head, so they can look at the output, the build, and they can actually interrogate it and work through it. They're not having to wait for QA or product or anyone else to come into the room.

258
00:42:28.560 --> 00:42:31.339
John Harbison: And prove out the work that's being done.

259
00:42:31.440 --> 00:42:40.819
John Harbison: And then this has also increased our autonomy, because the engineers, and really product and design as well, because they're picking up the technical side,

260
00:42:40.990 --> 00:42:57.569
John Harbison: they're getting all of this context in their head now, so they're really involved with the business. They understand our strategy, they understand go-to-market. You know, are they experts at all of it? No, of course not, but they have all this context now, they… it's asking really valuable questions.

261
00:42:57.570 --> 00:43:04.790
John Harbison: They're able to really take on things from product and from business and have that, discussion.

262
00:43:04.910 --> 00:43:12.700
John Harbison: And then, one of the things that we also realized as well, and this is kind of like getting into some stuff we'll go on here in a second, is

263
00:43:13.840 --> 00:43:23.029
John Harbison: when we first started, and especially with the way AI was, we were all, like, we're prompt engineers now. We're writing really long prompts.

264
00:43:23.030 --> 00:43:37.209

John Harbison: We put a ton of context in those prompts. We were treating it like, you know, a toddler, where, you know, you're telling them, don't touch the stove, it's hot, right? Well, what do toddlers do, right? Until they have the practical experience.

265
00:43:37.490 --> 00:43:46.710
John Harbison: what we learned, especially with the state that AI is in now, is we don't have to go crazy on a very large context.

266
00:43:46.840 --> 00:44:04.239
John Harbison: You give tools, skills, and you constrain with hooks so that AI can go find the information it needs, and have that domain knowledge, and then also be like, here's the guardrails to operate in, don't go outside these boundaries. So that was one.

267
00:44:04.240 --> 00:44:12.499
David Mantica: A couple questions, John. A couple questions, John and Jess, real quick, is, what are your current percentages of usage through the different models?

268
00:44:13.430 --> 00:44:18.959
John Harbison: We don't track it by the model, we just track a percentage of AI work overall.

269
00:44:19.090 --> 00:44:24.630
John Harbison: And we're near 100% of all work is being done with one of the models being used.

270
00:44:24.980 --> 00:44:30.400
David Mantica: And then the second question, did the story writers need specific training, and how do you share the learning?

271
00:44:31.120 --> 00:44:40.419
John Harbison: The story writers didn't need specific training to write the stories. Where we got into with it was exposing them to our domain.

272
00:44:40.560 --> 00:44:51.700
John Harbison: And that was something that was more of a bigger challenge, was because we're in healthcare, and we're in a specific section of healthcare, not everyone, especially from the engineering department, even knew

273
00:44:51.700 --> 00:45:02.489
John Harbison: how some of our flows and our processes have worked or applied in whole. They were, you know… I think it was the old Apple story, right? When they were first doing the iPhone, like, everybody built a certain part and didn't know what the hole did.

274
00:45:02.610 --> 00:45:20.509
John Harbison: we felt like there was a lot of that going on. So, a lot of the trainings that we worked through were actually in sharing the business domain, the business knowledge, getting them into the product space, learning, like, how to interpret user research results, and watch a panel, and those kind of things. So they were…

275
00:45:20.510 --> 00:45:23.179
John Harbison: New areas that they were going into.

276
00:45:23.610 --> 00:45:24.429
John Harbison: Thank you for.

277
00:45:24.430 --> 00:45:27.040
David Mantica: Thanks for that. Appreciate both chatting going.

278
00:45:27.710 --> 00:45:37.119
Jess Wolfe: Yeah, it's actually… it was a good segue into what we're talking about next. There's one more question, John. Does it mean we are moving from product-centric to a engineer-centric approach?

279
00:45:37.600 --> 00:45:55.020
John Harbison: That is a good question. So that actually came up when I was talking with my project manager the other day, and she was indicating that, like, those that have a command of AI tooling and an engineering background are at an extreme advantage now.

280
00:45:55.170 --> 00:45:59.430
John Harbison: And I kind of… Yes and no, right? Like.

281
00:45:59.570 --> 00:46:10.759
John Harbison: as an engineer, I can instruct AI to help me with a marketing campaign and putting out emails and trying to capture, but I'm not an expert at it, I'm not a marketer.

282
00:46:10.760 --> 00:46:22.299
John Harbison: Right? So, I think that's… that's more of the shift, is it's not necessarily that we're moving towards an engineer-centric approach, it's that everyone's having to now become technical.

283
00:46:22.590 --> 00:46:34.710

John Harbison: in order to use these tools. Like, I've got people learning Python that would have never considered it before to do their job. And that's… that's really the change for me, is we're changing the way people work.

284
00:46:36.800 --> 00:46:43.020
John Harbison: So, this kind of actually goes into the org's shift. So, good, good question, right time.

285
00:46:43.280 --> 00:46:56.629
John Harbison: So, we talked about, you know, the business and tech domain mastery, and this is requiring that people pick up more skills, more tools, more techniques in order to leverage all of this power that's at their fingertips.

286
00:46:56.810 --> 00:47:02.949
John Harbison: We just put an initiative out org-wide where we got everyone into Claude Co-work, and

287
00:47:02.990 --> 00:47:17.209
John Harbison: talk them, like, even basic skills of, like, here's how you open a terminal, here's how you install VS Code, and we're talking about people that, like, just pick up the phone all day and call people. They're having to learn these skills now to take advantage of all of this capability.

288
00:47:18.450 --> 00:47:36.410
John Harbison: org investing context, this was huge for us, this was kind of what I was alluding to. We have spent so much time putting on training sessions, lunch and learns, one-on-ones, familiarizing everyone in the company, you know, primarily focused on engineering, but we're open up to everyone that wants to attend.

289
00:47:36.770 --> 00:47:55.069
John Harbison: teaching them the business and the product and the design side so they have that context in the head. And again, we're not expecting them to replace those functions, we're not expecting them to become masters of those functions. It's giving them enough context to participate in the discussion and be curious for the knowledge so that they can

290
00:47:55.070 --> 00:47:58.970
John Harbison: Sit at the table and have that discussion, and not just a listening session.

291
00:47:59.840 --> 00:48:15.560
Jess Wolfe: Real quick to add to that one, too, to Sasan's point, that is still the same as really being product and customer centric, right? We really need to understand the context, so the same things that we were applying before AI was introduced really still apply now. Go ahead, John.

292
00:48:16.000 --> 00:48:27.609
John Harbison: Yep. And then, you know, the manager coach, like, this is huge for us. I have such an advantage where I work, because my CTO is extremely technical, extremely invested in this.

293
00:48:27.610 --> 00:48:47.360
John Harbison: And the leadership team is the same way. You know, my CTO is pushing code to branches and releasing PRs and doing features. I'm doing the same thing, and it's really changed that now to where it doesn't force us all to be contributors, but we can be. We're enabled to be. And we also like to lead, like.

294
00:48:47.360 --> 00:48:59.420
John Harbison: You know, we will put out good examples of what is good and how to work. And instead of it just being, oh, we just closed a ticket, it's now, here's a reference, let's have a lunch and learn, let's have a workshop.

295
00:48:59.660 --> 00:49:10.709
John Harbison: let's talk about why this is a reference and how you can apply it to your work. So, it's really, pivoting. So, Jess is going to show us, some of the metrics that we've been able to capture

296
00:49:10.840 --> 00:49:14.250
John Harbison: That, that, let me hit that one question really quick.

297
00:49:14.250 --> 00:49:14.930
Jess Wolfe: Yeah.

298
00:49:14.930 --> 00:49:18.930
John Harbison: That just came in. So this is really interesting. So, we were very…

299
00:49:19.710 --> 00:49:22.360
John Harbison: Interesting, about a year and a half ago.

300
00:49:22.620 --> 00:49:31.670
John Harbison: We… so we still, to this day, we're continuous delivery to production, and you'll see that in our metrics here when we start showing a couple of these things.

301
00:49:31.720 --> 00:49:48.929
John Harbison: We were very just in time in the beginning, and we kind of evolved for a hot minute into Scrum, and then we went in to add, Kanban and other things, and now we're kind of… we're at this point, I don't want to spoil it too much, but it's,

302

00:49:48.930 --> 00:50:00.930
John Harbison: We're getting… we're testing workflows of, like, what is it like when we just create a knowledge document for work to do, and put it on a roadmap so we know when we're gonna do it?

303
00:50:01.100 --> 00:50:16.350
John Harbison: And then the rest of it is the engineer working that and adding context and tasks in their local environment, and not having to have a big platform or a piece of software or a process or a methodology. We focus on

304
00:50:16.490 --> 00:50:25.550
John Harbison: feature delivery. The business needs a certain amount of features, they need those features at a certain amount of time, and are we delivering those features when the business wanted them?

305
00:50:26.710 --> 00:50:29.489
John Harbison: So, Jess, you want to hit metrics really quick?

306
00:50:29.630 --> 00:50:43.289
Jess Wolfe: Absolutely. So, what does this look like in practice? Well, if we look at AI impact for John's organization in the last 30 days, one of the things that we mentioned is that John's team is not

307
00:50:43.290 --> 00:51:06.379
Jess Wolfe: writing code with their fingers anymore. What you're seeing here is that, the way Swarmia works is that we integrate to the Enterprise license. We don't integrate to the individual licenses yet. That potentially is coming. But what you'll see here is that there's work with no AI, which is the stuff that they're doing at the side of their desk, compared to the licenses. So now we can look at the impact of the different tools.

308
00:51:06.380 --> 00:51:14.189
Jess Wolfe: against each other. Now, I group this by any AI. We can differentiate between Claude, and all of that.

309
00:51:14.390 --> 00:51:30.820
Jess Wolfe: But in here, you can see, what it's doing with their cycle time, that they're able to get a PR through in 3.3 days. With no AI, it's 35 hours, but that's, again, not no AI, that's what's on the side of the desk.

310
00:51:30.940 --> 00:51:43.320
Jess Wolfe: John, do you want to add more here to give some context around how you got to this place, and why, for instance, the batch size here, is maybe less of a concern for you these days?

311

00:51:43.790 --> 00:51:53.439
John Harbison: Yeah, so one of the things that we saw was interesting is, like, you know, I'll even talk about, like, last week. Last week, we had to spin up a decent-sized

312
00:51:53.440 --> 00:52:05.790
John Harbison: Microservice, which was… if anyone's familiar with health exchanges in the healthcare space, it's basically, like, a central point to push your entire medical record in and out of. We did that in, like, 3 days.

313
00:52:06.020 --> 00:52:16.909
John Harbison: And it was all green… thankfully, it was greenfield, that helps. But what we're also realizing when we're starting to use our measurements and look at these tools is

314
00:52:16.980 --> 00:52:36.119
John Harbison: it's so context and subject aware. Like, if I'm doing a big greenfield push, and I'm doing, you know, a flagship epic-level event, I'm expecting large batch sizes. I'm expecting a little bit larger cycle time on that, versus if I'm just doing iterative development.

315
00:52:36.120 --> 00:52:46.139
John Harbison: or I'm fixing a bug. Those numbers are going to be significantly less. And one of the things that changed, which we'll see in one of the other slides here when we start looking at some volume numbers, is because we've

316
00:52:46.190 --> 00:52:48.650
John Harbison: X'd our volume so much.

317
00:52:49.310 --> 00:52:57.689
John Harbison: This gets lost in the weeds now, because the averages just don't make sense when you see the amount of volume going through the pipe with the team size.

318
00:52:57.980 --> 00:52:58.820
Jess Wolfe: Hmm.

319
00:52:59.380 --> 00:53:07.430
Jess Wolfe: Absolutely. And when we're looking at this, when we're thinking about what can the agents do, what can AI do.

320
00:53:07.430 --> 00:53:19.980
Jess Wolfe: they're still the same metrics, right? Are we getting to the first time to review faster? How long are we in review? Are we getting the PR to move through the cycle faster? But it didn't just

321
00:53:20.180 --> 00:53:30.020
Jess Wolfe: start that way. If we look at the last year of AI impact data, you can see that it was a journey. And John, do you want to talk a little bit about this journey?

322
00:53:30.020 --> 00:53:48.309
John Harbison: Yeah, so we've tried all the tools, well, I mean, you know, they're changing all the time, right? We started with ChatGPT, we went through a co-pilot run, and then we've eventually, you know, our current state is focused around Claude. And, you know, this is one of the things Jess mentioned with the no AI, is

323
00:53:48.370 --> 00:53:52.849
John Harbison: We have a mix at prescriber Point between we have our team plan.

324
00:53:52.850 --> 00:54:11.439
John Harbison: Which is the orange bar that you're seeing here. And then we also have max personal plans that we give to most of our engineering team, just because it's more cost-efficient. And we… that shows up as the no AI in here. So, if you want to see the AI impact, you kind of take the gray bar and the orange bar and add them together.

325
00:54:12.290 --> 00:54:20.549
John Harbison: we may have, like, a DevOps ticket, or something like that, where it was just, like, refresh the CDN, or something that was on a CLI that…

326
00:54:20.770 --> 00:54:26.130
John Harbison: may have just been a real no-AI use case, but they're few and far between.

327
00:54:26.130 --> 00:54:35.500
Jess Wolfe: And I see a whole bunch of great questions in chat. Let's hold them till we get through the rest of these metrics, because I think it'll be relevant, and some of what we'll talk about, I think, answers the questions.

328
00:54:36.450 --> 00:54:43.380
Jess Wolfe: Alright, so now let's take a look at, you know, we looked at where we are now, but we also looked at the journey.

329
00:54:43.390 --> 00:55:01.279
Jess Wolfe: And it started a year ago, right? So, John, where you were a year ago, we can see that your cycle time didn't really change much, right? But what we can see changed dramatically, and we'll look at the actual active contributors, too, what you did with the amount of people that are actually on your team.

330

00:55:01.280 --> 00:55:06.749
Jess Wolfe: You went from 42.7 PRs per week.

331
00:55:07.250 --> 00:55:07.700
John Harbison: Thanks.

332
00:55:07.700 --> 00:55:11.660
Jess Wolfe: All the way up to, in the last 30 days, 113.

333
00:55:11.660 --> 00:55:21.410
John Harbison: Yeah, and one thing I'll note, too, which the cycle time doesn't show on this slide, is the baseline cycle time was

334
00:55:21.510 --> 00:55:26.010
John Harbison: From us looking at, like, task and subtask granularity.

335
00:55:26.130 --> 00:55:28.950
John Harbison: The now cycle time is looking at Epic.

336
00:55:30.860 --> 00:55:51.790
Jess Wolfe: That's actually a really good point. So, one of the things that, when we first started, John's team had a rule about, every story has one PR, right? So you do the one-to-one match. And what they're realizing, and I think is interesting for folks like Atlassian, is that it seems like

337
00:55:51.790 --> 00:55:58.779
Jess Wolfe: The work visibility at the lower level isn't needed as much anymore when you have

338
00:55:58.780 --> 00:56:02.579
Jess Wolfe: AI in the picture, and potentially not at all.

339
00:56:02.970 --> 00:56:09.240
John Harbison: Yep, I don't get asked by the product team about a subtask. I get asked about an epic.

340
00:56:10.960 --> 00:56:23.490
Jess Wolfe: Yes, so there is a question there, how relevant are Agile methodologies in your company? If Agile is to inspect, adapt, iterate, and learn from what you're doing, I'd say they're still relevant, and they have to change.

341
00:56:24.990 --> 00:56:42.590

Jess Wolfe: with that said, the amount of PRs in progress has almost doubled, too. So you're able to do a whole lot more with less, keeping about the same amount of people, which is significant. Do you want to talk a little bit about what that felt like for your team?

342
00:56:43.240 --> 00:56:47.879
John Harbison: If you've ever heard the analogy, you know, with the deer in the headlights, like.

343
00:56:48.210 --> 00:56:59.310
John Harbison: This has been a massive learning experience for us, you know, and I'm sure with everybody, right? Like, we didn't have AI years ago. We're all going through this together, and we're all on, you know, hopefully

344
00:56:59.810 --> 00:57:09.509
John Harbison: having a positive experience with it. There was a lot of start-stop as we were trying to, like, listen to experts, figure things out.

345
00:57:09.580 --> 00:57:21.770
John Harbison: And get going. I would say overall, though, it's been extremely positive. We just completed a hackathon not too long ago, where we just stopped work for an entire day, and we told the engineers

346
00:57:21.770 --> 00:57:36.459
John Harbison: You know, and we allowed product and design into the room, and actually our data team as well, and we said, just partner up, and like, go make some cool agents that just do something, and let us see what it does, let's see how you implement it, let's see the problems you solve.

347
00:57:36.500 --> 00:57:45.609
John Harbison: They're having fun with all of this technology, and it's completely changed the way they work, not just from a technical perspective, but the fact that, like.

348
00:57:45.610 --> 00:57:57.689
John Harbison: they can actually have a business and a product discussion with us, and they feel like valuable stakeholders in the conversation now, because they're not just looking at a ticket and saying, okay, I'm going to do that today.

349
00:57:57.890 --> 00:58:15.979
Jess Wolfe: Yeah. So, when these metrics here, these are our leading indicators, right? This is the pull request, but the pull request still has to be deployed to production, and was it right? Did we build the right thing? So, when we're looking at deployment frequency, we also see exponential,

350

00:58:15.980 --> 00:58:22.759
Jess Wolfe: changes there. Before, it was only 3.4 deployments per week, now all the way up to 14.

351
00:58:23.250 --> 00:58:36.590
Jess Wolfe: before, they didn't know even what their deployment time was. Sometimes just even getting the metrics in place can really help understand where you're at to make decisions. If you don't know, it's hard to make a decision around that.

352
00:58:36.590 --> 00:58:37.380
John Harbison: Absolutely.

353
00:58:37.380 --> 00:58:46.630
Jess Wolfe: Yeah, and John, I guess some of the things that have changed, right, we've got me time to recovery is a little bit different now than here, but the way you work is very different, too.

354
00:58:46.630 --> 00:58:56.499
John Harbison: Yeah, so one of the… one of the things that's been very interesting for us is… I mentioned in the beginning, we were very just-in-time, right? Like, as soon as a pull request made it through.

355
00:58:56.510 --> 00:59:03.939
John Harbison: All the testing, and we liked it, you know, we'd put a feature flag around it, and we would just deploy it to production, and it would just go.

356
00:59:03.940 --> 00:59:19.859
John Harbison: And now, because we have this extra bandwidth in our heads, and we're having all this AI tooling, we're actually kind of getting more into release management. So we're taking some of our more mature lines, and we're actually slowing them down and putting structured releases in place.

357
00:59:19.860 --> 00:59:25.959
John Harbison: and bundling a bunch of functionality together. So, it's kind of gotten us to the point where it's like.

358
00:59:26.430 --> 00:59:44.019
John Harbison: you know, period A to period B, there's a difference, and it looks kind of detrimental, but this is where context comes in. It's like, we look at a continuous deployment team against a continuous deployment team, we're looking like night and day, instead of, like, slowed down what happened.

359
00:59:44.700 --> 00:59:52.480
Jess Wolfe: Yeah. So, as you can see, one of the things that we noticed when we first started all of this, when John said, how do I measure, you know.

360
00:59:53.100 --> 01:00:10.809
Jess Wolfe: the impact of AI now, we did look at mean time to recovery, we did look at change failure rates to see if, is AI producing quality code, right? And as you can see, this has improved in the last 90 days. John's team's still working on that, but even with that, and this is why Agile's still important, because

361
01:00:10.810 --> 01:00:20.339
Jess Wolfe: there are still problems that still happen. There are still ways that we need to optimize the way that we're working. It's just not going to look like Scrum in the future.

362
01:00:20.850 --> 01:00:26.789
Jess Wolfe: But there's other flavors of Agile, that can be leveraged.

363
01:00:27.330 --> 01:00:45.550
Jess Wolfe: All right, the other thing John mentioned was leveling up our throughput on our epics. John, you all went from being able to complete 9 epics in 30 days to 32. What was… was the… was the size of the epic different, or is it now that… Nope.

364
01:00:45.550 --> 01:00:55.459
John Harbison: No, the Epics are the same size as they… I mean, you know, Epics come in different sizes, but yeah, our mix has not changed. This is just the huge unlock of

365
01:00:55.670 --> 01:01:12.260
John Harbison: you know, letting the AI come in and help us with this work and get the throughput that we're looking at. And you'll notice that, like, the team size for this measurement's the same. Like, we didn't add a bunch of people to cheat the metric. It's, you know, all AI enablement and process improvement that got us here.

366
01:01:12.590 --> 01:01:13.440
Jess Wolfe: Yeah.

367
01:01:14.500 --> 01:01:15.270
Jess Wolfe: But…

368
01:01:15.450 --> 01:01:22.400
Jess Wolfe: Here's where I think the most important part of the story, really lands us, and that's that leadership

369
01:01:22.400 --> 01:01:36.510
Jess Wolfe: the leadership shift that we had talked about, and we have a question, actually, that came from the audience. In your experience, does developing business

acumen within engineering, whereas managers acting as coaches, actually improve outcomes?

370
01:01:37.150 --> 01:01:56.119
John Harbison: Yes. Yes, yes. You know, in our first year of running, but this was, like, before we really unlocked AI, before it was on the table, engineering leadership and product leadership had the domain knowledge. We've been doing this for 20-something years individually.

371
01:01:56.300 --> 01:02:08.719
John Harbison: So, when we would, you know, hand something off to engineering, we would talk to them, we would explain things to them. The context is huge in some of these things that we're doing. So…

372
01:02:08.830 --> 01:02:25.660
John Harbison: By changing that dynamic to where they're in it every single day now, and they're sitting at the table, and they're asking questions, that was one of the hardest things, I'd say, was getting engineers to put their hand up and say, I don't understand something, and having a safe space to do that in, where they can

373
01:02:25.830 --> 01:02:31.440
John Harbison: learn, that domain, and learn the missing pieces that they needed.

374
01:02:31.680 --> 01:02:36.169
John Harbison: So that was a huge shift for us, and it's made a night and day difference.

375
01:02:36.380 --> 01:02:40.989
John Harbison: We started small, which slide are we on now?

376
01:02:40.990 --> 01:02:44.280
Jess Wolfe: So I just put this, up so this way it can talk about the org shift.

377
01:02:44.280 --> 01:02:48.979
John Harbison: Yep, yep, sorry, I was… I was looking at the other one.

378
01:02:49.480 --> 01:02:58.910
John Harbison: Yeah, no, and, you know, all the things we had to do, you know, I think, and this kind of points to the previous session, is, like, one of…

379
01:02:59.050 --> 01:03:05.039
John Harbison: you know, the things that we've ran into, and I think there was a question, I'm going to kind of touch on this a little bit, is…

380
01:03:05.260 --> 01:03:09.880
John Harbison: Controlling AI slop is a lot about your process and your technique.

381
01:03:09.990 --> 01:03:18.179
John Harbison: If you're… what I would say, at least from us, is, like, it's still quite possible to have a bunch of AI slop enter the pipeline.

382
01:03:18.310 --> 01:03:33.010
John Harbison: And there are ways that you can help that, and we've gotten it down to a pretty good with our process, but that's also, like, our process and kind of unique. Like, not everybody's gonna do it the same way. The other part.

383
01:03:33.410 --> 01:03:35.630
John Harbison: is…

384
01:03:35.780 --> 01:03:49.100
John Harbison: Yeah, I would also agree. Discipline and governance is huge for this. I think where we've kind of moved that, though, is a lot of that's fallen to engineering to both implement and enforce, is one of the changes.

385
01:03:49.390 --> 01:03:56.990
John Harbison: The other thing, too, which is… is interesting is dom… is the context, like.

386
01:03:57.180 --> 01:04:02.849
John Harbison: If something gets created, or if there's architecture, if there's decisions made.

387
01:04:03.000 --> 01:04:18.360
John Harbison: Being able to put that somewhere where not only can the humans see it, but AI can also see it, know to search for it, and know it's relevant to the task that it's now starting, so that it can inherit all that knowledge as well.

388
01:04:18.400 --> 01:04:26.100
John Harbison: I'd say that's kind of one of our big things we're working on right now, is how do we always keep the AI fresh on previous context?

389
01:04:27.060 --> 01:04:37.209
Jess Wolfe: Absolutely. You know, the first thing we recommend after, you know, learning from John's story now is fix your data hygiene.

390
01:04:37.540 --> 01:04:42.989

Jess Wolfe: Because if… if your data's wrong, how can you trust it? It's really hard to make those…

391
01:04:43.130 --> 01:05:00.059
Jess Wolfe: changes if that data hygiene's wrong. You cannot accelerate a broken foundation. If your JIRA tickets are a mess, AI will create a mess faster for you. So, you can move fast over a cliff, or maybe move fast into the sunset. It's your choice.

392
01:05:00.150 --> 01:05:23.609
Jess Wolfe: Allow room to learn. We are in a learning environment right now, and in order to make decisions fast, as Steve was mentioning in the last talk, we need to enable the organization to be able to learn. And if you, as a leader in your organization, are expecting people to just do it at the side of their desk, it's going to fail miserably. You really have to deliberately invest

393
01:05:23.680 --> 01:05:24.740
Jess Wolfe: in this.

394
01:05:24.930 --> 01:05:36.819
Jess Wolfe: And the third is measure your outcomes, not the activity. The activity's gonna help us understand what's going on, where problems are, but at the end of the day, it's more about

395
01:05:36.920 --> 01:05:50.559
Jess Wolfe: combining building the right thing and building the thing right in order to know if you're actually moving the needle. So focus on features shipped and business value realized, not how many hours someone's spending at their keyboard.

396
01:05:50.590 --> 01:05:59.980
Jess Wolfe: Because especially if we can have agents doing this work, that's going to be a whole lot better. John has created a starter AI workspace I'd love to let him talk about, and…

397
01:05:59.980 --> 01:06:06.069
John Harbison: Yeah. So, one of the things that I thought was, you know, and ran into as a challenge is

398
01:06:06.280 --> 01:06:09.409
John Harbison: you know, and again, we were focused on Claude at the time.

399
01:06:09.550 --> 01:06:23.809
John Harbison: Especially if you're coming from a non-technical background, like, trying to read the Anthropic documentation, the Claw documentation, or anything else,

and like, oh, what are skills? What are hooks? What are this and that? It's overwhelming.

400
01:06:23.920 --> 01:06:43.289
John Harbison: So I've used the starter workspace to kind of bring the less technical folks along for the ride, and just kind of show them, like, what good looks like. You know, things like, hey, how do you let, you know, take your session findings and re-improve your prompt for the next round, and inject that context?

401
01:06:43.330 --> 01:07:03.399
John Harbison: So, if you're dealing with, like, a, you know, I mean, if you're technical, please go look at it, but, like, if you're in a non-technical space, and you just want to, like, I want to know what's possible with, like, good hygiene and good process, it's a good place to at least start that, and then either use it or go to the technical docs and the native functionality and implement from there if you're comfortable.

402
01:07:03.630 --> 01:07:08.920
John Harbison: The one thing I do want to hit real quick is Corey's comment here.

403
01:07:08.920 --> 01:07:09.440
Jess Wolfe: Hmm.

404
01:07:09.660 --> 01:07:21.120
John Harbison: We still have, and I'm assuming this was meant for the pull requests, but we still, do review on the pull request. We just change what we're looking at. We don't look at the code anymore.

405
01:07:21.410 --> 01:07:23.300
John Harbison: We test the build.

406
01:07:23.350 --> 01:07:36.959
John Harbison: And that's what even the engineers do, is we don't try to look at it from that standpoint. We click the buttons, if it's a web app, we read the acceptance criteria, the business process, and we try to make sure that it actually works the way it was meant.

407
01:07:36.960 --> 01:07:46.430
John Harbison: We hope that the tests that were written, the automation that's going on, the linter, all of those things are keeping the hygiene where it needs to be.

408
01:07:46.490 --> 01:08:02.800
John Harbison: And then we obviously have some things in place for, like, did you do something silly, like put a secret in here? Or, you know, like, you know, we have

that kind of stuff in place too, but the humans, it's too much code to look at now. We're shipping too much too fast. It's getting lost. You can't read all the code.

409
01:08:03.550 --> 01:08:04.530
Jess Wolfe: Yeah.

410
01:08:05.040 --> 01:08:15.369
Jess Wolfe: So if you want to learn a little bit more about John's story with Swarmia, you can scan the QR code here. That last QR code before was, for his,

411
01:08:15.720 --> 01:08:28.610
Jess Wolfe: workspace that you can also take a look, so I'll, like, take a moment for you if you want to scan that. And that's for the Starter AI workspace. It's a GitHub URL. And this one is the story.

412
01:08:28.609 --> 01:08:42.089
Jess Wolfe: There was another question, you know, oh, this is the data I'm looking for, right? That all came from Swarmia, and I want to mention that we are kind of on a path to continue to help with,

413
01:08:42.090 --> 01:09:01.039
Jess Wolfe: getting this context to you. Some things that are coming, are this engineering context engine, so if you wanted to, you know, ask, you know, what is happening with my context, or, you know, where is this particular, epic at, what's blocking it, you'll be able to ask Swarmia those questions in the future.

414
01:09:01.240 --> 01:09:04.850
Jess Wolfe: Amongst other things, too. I can go ahead and stop sharing.

415
01:09:05.810 --> 01:09:12.359
David Mantica: So, we've got some questions here, you've got some time, so tips about data quality, AI, using AI to help with data quality.

416
01:09:13.350 --> 01:09:16.010
John Harbison: Data quality, like, actual data quality.

417
01:09:16.010 --> 01:09:25.609
David Mantica: Yeah, so remember you talked about it, the foundation crap, you're gonna get crap out. So I got a bad… so say I got a crappy foundation, can AI help me clean my foundation?

418
01:09:26.450 --> 01:09:31.380
John Harbison: That is a loaded question, and it's a very hard question.

419
01:09:31.880 --> 01:09:34.239
Jess Wolfe: I was gonna say, I'd argue, I think Swarmia helps you with that.

420
01:09:34.240 --> 01:09:47.629
John Harbison: Yeah, I mean, that was kind of where we went as well, is we… we used Swarmia to kind of expose what we had, and it was like, because it, you know, connected to Jira, it connected to GitHub, and…

421
01:09:48.130 --> 01:09:57.079
John Harbison: then I was like, oh, this is not great, and then I could actually start working either with myself or with AI to actually now fix the foundation.

422
01:09:57.080 --> 01:10:02.149
David Mantica: Alright, so help you find your problems. That's great. That's pretty cool.

423
01:10:02.150 --> 01:10:02.690
Jess Wolfe: Yeah, of course.

424
01:10:02.690 --> 01:10:03.529
David Mantica: Good, good job.

425
01:10:03.530 --> 01:10:19.979
Jess Wolfe: I was gonna say, here's an example, if you wanted to see, you go right into your inbox, and it tells you everything that you're doing. Here's an example. You can see what's unlinked, so this way I can now get this work linked to Jira, very easily. So, it really just helps you clean up that foundation and get things associated to where they need to go.

426
01:10:19.980 --> 01:10:32.899
David Mantica: And that's what conference is about. Conferences are about making you aware of all these opportunities. The big question for me, this comes straight from me, is business acumen skills. So what are the skills that you would want to give on business acumen?

427
01:10:33.640 --> 01:10:39.180
John Harbison: Oh, so I can… I can tell you, like, from our standpoint,

428
01:10:39.370 --> 01:10:47.680
John Harbison: you know, we were very lucky with the engineers that we have, both from our contractors and from our hires.

429
01:10:47.740 --> 01:11:06.399

John Harbison: I think that's one of the things that people don't realize is, you know, and I think this is across every department, and I just speak for engineering, because that's where I come from, is a lot of the problems to actually answer this question are the obstacles that are being put in front of the team and the silos.

430
01:11:06.540 --> 01:11:16.380
John Harbison: You know, if you've got the right people, no matter what function they're in, they're going to be naturally curious about all this stuff. They're going to want answers to these questions. They're already going to think this way.

431
01:11:16.380 --> 01:11:32.560
John Harbison: they don't have the ability to get at the data and the investment from the business and the other teams to teach them this. Now, you can always have a bad actor or something like that, but generally, what I found is once I took the hurdles away and said, hey, we want you to come sit at the table and have this.

432
01:11:32.560 --> 01:11:42.040
John Harbison: We didn't have to go into any training on the skills itself. It was exposure to the domain. They asked the right questions, they intelligently thought about the process.

433
01:11:42.120 --> 01:11:49.640
John Harbison: I don't know if everyone's mileage is gonna be the same on that, but that's the experience I had, is it wasn't so much.

434
01:11:49.640 --> 01:11:56.980
David Mantica: Is it more around customer need? Is it more like learning about what the customer wants, how the customer engages the product, how to position the product?

435
01:11:57.530 --> 01:12:19.359
John Harbison: unique to every group, right? Like, in our particular case, there was some of that, but there was also, like, strategic vision, market demand, right? Like, not every org is going to need the same thing, so that's why I just say it's like, tear the wall down, let them come to the table, let them ask the questions and talk about what they want to know.

436
01:12:19.530 --> 01:12:27.829
John Harbison: And then you can help lead, you know, a little bit on that discussion if you're finding people that need it, but tear the wall down first and let them come in and talk.

437
01:12:29.070 --> 01:12:47.079
Priyanka Malkoti: I also feel that creating user experience that is more intuitive helps people understand that, right? You, at one hand, you have the employees who

need to be curious, who need to understand how we are going to implement and integrate all these AI agents into our workflow.

438
01:12:47.140 --> 01:13:07.139
Priyanka Malkoti: At the other end, you need to have designers, specialized US designer, and that is where you can take the help of AI agents as well, to have something that is intuitive, where people don't really have to ask how to use the system. So that is something that probably then all these AI agents need to work on as well, to make it more user-friendly.

439
01:13:07.410 --> 01:13:26.920
John Harbison: Yeah, one of the things, and that actually was an interesting one we went through, is… so we had a dedicated design team that was UX UI, and we had research as well over there. And they're still here today, they're still doing their function. One of the things that we invested in as a company is we actually wrote a really robust design system.

440
01:13:27.190 --> 01:13:30.250
John Harbison: And we taught the AI how to use that.

441
01:13:30.460 --> 01:13:34.370
John Harbison: So, when we're now working as engineering.

442
01:13:34.460 --> 01:13:47.630
John Harbison: If it's already designed atoms and molecules that we're working with, the engineer can do 80-90% of the lift, and then bring it to design to refine, versus having to wait on a high fidelity to start with.

443
01:13:47.630 --> 01:13:57.069
John Harbison: and go to engineering afterwards. Now, for our more valuable assets, the things that we really want to make sure that we're focused on the UX UI as first

444
01:13:57.070 --> 01:14:15.009
John Harbison: number one priority. We still do it the traditional route. We bring the UX and design in first, run research, run user adoption, everything else that needs to happen, but we can now have that optionality as to what's the most efficient way and what's the value cost on the item we're trying to ship.

445
01:14:15.550 --> 01:14:40.469
Priyanka Malkoti: I think even though we have all these agents coming into picture, all this technology at our hand, few things are basic, few things are just simple, customer-centric. You need to have a system that the user wants to come back to. The customers want to… they don't want to call you to understand how to use your application. So I think those are, like, very basic things, and once we start from the

446
01:14:40.470 --> 01:14:45.519
Priyanka Malkoti: basics, and then expand about it, I think everything just falls into place.

447
01:14:48.200 --> 01:14:49.430
Jess Wolfe: Absolutely.

448
01:14:52.730 --> 01:14:53.890
Jess Wolfe: Alright.

449
01:14:54.170 --> 01:14:56.300
Jess Wolfe: Were there any more questions?

450
01:15:00.020 --> 01:15:17.619
Sasan Afsoosi: Sure, I have a question, great conversation. My question is focusing on the organization shift, and also about the role of the leaders and the managers. We mentioned that this is a shift left, not only as the agile practitioner says, during the past 10, 15 years, this is the new shift.

451
01:15:17.930 --> 01:15:20.930
Sasan Afsoosi: And the role of the managers are different.

452
01:15:21.400 --> 01:15:36.419
Sasan Afsoosi: the question is here that where do they fall on this? Because at some point, they need to run the organizations, no time to learn, and also this is a very, very deep shift, and where does it fit in the reality?

453
01:15:37.650 --> 01:15:47.360
John Harbison: Yeah, so I don't know if this is gonna hit with everybody, because, you know, of our org size and the way we're working, but some changes that have happened in mine. So, like.

454
01:15:47.940 --> 01:16:05.699
John Harbison: A year ago, I would say that 70-80% of my day was split between two things. One, just being a manager for my team of engineers and helping them. And two, my, excuse me, my roles with product, helping them, on that side.

455
01:16:05.990 --> 01:16:11.669
John Harbison: That has changed greatly now, as I maybe spend 10% of my time being a manager.

456
01:16:11.800 --> 01:16:29.650

John Harbison: And now I'm in the weeds with the engineers and saying, okay, I'm helping you build. I'm taking ownership of a… one of our epics. I'm going to help run a session with product on something. So that… that's one of the big changes for me, is

457
01:16:29.750 --> 01:16:45.500
John Harbison: it's changed my personal focus, and my CTO has done the same thing, is he is writing code all the time now, he's working with product, we're not just focused on the strategic level of the business and the people management of the business.

458
01:16:45.500 --> 01:17:04.919
John Harbison: as different team sizes scale, that's going to change as maturity on this curve within an organization, because a lot of this is… is an interesting maturity exercise to go through, right? So, everyone's going to have a different feel of that, but I'd say that's one of the things that's changed mostly for me is,

459
01:17:05.090 --> 01:17:15.290
John Harbison: you know, I'm still in the strategic conversations and the day-to-day and the management loop. I've just changed now the balance of that versus the other things that I'm doing.

460
01:17:15.920 --> 01:17:24.269
Jess Wolfe: John, would you also say it's even more important now to lean into the coaching and apprenticeship if you're a manager, to really help for a…

461
01:17:24.270 --> 01:17:24.900
John Harbison: Absolutely.

462
01:17:24.900 --> 01:17:30.229
Jess Wolfe: safe organization. And I mean, not split agile framework, but, like, safety.

463
01:17:30.230 --> 01:17:42.239
John Harbison: Yeah, so this is something I've yet to wrap my head around, other than just, you know, like, pure capitalism, is, like, you hear the stories, right, that are out there, you know, the toxic environments and the things companies are doing.

464
01:17:42.640 --> 01:17:45.740
John Harbison: My perspective is a complete different, like.

465
01:17:45.840 --> 01:17:53.330
John Harbison: We have looked at it from a standpoint of this allows us to ship more product out the door, you know, not as a cost-savings initiative.

466
01:17:53.330 --> 01:17:56.510
Sudhakar Partheepan: I'll save this initiative, I'll save this initiative, I'll say this initiative.

467
01:17:56.510 --> 01:17:58.640
John Harbison: echo somewhere.

468
01:17:58.980 --> 01:18:17.500
John Harbison: And then, additionally, it is so easy to teach and learn and get people excited now. One of the things I fear is, like, a lot of the juniors and intermediates have been left out of the company or having a hard time getting a start, you know, not… just speaking broadly, right? And…

469
01:18:17.660 --> 01:18:26.779
John Harbison: we're gonna need those people to have the experience and have the training in 5 to 10 years as attrition, and people move on, and people move up. And that's where…

470
01:18:26.780 --> 01:18:40.100
John Harbison: you know, I look at this as an amazing opportunity to bring someone in and start them in this process and start their development. If you have the right process and the right foundation, it's so easy now compared to even, like, a year or two ago.

471
01:18:42.430 --> 01:18:46.309
Priyanka Malkoti: And I also feel that for… even for entry-level employees.

472
01:18:46.310 --> 01:18:46.760
John Harbison: I think.

473
01:18:46.760 --> 01:19:07.019
Priyanka Malkoti: now the kind of exposure that they have, right? You have all the information in your hand, the industry that you are focused on, you just can't get the information about it. Yes, of course, you do not have the experience to have that kind of strategy in place, but the ideas are so easy to implement, to have

474
01:19:07.040 --> 01:19:31.880
Priyanka Malkoti: have a demo, a pilot, to go and show to wherever you're pitching, wherever you want to be. It makes it so easy for you to have those innovative ideas, because that has always been the case for… I have been with Chase for a long time, and most of my experience is around investment banking, and most of the things that a lot of new joiners

475
01:19:32.530 --> 01:19:42.950

Priyanka Malkoti: they struggle with is having their ideas put across to the management. Now, it is so easy for those people. These new ideas is what people need.

476
01:19:42.950 --> 01:19:49.769
Priyanka Malkoti: And they can… they can just come up with a pilot and show it to the senior management to get that into a fully… fully-fledged product.

477
01:19:49.990 --> 01:19:50.550
Jess Wolfe: Yeah.

478
01:19:50.550 --> 01:19:51.140
Priyanka Malkoti: So…

479
01:19:51.310 --> 01:20:08.709
Priyanka Malkoti: Yes, every time there is something new, there is… there is an unknown, which everyone, feels a little challenged about, but I think it is good. I think just at different levels, learning is going to be different, and I think that is what we all need to adopt.

480
01:20:09.530 --> 01:20:10.600
John Harbison: Absolutely.

481
01:20:10.600 --> 01:20:23.100
David Gijsbers: All right, well, John and Jess, thank you. This is exactly what we wanted. I think a lot of people came for the real-world stories, and, you know, having a data-backed story is,

482
01:20:23.220 --> 01:20:28.650
David Gijsbers: you know, certainly something that has a lot of credibility. Thank you for the questions.

483
01:20:30.550 --> 01:20:38.050
David Mantica: Dave G, real quick, John and Jess, will you stay for a little bit if anybody wants to private chat you some questions on the side? Are you okay with that?

484
01:20:38.050 --> 01:20:38.620
Jess Wolfe: Yeah.

485
01:20:38.620 --> 01:20:44.460
John Harbison: Yeah, yeah, and also, like, LinkedIn messages, like, just ping me on there as well if you think of a.

486
01:20:44.460 --> 01:20:46.649

David Mantica: If you want to share your LinkedIn, go for it, we'd love to…

487
01:20:46.650 --> 01:20:49.909
John Harbison: I think it's only one of the QR codes that'll come out with the.

488
01:20:49.910 --> 01:20:53.739
David Mantica: Okay, great, excellent, excellent, excellent. Alright, sorry, Dave G, keep going.

489
01:20:53.980 --> 01:20:55.789
David Gijsbers: Oh, why don't we,

490
01:20:56.270 --> 01:21:01.930
David Gijsbers: we're gonna pause for a second. We are gonna firstly thank our sponsors, of which,

491
01:21:02.080 --> 01:21:10.559
David Gijsbers: Swarmir and Dot Work have already presented, so thank you to both Swarmir and to DOTWORK.

492
01:21:10.720 --> 01:21:15.740
David Gijsbers: And we are going to hand it over to one of our sponsors for a, 10-minute presentation.

493
01:21:15.880 --> 01:21:18.430
David Gijsbers: Chris Kovachek, you ready to go?

494
01:21:19.100 --> 01:21:35.690
chriskovalcik: I am ready to go. An awesome conversation, John. I also dig the Millennium Falcon. Very cool to see that in the background. And do you realize we're a few minutes behind, schedule, so I'll try and kind of keep things brief, as I know everyone loves hearing from the sponsors, but just wanted to give

495
01:21:35.690 --> 01:21:57.579
chriskovalcik: you just… a high-level overview of Interpros from a perspective of where we fit within the AI space. So we've been in business for roughly about 30 years, top 100 women-owned business here in Boston, actually founded by my mom. So, near and dear to my heart, and we support one of the world's top two cloud providers, so we're seeing things that I think are very unique.

496
01:21:57.580 --> 01:22:06.770
chriskovalcik: In the marketplace in terms of how fast the market is moving, how quickly tools are being adopted, and, you know, we've been obviously doing this at scale for quite some time.

497
01:22:06.770 --> 01:22:22.949
chriskovalcik: But our focus is really, you know, trying to bring some of the skills and knowledge that we've, you know, been exposed to over the last couple years to the marketplace in just various different ways. In the bottom left, you'll see there's a QR code for an AI maturity scorecard. It's totally free. If you want to

498
01:22:22.950 --> 01:22:46.469
chriskovalcik: follow along, listen along, you can complete this by the time I'm done talking. The beauty of this is you'll get a baseline for where your company's at relative to your industry peers, which we've heard from a lot of other folks. It's hard to be compared to, you know, the Fortune 100s, depending on your size, your industry, different companies are at different stages, so really just trying to get a baseline of where you're at relative to your peers is a huge advantage.

499
01:22:46.470 --> 01:22:54.880
chriskovalcik: And also knowing where to go. But from that, you know, we do have a few different offerings from a capability standpoint, and this has really been developed

500
01:22:54.880 --> 01:22:59.280
chriskovalcik: As we've continued to work with some of the, you know, largest companies in the world, you know, our first

501
01:22:59.280 --> 01:23:12.679
chriskovalcik: focus with most of our clients is really around this readiness assessment. So that's really an add-on to the free scorecard. This is sitting down with your teams for anywhere from a week to two, really digging in, understanding what's going on in the business.

502
01:23:12.680 --> 01:23:37.110
chriskovalcik: where they're at, what to prioritize, and then ultimately, you know, how to justify and get funding and build that business case. Because right now, I think the market is at a point where they've spent a lot of money on AI and haven't given the results, and we're seeing a lot of the boards looking for some type of an ROI, whether it's, you know, truly reducing costs, increasing revenue, or improving operational efficiencies, but looking to take advantage of the benefits of AI.

503
01:23:37.460 --> 01:23:49.730
chriskovalcik: And then from there, it's really around, how do we move forward? If we have this plan, what do we do first? How do we prioritize? Do we have the right resources to do so? And if we don't, you know, is there a way for us to support that?

504
01:23:49.730 --> 01:24:00.149

chriskovalcik: And so what we've essentially done is built out what we call our forward deployment pods. So this is a team that's, you know, designed to come into your environment and work in really a tool-agnostic way.

505
01:24:00.150 --> 01:24:14.480
chriskovalcik: And what we've seen with a lot of the folks in the market are that they're really tool-focused, and focused on bringing in new tools, or helping you adopt the tools that the vendor's trying to sell you, whereas our approach here is really to try and maximize a lot of the investments you've had

506
01:24:14.480 --> 01:24:24.240
chriskovalcik: Just a little anecdote, I was with one of the heads of cloud strategy for Alphabet, and one of the things they said was that, you know, the Google Workspace is one of the most underused

507
01:24:24.240 --> 01:24:44.050
chriskovalcik: assets that companies have and all the investments they're making in, their data's telling them that it's just really not being maximized, and if companies can just spend a little bit of time and effort there, there's huge rewards and gains to be had, which ultimately leads into just skills training. I think we… we talked briefly about this in the last call, that there's a huge need to up-level and upskill

508
01:24:44.050 --> 01:24:52.230
chriskovalcik: some of the folks on our, you know, in our organization to get there, because this is a big change. And whether that's engineering folks or line of business folks.

509
01:24:52.230 --> 01:24:56.970
chriskovalcik: Really having that ability to up-level those skills on the business side.

510
01:24:56.970 --> 01:25:20.419
chriskovalcik: Because they are closest to the problem, and everything we're seeing is that if we can activate the business users to understand and identify their own use cases, they're more likely to embrace and adopt it versus it being forced down from IT. So that's really how we go to market. If we look at kind of the way we're assessing things, a lot of these topics have already been covered, but, you know, just at a high level, what we're seeing is that a lot of organizations have

511
01:25:20.420 --> 01:25:45.170
chriskovalcik: the strategy aligned from an executive standpoint. They're on a little bit of a more mature curve there. They have some of the right tools in place from a technology standpoint, and they have a lot of the governance in place. But where they're lacking is the talent, the culture, and I think in John's talk, they talked a lot about the data quality, data cleanliness, which is very common to have those three as the laggards, in terms of having your organization ready and not

512
01:25:45.170 --> 01:25:47.250
chriskovalcik: Obviously, we talked about, you know, that data swamp.

513
01:25:47.250 --> 01:26:05.170
chriskovalcik: being a major issue, but, you know, having that focus and, you know, what we would focus on with our teams is going in and actually assessing where you're at and where to make those investments to see that return relative to the opportunity you're on. So it's just an example output of what that exact summary would have. As I mentioned, the strategy, governance, and data.

514
01:26:05.170 --> 01:26:29.699
chriskovalcik: For this particular client, we're on the higher end. However, their talent culture and technology were lagging. They were ahead of their baseline cohort by 1.2, just given this industry they were in was not as mature, so the executive team was excited to see that, but wanted to make additional investments on the talent side in this specific use case, because they believed that they already had the strategy, they just didn't have the people to

515
01:26:29.700 --> 01:26:40.340
chriskovalcik: truly drive out, and in bringing that talent, they could ultimately try and up-level the culture. So just, again, high-level example of what this would look like with some of one of our clients.

516
01:26:40.360 --> 01:26:49.360
chriskovalcik: The scorecard I mentioned, as you guys go through this by the end of this chat, you should get your own scorecard, but this is completely free. It's, again, it's kind of a…

517
01:26:49.360 --> 01:27:13.299
chriskovalcik: a sample size of this to be self-assessed. It's great to send out to the rest of your teams to baseline and compare how line of business sees things versus technology, as there are two different paths for this, and the goal is really to bring those together to force a conversation around how business sees things, and how technology sees things. Because, you know, a lot of times, from a technology standpoint, we think everything is great, everything's been deployed.

518
01:27:13.300 --> 01:27:24.010
chriskovalcik: And then the business says, you missed the functional requirements that I was looking for for this to actually work. So this helps foster and culture that conversation, which ultimately leads to one of our customer success stories.

519
01:27:24.010 --> 01:27:38.889
chriskovalcik: So with this case, they did go forward with a four-deployment pod. They were in the middle of trying to understand where their advertising performance and customer targeting were lacking. They were seeing about a 15% drop-off over the last trailing 12 months.

520
01:27:38.890 --> 01:27:47.670
chriskovalcik: In their conversion, which, you know, obviously, at their size, was resulting in tens of millions of dollars of decreased revenue, and they just found the models were underperforming.

521
01:27:47.670 --> 01:28:09.299
chriskovalcik: they didn't have the right skill sets in place, and they needed to move fast to up-level their existing teams, so they ultimately turned to us to bring in that expertise, and I think someone was asking, where does an Agile coach, agile engineer came, it was actually the change specialist, was an Agile coach that was brought in to help lead and drive this change, and corral the different teams across this to ultimately drive that acceleration.

522
01:28:09.300 --> 01:28:13.349
chriskovalcik: They saw about 11% increase in advertising conversion rates.

523
01:28:13.350 --> 01:28:36.879
chriskovalcik: based on the new models that were deployed, which ultimately generated… this was a specific team and business unit, wasn't across all of their segments, but for this team, it was a 12% increase in incremental revenue, and they were able to basically do experimentations, you know, twice as fast, which was how they were able to unlock this. It was actually the speed that unlocked the opportunity, not necessarily the models, but the ability to iterate fast.

524
01:28:37.170 --> 01:28:55.090
chriskovalcik: So ultimately, you know, it was a huge success for them, and then they're looking to continue with this pod for the next 6 months as they're tackling a couple different additional enhancements. And that's all, really, we have. Would love to, you know, connect with anyone after. If they're interested in learning a little bit more about us, you can look at us at interpros.com forward slash AI.

525
01:28:55.090 --> 01:28:59.940
chriskovalcik: For a little bit more of the service offering, and obviously we're happy to stick around and answer any other questions.

526
01:29:00.430 --> 01:29:01.439
chriskovalcik: Thanks so much.

527
01:29:01.640 --> 01:29:12.889
David Gijsbers: Thanks very much, Chris. Next up, we did want to get a developer's perspective. So, as we move from, you know, the strategy into

528
01:29:12.920 --> 01:29:23.510

David Gijsbers: kind of the coaching and the management of the transition, we also wanted to get a view from… from the developer desk, and we're so excited to have Lada Kessler

529
01:29:23.640 --> 01:29:31.919
David Gijsbers: With us. So Lara, if you're on the line, I'd love for you to, take over the screen and, take us through the next presentation.

530
01:29:32.450 --> 01:29:35.109
Lada Kesseler: Sounds great. Give me a moment to rearrange some things.

531
01:29:35.290 --> 01:29:36.450
Lada Kesseler: Can you see my screen?

532
01:29:36.830 --> 01:29:38.280
David Gijsbers: Yes, you sound great, too.

533
01:29:38.790 --> 01:29:46.140
Lada Kesseler: Fantastic. One second, I wish to see people here, and Zoom is not… Being helpful right now.

534
01:29:49.140 --> 01:29:55.509
Lada Kesseler: Right, and I know we're running a bit behind, am I good with the…

535
01:29:55.510 --> 01:29:58.910
David Mantica: Yeah, Good, we're only 5 minutes behind, we're doing great.

536
01:29:59.370 --> 01:30:01.950
David Mantica: Do your… do your thing, the way you want to do it.

537
01:30:02.440 --> 01:30:03.800
Lada Kesseler: Okay, sounds great.

Core Patterns for Coding with AI

538
01:30:04.910 --> 01:30:13.429
Lada Kesseler: Okay, fine, fantastic. I see you now, too. Okay, so let's start. Hello, I'm Lena Kessler, and this is Core Patterns for Coding with AI.

539
01:30:13.730 --> 01:30:25.469
Lada Kesseler: So I wanted to take a little, like, step back and think about the experiences you had throughout the last, maybe, year and year and a half as you were working with AI.

540
01:30:25.880 --> 01:30:44.710
Lada Kesseler: Right? So, what would they look like? So, maybe there were some good results, right? So, what would that look like? You go to AI, right, anywhere in the browser, in the Agentic tool, or anywhere else, and you give it a task, and out comes exactly what you wanted, and you're happy.

541
01:30:45.090 --> 01:31:01.589
Lada Kesseler: If that's all you experience with AI, you probably don't need that stock. My guess is, like, if you're anything like me, that's not the case. And maybe, like, the least experience you have, maybe the least… it looks like this, right? So… and the bad result is pretty much, like, what you would expect. You go to the AI,

542
01:31:01.970 --> 01:31:04.470
Lada Kesseler: Ask for a thing, and out comes something.

543
01:31:05.140 --> 01:31:13.160
Lada Kesseler: it wasn't what you intended, and it wasn't what you wanted, it was a slop, maybe it doesn't even work, right? And question is, like.

544
01:31:13.330 --> 01:31:24.919
Lada Kesseler: why, and what can we do to get more of the first, right, and less of the second? If you… as you look at it, like, what does this slob consist of? So, many people are like, oh.

545
01:31:25.070 --> 01:31:39.369
Lada Kesseler: if that's your first experience with AI, right, or the second or third, then it's kind of easy to make a conclusion of just AI is bad, right? It's just not ready, it's, like, not… not there yet, and maybe, like, the whole technology is flawed or something.

546
01:31:39.390 --> 01:31:58.709
Lada Kesseler: But I've actually been, like, finding a lot of value in it, right? And, like many people, like, said here, like, the way I work completely changed in the last year, and I'm able to do things I wasn't able to do before, not even remotely, so it's kind of mind-blowing. And what I find is, like, it's really helpful when you find

547
01:31:58.730 --> 01:32:03.259
Lada Kesseler: Output that you don't like, to kind of give it a chance, and…

548
01:32:03.390 --> 01:32:10.569
Lada Kesseler: and see how it failed, right? And I find different ways in which it fails, and I kind of map them to patterns a little bit.

549
01:32:10.630 --> 01:32:24.349
Lada Kesseler: this is, like, a shorter talk, so here we're gonna just cover six of them, but those are gonna be foundational areas where you'll see the most impact, and the foundational patterns are going to kind of reinforce each area. All right, we're gonna go through all six,

550
01:32:24.430 --> 01:32:26.570
Lada Kesseler: And see how you can tweak.

551
01:32:26.610 --> 01:32:41.440
Lada Kesseler: So I've been coding with Agentic AI for the last year and a half, a lot, like, for hours on end, right? And I find that those patterns help me a lot, and when I share it with others, they seem to help them too. So I hope it'll be helpful to you. So,

552
01:32:41.440 --> 01:32:52.409
Lada Kesseler: There's resources to the stock, so if you're interested with the slides and some of the things, like, at the end I'll mention more additional resources, they will be here on this page.

553
01:32:52.410 --> 01:33:00.200
Lada Kesseler: I will also show this slide at the end, because you might not care about it, yet, right, because you don't know what the talk is about. All right, so let's dive into it.

554
01:33:00.290 --> 01:33:02.959
Lada Kesseler: And take a look. So…

555
01:33:03.100 --> 01:33:16.499
Lada Kesseler: So why are we getting bad results, right? So, what is the first, kind of, area that's gonna probably show up? So, we have two tasks, right? We gave it task A, and there's AI that gave it task B. This one produced good results, that one didn't.

556
01:33:16.810 --> 01:33:26.700
Lada Kesseler: When you cannot code with AI, or do… actually do any kind of work with AI, and what I'm talking about is actually going to be applicable not just to coding.

557
01:33:26.870 --> 01:33:33.040
Lada Kesseler: In my experience, it's foundational things that understanding that helps with building agents with just

558
01:33:33.040 --> 01:33:56.510

Lada Kesseler: talking to it as a product, writing, pretty much in all areas. So, okay, so there's relevant knowledge for the task, whatever that task is, right? And some of this… so here I'm just depicting them as those two pieces. You can think about it as, I don't know, knowledge of programming language, or information about your preferences, or what new features are available, and lots and lots of things, information about the project.

559
01:33:56.540 --> 01:33:58.660
Lada Kesseler: Right? There's some relevant knowledge to the task.

560
01:33:58.660 --> 01:34:22.890
Lada Kesseler: And AI knows some of it, right? Say, for example, programming languages, it was trained on extensively, right? It knows it a lot, you don't have to train it again. But some pieces, like, some tasks have… are dependent on some pieces, need some knowledge that is missing. So, for example, this red bit is not in the AI training, right? Maybe it's your specific thing that you, like, maybe it's your preference, maybe it's your project.

561
01:34:23.110 --> 01:34:28.539
Lada Kesseler: what's your project is about, what… how do you like to code? And so on. It's not there.

562
01:34:28.960 --> 01:34:36.610
Lada Kesseler: maybe it's actually feedback from the, environment, right? So, like, logs, or, how… what is the error message that you're seeing, and so on.

563
01:34:36.610 --> 01:34:49.540
Lada Kesseler: So, when the task depends on that, basically, the likelihood of it failing is very big, and the likelihood of it failing, just not working, and AI may be even saying that everything is good, is also pretty high.

564
01:34:49.540 --> 01:35:05.470
Lada Kesseler: Because it's really trying, right? You didn't give it enough information, but it's really, really trying to please you, but it's just, like… you know how, like, you wouldn't… like, you wouldn't get an intern and send them to do something big without giving them the tooling and the training? It's kind of something like that.

565
01:35:05.510 --> 01:35:15.859
Lada Kesseler: AI is kind of like that, so you kind of need to prepare the context for it to be successful. Otherwise, like, you can blame it and scream at it and so on, but, like, it really doesn't help. It's not…

566
01:35:16.170 --> 01:35:29.369
Lada Kesseler: like, understanding that helps a lot, I think. And that's the kind of the core foundation of, like, what the contest management is, and that's kind of the

core thing that you need to worry about. Right, so here, this is the task piece, so we're getting the swap result, so what can we do?

567
01:35:29.370 --> 01:35:38.329
Lada Kesseler: We're still not ready to give up. So we steer it a bit, right? We give it more instructions, like, no, no, no, this is not quite right, this is not what I wanted.

568
01:35:38.330 --> 01:35:45.580
Lada Kesseler: And you give it a little bit of, like, information of where you want to be, right? Maybe some of this is a log, this is how it's failing.

569
01:35:45.580 --> 01:35:55.690
Lada Kesseler: Maybe some of it is information about, no, I actually wanted a different technology, or you did something wrong, right? And maybe it's still, like, it's not quite right, produced bad result again.

570
01:35:55.690 --> 01:36:13.580
Lada Kesseler: And then you give it a bit more instructions, and then suddenly you are kind of closer to where you wanted to be. Right, so what happened here is, like, you basically added these things that it needed, or information it needed, to be successful. And it's not deterministic, so it's not, like, 100%, but roughly, I think it's a helpful model to have.

571
01:36:13.860 --> 01:36:28.010
Lada Kesseler: And you see how you put information in this context, and the problem with this context is it's fleeting, right? As soon as I close it, it will go away. But you actually taught it things, and unlike intern, which won't forget the things it taught.

572
01:36:28.010 --> 01:36:43.999
Lada Kesseler: this one, this one will, right? You close the window and it's gone. So it wouldn't it be nice to just actually have a library of these things that you taught it, and put it somewhere, right? So you kind of grab this knowledge and put it somewhere, you can find it, and all the sessions can basically be taught that by just pointing.

573
01:36:44.400 --> 01:37:04.300
Lada Kesseler: So, that's the idea of knowledge documents, right? So you basically have information that you care about in your organization, your company, yourself as a developer, or as any kind of, like, person, right, even for personal stuff. And you're gonna start assembling a collection of those things. And I think people are starting to realize how powerful this is, because we've been doing this actively.

574
01:37:04.300 --> 01:37:19.870

Lada Kesseler: And this kind of led to… this and some other things led to things like skills, and now you can teach your AI a whole toolset or skill set just by giving it a skill, which is kind of sort of a knowledge documents plus some idea of…

575
01:37:19.980 --> 01:37:33.640
Lada Kesseler: how to manage context, additional ideas, right? So it really makes you really powerful, because now you have capabilities that you can give to your AI, and make it much better. All right, so that's the idea of knowledge documents. Let me see.

576
01:37:33.900 --> 01:37:51.860
Lada Kesseler: And so, basically, that's the foundation, right? You have AI with some pre-trained knowledge, but not everything is there. You need to kind of figure out what needs to be there, and give it to it to be successful. And the thing is, some of the tooling has been evolving, right? So, there's some stuff, like.

577
01:37:52.490 --> 01:37:54.500
Lada Kesseler: There's some stuff like…

578
01:37:54.500 --> 01:38:15.150
Lada Kesseler: So to-do list used to be a thing I would write, right? A half a year ago. I still write them, to be fair, but actually, it's in every tool. It's in every agent now, right now. They already kind of incorporated that tool, that to-do MD in there, and it's really helpful to get the agent to structurally do things one at a time, and, like, not forget pieces.

579
01:38:15.150 --> 01:38:31.099
Lada Kesseler: And then CloudMD is a thing, basically, or AgentMD, right? This is a file that, whenever you open an agent, everything in there is in your context. You don't have to pull it in, you don't have to do anything, it's already there. So kind of tools… tooling is adapting to this, but I find that

580
01:38:31.100 --> 01:38:49.019
Lada Kesseler: understanding the basics is still helpful, because when the tooling are not, like, they're not gonna cover everything, right? It's helpful because it's easier to start with them as a beginner, but understanding the foundations will actually help you work with any agendic AI and build Agentic systems anywhere, so I think it's really helpful to keep that in mind.

581
01:38:49.020 --> 01:38:52.279
Lada Kesseler: And the fact that tooling is adopting it doesn't make it obsolete, I think.

582
01:38:52.280 --> 01:39:05.169

Lada Kesseler: All right, so we are talking about context management, right? So, the idea that you can make UI really powerful and teach it things by building and starting to build your own knowledge library, or the knowledge library for your team.

583
01:39:05.350 --> 01:39:22.469
Lada Kesseler: But then the question is, like, okay, we need to teach it missing things. How do you know what's missing, right? Is there a way to kind of find out what knowledge it needs without, you know, running into walls every time, without pain? I find a way that's really helpful to me.

584
01:39:22.700 --> 01:39:38.340
Lada Kesseler: Basically, think about it this way. There's some kind of, like, here we have this relevant knowledge. We don't know what it is. How do we find out? Well, you know, like, one of the things that people do a lot when they start working with AI is, like, treat it as an order taker. You're like, hey, go do this thing.

585
01:39:38.600 --> 01:39:40.250
Lada Kesseler: And AI's like, sure, sure.

586
01:39:40.390 --> 01:39:44.699
Lada Kesseler: And so on, and does the stuff. But I find it a lot of, like, help.

587
01:39:44.700 --> 01:40:01.909
Lada Kesseler: can come from just reversing this direction, right, and seeking output from AI. You are looking for it to give you things, to tell you things, and so on, and show you things. Show you things is especially powerful because, like, it is so well read, it can give you so much options that you don't know and didn't think about.

588
01:40:02.020 --> 01:40:20.689
Lada Kesseler: So, I call this reverse direction. It's a, like, bigger pattern that has a little bit more, different flavors to it, but the idea is basically thought partnership, right? So, how… let me give you a few examples of what it would look like. So, you have this task still, with unclear what it needs, right? And relevant knowledge…

589
01:40:20.890 --> 01:40:34.890
Lada Kesseler: So, the one thing I do when I give it a task, I'm like, hey, what's unclear, right? And that immediately makes it visible to me. It asks me questions. It shows me, hey, this is not quite right, or something like that. And now I can actually adjust my relevant knowledge and give it the enough information.

590
01:40:34.890 --> 01:40:56.139
Lada Kesseler: And that's how I found out, and maybe some of that goes to my knowledge library, but now, like, I gave it the knowledge it needs. Pretty straightforward, but many people are not doing that, and I think, no, this is

actually super helpful. And then, another flavor to this is, like, tell me what you plan to do before you do it, right? So, I gave it a task, we had a conversation back and forth, so I gave it… maybe I'm doing spec-driven development, right?

591
01:40:56.140 --> 01:41:03.750
Lada Kesseler: it's… before it goes and, like, for 30 minutes works on something, then produces something, God knows what, I can actually…

592
01:41:04.210 --> 01:41:19.019
Lada Kesseler: see a little bit into it by asking, hey, what are you gonna do? Show me. Real briefly. And actually, that catches a lot of silliness, right? Right here. So, here it's like, oh, I'm gonna do this, and then… and like, well, you're missing a step here.

593
01:41:19.020 --> 01:41:28.390
Lada Kesseler: please fix it, right? So, and now I know that I need to kind of bring its attention to that, and it goes to my realm… I gave it that knowledge, right? That's how I found out.

594
01:41:28.480 --> 01:41:36.889
Lada Kesseler: Another flavor to this is… so, especially when it's like, hey, what do you, like, we're doing something, and it's like.

595
01:41:36.950 --> 01:41:50.280
Lada Kesseler: give me… it asks my… it asks me a question, what would you like to do, right? And I'm like, I'm reversing direction and saying, hey, give me some options. I don't think what I want to do, I… hey, show me the options. And this gives me, like.

596
01:41:50.280 --> 01:42:03.909
Lada Kesseler: like a menu, a big menu, and it gives me ideas, right? And here, for example, it gave me options… these options, and maybe I didn't even think of B at all, but I'm like, oh, I like B. Actually, let's apply B to everything.

597
01:42:03.930 --> 01:42:11.800
Lada Kesseler: I just prefer B, and use B in the whole project, maybe, right? So now I can… I kind of gave me ideas, and so… so this kind of…

598
01:42:11.990 --> 01:42:19.789
Lada Kesseler: this kind of concepts are super helpful. And another flavor, the last one, I guess, is criticize me, right? That's also pretty helpful.

599
01:42:20.280 --> 01:42:29.609
Lada Kesseler: it asks me questions, and now I know that I need to tell it more, or, like, adjust myself, what I'm doing, and so on, and that goes into relevant knowledge. So…

600
01:42:29.610 --> 01:42:41.840
Lada Kesseler: this is what reverse direction, roughly, is about, and, like, what I meant… what I went through is, so ask me anything unclear is one of the things you can do. Tell me what you plan to do before you do it.

601
01:42:41.840 --> 01:42:56.609
Lada Kesseler: give me options, so see more, see a menu of things as possible, and then criticize my plan. This is some of the things you can do, you can do much more, but the idea is basically, you're dealing with something that has invisible brain, right? But it has some kind of mental state.

602
01:42:56.610 --> 01:43:05.309
Lada Kesseler: How do you… how do you find out what's there? This is some of the techniques you can use to kind of make it more visible to yourself as you work with it. And then.

603
01:43:05.400 --> 01:43:10.070
Lada Kesseler: that helps a lot by kind of aligning the mental models and getting better results, I found.

604
01:43:10.510 --> 01:43:14.559
Lada Kesseler: Alright, so that's the pattern reverse direction, really helpful.

605
01:43:15.050 --> 01:43:32.060
Lada Kesseler: And we started with, basically, AI doesn't know everything, right? We need to build our knowledge library to be more powerful, and then how do you find what's missing? Basically, reverse direction is a way to find out what's missing. That's really helpful.

606
01:43:32.600 --> 01:43:35.409
Lada Kesseler: Alright, so now let's…

607
01:43:35.620 --> 01:43:41.709
Lada Kesseler: So now we have… we're building knowledge documents, right? So, and now our knowledge library is growing.

608
01:43:41.730 --> 01:43:53.779
Lada Kesseler: And we kind of like it a lot, right? We're getting good results, like, we're excited. So here's the knowledge library, and now we're, like, starting to be like, oh, okay, let's just add everything and give it all to you. We're trying to be helpful, right?

609
01:43:53.780 --> 01:44:02.439

Lada Kesseler: So we just grab everything, put it to the knowledge thing, and give it to AI. And then give it a task. And then… we get something not great.

610
01:44:03.110 --> 01:44:14.050
Lada Kesseler: Whoa, no, what's happened? Why? Why is that happening? Right? And you go to it, and like, hey, AI, I told you this thing, why didn't you follow it? And maybe you shouted it for a bit.

611
01:44:14.220 --> 01:44:18.699
Lada Kesseler: But it's a little bit like shouting at a kid who's, like, your 4-year-old, you know, like.

612
01:44:18.960 --> 01:44:25.070
Lada Kesseler: maybe don't do that, but also, like, to the child, but also, like, AI doesn't mind, but obviously, but…

613
01:44:25.240 --> 01:44:33.479
Lada Kesseler: it's not helpful. It's… you're dealing with some limitation that is, like, more of a cognitive thing, right? AI has something like a focus, and just to illustrate the concept.

614
01:44:33.620 --> 01:44:36.399
Lada Kesseler: I'm gonna give you a little, tiny, small exercise.

615
01:44:36.660 --> 01:44:55.790
Lada Kesseler: So, I'm giving… I'm gonna give you seven sentences twice. The first time and the second time. Just let me read your sentences and try to remember as much as possible. Like, don't write anything down, but just for yourself. And… and think about how many things you could remember the first time and the second time, right? So, here's the first sentences. So…

616
01:44:56.070 --> 01:44:58.940
Lada Kesseler: A unit test should test one thing at a time.

617
01:44:59.550 --> 01:45:01.740
Lada Kesseler: A unit test should be small and focused.

618
01:45:02.460 --> 01:45:05.280
Lada Kesseler: And your intest should be… should run fast.

619
01:45:05.900 --> 01:45:07.559
Lada Kesseler: They should be deterministic.

620

```
01:45:08.390 --> 01:45:10.729
Lada Kesseler: It should not depend on external systems.

621
01:45:11.380 --> 01:45:13.989
Lada Kesseler: It should clearly express its intent.

622
01:45:14.380 --> 01:45:16.439
Lada Kesseler: And it should fail for only one reason.

623
01:45:17.240 --> 01:45:29.510
Lada Kesseler: Got that part? I'm gonna talk a little bit, so the last bit is coming
out of your brain, and you're not cheating. So now think about how many things you
can remember from that. Just roughly, like, know to yourself.

624
01:45:34.590 --> 01:45:45.410
Lada Kesseler: All right, and then let's try a new one. So, here's a few sentences.
Same thing, seven things. Start with a clear purpose. Write down the problem you're
solving and who it's for.

625
01:45:45.990 --> 01:45:49.909
Lada Kesseler: Involve the right people early, especially those who will use or
maintain the system.

626
01:45:50.930 --> 01:45:55.670
Lada Kesseler: Break work into small, testable increments that can be completed in
days, not months.

627
01:45:56.680 --> 01:46:00.360
Lada Kesseler: Keep the main branch always releasable through continuous integration.

628
01:46:01.100 --> 01:46:05.489
Lada Kesseler: Automate builds, tests, and linting, so quality checks are not
optional.

629
01:46:06.360 --> 01:46:09.620
Lada Kesseler: Write code that optimizes for readability over cleverness.

630
01:46:09.740 --> 01:46:13.960
Lada Kesseler: And document decisions. Why? Not just APIs. What?

631
01:46:14.330 --> 01:46:20.960
Lada Kesseler: So, kind of same idea, right? 7 sentences, and just for yourself, take
it out. Which one was easier for you to remember?
```

632
01:46:21.850 --> 01:46:22.870
Lada Kesseler: Do you think?

633
01:46:23.200 --> 01:46:35.559
Lada Kesseler: You also might vary, because I realize that the first one is about testing, and you might not care about testing, but, like, if you're like most people, I think you would find that it's a bit easier

634
01:46:35.560 --> 01:46:50.739
Lada Kesseler: to remember the first versus the second, because those first seven sentences are about the same thing, it's about unit tests. And here, we're just all over the place, right? We're talking about this, that, and this thing over there. It's all interesting and important, but nothing connects them, apart from, this is good stuff!

635
01:46:51.780 --> 01:47:16.769
Lada Kesseler: So, what we're dealing with here is a little bit of focused versus scattered brain, right? Here, the ideas here reinforce each other, right? They're kind of zeroing on into the idea of good… what good, unit tests looks like. And here, like, we are all over the place. And the thing is, if you need to do something specific here, like, how likely… like, you have to run all over the place, right? Your brain is scattered. And I find that the same thing is kind of happening to the

636
01:47:16.770 --> 01:47:21.160
Lada Kesseler: And this is the thing that I find the most people getting… not getting.

637
01:47:21.240 --> 01:47:31.820
Lada Kesseler: like, even the people who have a lot of experience are messing it up and, like, kind of blaming it on AI, when instead you're dealing with a physical limitation. There's a concept of focus or attention.

638
01:47:31.820 --> 01:47:48.090
Lada Kesseler: That just doesn't have as much. Some people are joking that AI has, like, 7,000 plus minus 2 attention, or, like, things to keep in mind, but I think it's actually wrong. I think it's actually plus 5 plus minus 2 as well, like, for humans. It actually is not much better at, like.

639
01:47:48.090 --> 01:48:12.410
Lada Kesseler: focusing goes a long way. So then, we don't want to actually dump everything we know into the AI. First of all, it won't fit. Second one, this kind of brain thing. So what do you do? Especially when you want really, really good results, especially if it's an important task, you want to basically focus the agent. How do you do that? Well, give it a task. A dedicated task helps a long way. So, like, you can have a generalist agent that does everything, or you can have

640
01:48:12.410 --> 01:48:14.639
Lada Kesseler: We have an agent that just does this one task.

641
01:48:14.640 --> 01:48:16.129
Lada Kesseler: The second will be.

642
01:48:16.130 --> 01:48:32.579
Lada Kesseler: crazy more, reliable. Like, it's not, like… the answer to me is, like, the difference I saw is, like, between never and always, following the instruction that I gave it. Pretty dramatic. And then you give it the specific things that it needs, only those things, and you get good results.

643
01:48:32.580 --> 01:48:37.090
Lada Kesseler: Right? Dramatic difference, this is important.

644
01:48:37.190 --> 01:48:48.439
Lada Kesseler: So, and again, here… see, see, this is the generalist, right? So, we gave it everything, and we asked maybe to give a bigger task, do a bigger task, and here we just have this thing. Depend on the same thing.

645
01:48:48.470 --> 01:48:57.420
Lada Kesseler: Both of them, both of those tasks. But this is distracted, it's scattered all over the place. Maybe us to do all of those things at once, versus this one is just focusing on one thing.

646
01:48:57.960 --> 01:49:00.290
Lada Kesseler: Not even comparable performance.

647
01:49:00.650 --> 01:49:02.530
Lada Kesseler: Alright, so…

648
01:49:02.720 --> 01:49:20.339
Lada Kesseler: This is the idea of focused agents. Make your agents focus, but if you're doing, like, agendic systems, so actually writing agents, that'll also go a long way. As soon as you start, like, seeing, oh, it doesn't follow the instructions, this is where you need to get suspicious, and like, oh no, maybe I need to narrow down, maybe I need to split it up a bit.

649
01:49:20.580 --> 01:49:41.520
Lada Kesseler: Right? And so the thing is, with Focus, I think the industry understood it, because… so Entropic had MCP service, right? And they seemed to be a good idea, except they really, really, really, really didn't understand context

management, because they were preloading everything about something into the context, so then the agent is always distracted, and the more, like, MCPs you have.

650
01:49:41.520 --> 01:49:42.950
Lada Kesseler: The worse the problem is.

651
01:49:42.950 --> 01:50:07.219
Lada Kesseler: So the skills that they just released, and I think everybody's adopting now, are actually getting it really, really well. So this is a combination of knowledge documents plus a mechanism for focusing it. So, like, I can pull things… it automatically can pull things that are important. As you activate the skill, you get the, like, a main file and then references, and it will pull those references when they're needed. Basically doing the same thing that

652
01:50:07.220 --> 01:50:18.979
Lada Kesseler: you, I was doing, right, as a person, to kind of manage the context, but now it's kind of automatic. But again, still relevant, despite tooling adopting it, because you'll find places where this is not doing it, yeah?

653
01:50:19.210 --> 01:50:20.520
Lada Kesseler: Alright, so…

654
01:50:20.790 --> 01:50:39.570
Lada Kesseler: again, we started with knowledge library, building knowledge library. How do you find out how to create those documents? What's missing? You reverse the action and ask it, and partner with it. And now you have focus, right? So, knowledge library is growing, but more is not better. You find that you actually get worse results suddenly.

655
01:50:39.570 --> 01:50:48.079
Lada Kesseler: Why is that? Because there's an attention, right? And you need to kind of focus your agent to get good results. And dumping everything doesn't help. So…

656
01:50:48.150 --> 01:51:11.850
Lada Kesseler: But then the next question is, like, what if all the knowledge that is actually… like, the task is so big that all the knowledge that I give it is actually relevant, it's just the task is enormous, right? What do you do? So, here we have all this stuff, right, we just add it because it's all relevant, like, we're trying to focus, but the task is big, and then we pivot a giant task, and it goes away for a long time.

657
01:51:11.850 --> 01:51:15.550
Lada Kesseler: And you probably get something that's not working, right? It depends what you're doing, but…

658

01:51:15.550 --> 01:51:18.259
Lada Kesseler: If you're just… just trying to one-shot this giant thing.

659
01:51:18.610 --> 01:51:22.470
Lada Kesseler: you're likely to get something wrong. So, what do you do?

660
01:51:22.760 --> 01:51:30.949
Lada Kesseler: Same idea as what us developers or humans do in complex situations. You manage complexity, you chop it, like, how do you do an elephant, right?

661
01:51:31.140 --> 01:51:42.760
Lada Kesseler: Yeah. So, chop the problem up into small pieces, and this can be done with AI different ways. So, first of all, you can just… if the tasks are independent.

662
01:51:42.760 --> 01:51:45.969
Nancy Nickel: It feels like dating. It really does. Like, we…

663
01:51:47.390 --> 01:51:49.869
Lada Kesseler: Alright, if the task is independent, then…

664
01:51:50.090 --> 01:52:03.260
Lada Kesseler: basically, you can just do them all in parallel, or, like, just, like, it doesn't matter which one is done at what point, and then combine, right? But here, each task gets its own dedicated knowledge, and it's smaller, right? And then you can kind of…

665
01:52:03.260 --> 01:52:10.540
Lada Kesseler: see what the result is. Maybe even here, it's nice that you can, like, even re-roll the die a bit and redo this piece if it failed.

666
01:52:10.540 --> 01:52:27.809
Lada Kesseler: And, like, helpful. And then you kind of… if they're not independent, you can start kind of doing them in chunks, right? So you start with a small piece, and then give it only the information here, then… and you give it a little bit of results, and you like the result. Then you go and give it a bit more information than the next task.

667
01:52:27.950 --> 01:52:31.220
Lada Kesseler: And so on. And here, it's interesting, because… so…

668
01:52:31.370 --> 01:52:41.159
Lada Kesseler: This will be a bit better than, like, this, because here, somewhere, like… does it remember this, like, blue thing over here? Over there?

669
01:52:41.160 --> 01:52:54.819
Lada Kesseler: Maybe, but it's likely to forget. But here, it's less likely to forget, because it's in place, and AI has this recency bias, right? It's more likely to forget this bit at this step, and this bit, because this bit is just more recent.

670
01:52:54.820 --> 01:53:12.299
Lada Kesseler: And so on. And so you kind of go in chain like this, and this is what I call chain of small steps, right? You just simply manage complexity. But it goes a long way between, like, getting good results, especially if you're trying to do a big thing. One-shotting is going to stop working, right? And this is what we have to do.

671
01:53:12.770 --> 01:53:36.960
Lada Kesseler: All right, and one thing that gives you this chain of small steps is steerability, which is kind of priceless, because… so you get feedback, right? You get quick feedback. You have this task, you did some things, you suddenly hear you have something bad. First of all, you probably have a checkpoint here, so you can re-roll, you can bet a result, but also you can steer, you can tweak, and you can maybe even go here and adjust things here.

672
01:53:36.960 --> 01:53:49.799
Lada Kesseler: So now you have much more, like, ability to control your result, and, like, as soon as you get bad results, you can actually adjust, as opposed to, like, oh, the whole thing is done now, all of it is garbage and wrong.

673
01:53:49.800 --> 01:54:09.229
Lada Kesseler: So, and you can do it in one context or several, so here we have this one context, and this is the next one starting from a different thing. Again, you're working in small chunks, so you can kind of split them up a lot, and there's a lot of value in chunking them separately into separate contexts, because the context is not universal, and here you basically get

674
01:54:09.230 --> 01:54:17.970
Lada Kesseler: So this… this yellow piece is very likely to get… it to get from here at the end. There's a concept, a context chart that goes with this.

675
01:54:18.070 --> 01:54:29.950
Lada Kesseler: All right, so you can continue in a new context. So this is the idea of chain of small steps, just manage complexity. If everything is relevant, you probably have just too big of a thing. Chop it up, or, like, you probably won't get good results.

676
01:54:30.220 --> 01:54:31.790
Lada Kesseler: So, again.

677
01:54:31.830 --> 01:54:50.839

Lada Kesseler: building knowledge library, we find out what we're missing, as… as we work actively with AI and get feedback from it. We're focusing the agent because dumping everything into Asian doesn't help, it actually scatters the brain, and now we understand that there's some tasks that are so big that you kind of need to manage complexity, right?

678
01:54:51.750 --> 01:55:02.809
Lada Kesseler: Alright, and as you're kind of managing complexity and doing more things, it's likely that as you develop more features, the size of the documents you have starts to grow.

679
01:55:03.060 --> 01:55:08.260
Lada Kesseler: Because you need to add to them, right? As new features change, you're trying to maintain the documents they're creating.

680
01:55:08.420 --> 01:55:19.439
Lada Kesseler: And suddenly you have a question of, like, why… wait, wait, it's too big, it's, like, 2,000 lines already. How do I maintain this? And also, maybe some information there is outdated. Oh no, what do we do?

681
01:55:19.520 --> 01:55:34.660
Lada Kesseler: Right? So, how do you maintain the documents? So one thing that AI has that is really, really problematic, and people cannot tolerate it for some reason, I think they may be just not aware that they don't have to have this pain. So, I go to AI and like, hey.

682
01:55:34.760 --> 01:55:37.479
Lada Kesseler: Something simple, small thing, right? Hey.

683
01:55:37.650 --> 01:55:42.070
Lada Kesseler: ask a small question, and I got this as a response.

684
01:55:42.550 --> 01:55:52.729
Lada Kesseler: questions, 10 questions, and a lot of noise, and a lot of text. And so many people go and answer every one of those questions in there, and just… I'm just, like.

685
01:55:53.170 --> 01:56:06.420
Lada Kesseler: you know, have a lot of compassion, because I don't tolerate this stuff, and you don't have to either. What I do here is, like, ask me only one question at a time, right? And this is much, much easier to work with. Now you can have a back and forth.

686
01:56:06.490 --> 01:56:23.730

Lada Kesseler: So, from noisy, force it to give it the noise in chunks if you have to, if you have to answer all those questions. Another thing that is manifesting is, again, you ask something simple, you have this page of stuff, maybe several pages of stuff, really, really noisy.

687
01:56:23.730 --> 01:56:36.749
Lada Kesseler: And I just default to much more succinct plays everywhere, like, much more succinct, please, because this is actually super scannable, and I find that this noise is really in the way and is not adding much, and when I have the cure, it's kind of defaulting to

688
01:56:37.180 --> 01:56:45.640
Lada Kesseler: the level that is above the level of my curiosity, like, too much detail, right? It's like, I'm so thorough, I'm trying to be so helpful, so here's all information humor.

689
01:56:47.390 --> 01:56:58.250
Lada Kesseler: scaling it down and asking it to be more succinct goes a long way, and this makes such tremendous difference in documents, it's not even, like, it's… it's just nuts. So…

690
01:56:58.250 --> 01:57:10.669
Lada Kesseler: how… so you have a new capability, you can zoom in and out of text. AI produces text. Make it produce higher-level text. Make it produce succinct, shorter text. If you need more detail, you can always ask the token machine to produce more.

691
01:57:10.760 --> 01:57:14.460
Lada Kesseler: Right? So I basically say, make it shorter, make it more succinct.

692
01:57:14.460 --> 01:57:37.509
Lada Kesseler: high-level architecture, TLDR, everywhere. And this goes into my CloudMD and everywhere, and it makes a lot of difference. And maintainable, like, how do you apply this to documents? So it makes the meaning scannable, but also in documents, what you do is you basically say, hey AI, take this giant doc, go through everything in the code, and double-check what's relevant, high level, make it extremely succinct, only the most important information.

693
01:57:37.510 --> 01:57:45.640
Lada Kesseler: If you need something, if you find out that you need something, you can always get it back. And you should be relying on Git, so you've stored everything anyway, so…

694
01:57:46.050 --> 01:58:01.569
Lada Kesseler: I think people are reluctant to delete, but we kind of need to fight this tendency with AI, because if one AI… you allow one AI to be noisy, if you're in

a group of five developers, five noisy AIs is just not maintainable. You're gonna drown in noise.

695
01:58:01.900 --> 01:58:06.409
Lada Kesseler: Right, so this is the idea of noise cancellation and how to make the documents more maintainable, right?

696
01:58:07.040 --> 01:58:20.589
Lada Kesseler: So… so we go from building knowledge library, to being active partner with AI to figure out how to… what knowledge it's missing, to… we can't just dump everything into it, to…

697
01:58:20.640 --> 01:58:37.350
Lada Kesseler: on large things, we need to take small steps, and now, like, the verbosity, right? How do we make maintain… things maintainable in documents? How do we kind of not… because if you don't do this, like, we're gonna undermine this step and this step, right? We need to cancel the noise, otherwise it's gonna drown us in it.

698
01:58:37.440 --> 01:58:53.109
Lada Kesseler: And this last bit, I'm just gonna cover quickly and end. So we find that, so we're building some processes, right? Some documents that have some repeated steps, and some of those repeated steps may actually be

699
01:58:53.110 --> 01:59:03.599
Lada Kesseler: Something that is always the same, and also can be scripted. So, one tendency I see is, once you have AI, you cannot, you know, use AI for everything.

700
01:59:04.020 --> 01:59:22.760
Lada Kesseler: And the thing is, all the tooling that it uses is kind of going into context and distracting the agent. So… and plus, if you ask AI to do something once, and it worked once, the second time it might not work, right? Because it's not a domestic system. You're dealing with something that's non-domestic.

701
01:59:22.820 --> 01:59:38.790
Lada Kesseler: And sometimes it can help with the noise if you just put it into different contexts, so do this noise lingering over there. But even better is just relying on the tools that are meant for this, for things that can be scripted. So I would say, basically.

702
01:59:38.790 --> 01:59:50.060
Lada Kesseler: So now, the task, right, you have three steps, but one of the steps becomes run the script. Instead of, like, making AI do this whole thing, the script does it, and the script is going to be reliably doing it.

703

01:59:50.060 --> 02:00:07.789
Lada Kesseler: So, I would say script everything is scriptable. I call it a flow-deterministic behavior of AI. Basically, anything that can be not AI, prefer this to be not AI, because that will be much better results. Your AI bits will work better, but also your whole overall system will work better, because this is kind of…

704
02:00:08.210 --> 02:00:21.839
Lada Kesseler: the code is good at reliable stuff, and if you're doing math, please don't do math with AI. AI is really bad at math. Ask it to do a script, and guess what? Yeah, AI can write those scripts, and they're pretty good, right? And produce realistic results. All right, so…

705
02:00:21.840 --> 02:00:35.490
Lada Kesseler: And the last bit is non-digitemurism. So, just to remind you the whole thing, and we're gonna wrap up. So, yeah, it's missing some knowledge, right? So, you kinda wanna… to be more powerful, you can build those knowledge libraries that you'll rely on.

706
02:00:35.670 --> 02:00:46.049
Lada Kesseler: And you can find out what knowledge it's missing by kind of reversing direction and asking it questions, and asking it to show its internal state to you so you can align the mental models.

707
02:00:46.070 --> 02:01:01.469
Lada Kesseler: as you grow libraries more, you see that, like, just adding everything into context is not better, because the agent gets distracted, so this idea of focus, so you have to be very explicit, and focusing the agent goes a long way to performance. And then managing complexity.

708
02:01:01.470 --> 02:01:06.840
Lada Kesseler: Basically, when you have big tasks, like humans, you want to chop it up, and that helps with focus too.

709
02:01:07.010 --> 02:01:24.500
Lada Kesseler: And then, to maintain the documents and just, you know, sanity, yeah, sanity, cancel verbosity, because it's too noisy, it will bloat everything, and just, don't let it. And it's pretty easy to not let it. And then, everything that can be non-AI, default to non-AI, because…

710
02:01:24.500 --> 02:01:34.390
Lada Kesseler: you know, code is good at code, and AI is very good, but, like, using it everywhere doesn't really… doesn't really help. Like, it's better… better to use it for things that it's good at.

711
02:01:34.760 --> 02:01:54.090

Lada Kesseler: That's… that's it. Here's the resources slide. So there's, like, you could find some of the other talks, so I have a much deeper talk and much, much more pilots talk on this subject. You can find it in resources if you're curious. And other than this, please, connect on LinkedIn, and I hope it was helpful. If you have any questions, I'm happy to answer any.

712
02:01:55.130 --> 02:01:58.099
David Gijsbers: Lotta, we do have a couple of questions,

713
02:01:58.680 --> 02:02:07.129
David Gijsbers: Renat, your question is a little bit long, so if you're on the line, do you want to unmute yourself and ask a question live?

714
02:02:07.130 --> 02:02:11.960
Rinat Sergeev: Oh, hello, thank you, thank you for inviting me for the question. So, my question is that,

715
02:02:12.930 --> 02:02:21.920
Rinat Sergeev: There is always a trade-off between making the agents of, like, more specialized and smaller one, like, targeting the smaller components of the knowledge and smaller

716
02:02:22.240 --> 02:02:25.619
Rinat Sergeev: components of functionality is that, you simply have to

717
02:02:25.860 --> 02:02:27.220
Rinat Sergeev: I have to use more of them.

718
02:02:27.620 --> 02:02:47.419
Rinat Sergeev: So, it's sort of like you can compare it with your, like, garage toolbox, and so it's like you can make a very specialized tool for, like, every single purpose, but then you may have to have a lot of those small tool sets, tools, and then when you actually need the tool that you need, you will still take a screwdriver and a hammer and just, like, you know.

719
02:02:47.460 --> 02:02:49.459
Rinat Sergeev: Instead of, like, going and,

720
02:02:49.560 --> 02:02:56.460
Rinat Sergeev: spreading attention to the others. So, how do you deal with the trade-off that you actually have to manage a lot of agents?

721
02:02:57.160 --> 02:03:01.340
Rinat Sergeev: without the… A dilution of the attention.

722
02:03:02.370 --> 02:03:15.450
Lada Kesseler: I think it's a great question. So, I think we're actually figuring out right now, it's the most interesting problem, one of the most interesting problems out there, I think, because orchestration, right? You talk about orchestration agents at the end. That's right. Yeah, so I think,

723
02:03:15.450 --> 02:03:29.009
Lada Kesseler: I'm figuring it out, honestly. Like, I don't know, the industry is figuring it out. I find that I like the approach of, like, I have the kind of master agents spawn other agents, and they do things, and I have dedicated skills for each of some of them.

724
02:03:29.010 --> 02:03:35.550
Lada Kesseler: And dedicated skill that helps this top agent kind of master the others. So I'm starting to kind of…

725
02:03:35.670 --> 02:03:38.909
Lada Kesseler: Delegate to AI to orchestrate for me, a lot.

726
02:03:39.010 --> 02:03:46.080
Lada Kesseler: And that helps a lot, but I think we're just still figuring it out. I absolutely agree, like, that's a problem, and that's probably the next problem we're trying to solve.

727
02:03:46.080 --> 02:03:51.409
Rinat Sergeev: I can also see a big part of that problem coming from multiple suppliers of those agents.

728
02:03:51.560 --> 02:04:04.460
Rinat Sergeev: So, like, when not all of them are equally standardized. So, if you are the only one supplier, you can standardize them, you can build a beautiful orchestrate on the top, where you can easily, like.

729
02:04:04.870 --> 02:04:09.090
Rinat Sergeev: Test that agent, test its efficiency, usability, maybe, like,

730
02:04:09.400 --> 02:04:19.940
Rinat Sergeev: disable it, or, like, retie it when it's needed, and so on. But when we're getting into the world, when everyone can provide with some engine.

731
02:04:20.430 --> 02:04:26.279
Rinat Sergeev: And then, Making, like, a unified orchestra… orchestrator may be quite tricky.

732
02:04:27.280 --> 02:04:29.689
Lada Kesseler: Oh, yeah, absolutely. I think it's hard, it's a hard problem.

733
02:04:30.330 --> 02:04:33.349
Lada Kesseler: It's like… it's a little bit like microservices reminds me of that, right?

734
02:04:35.640 --> 02:04:37.949
Rinat Sergeev: Yeah, same problem there, yes.

735
02:04:39.360 --> 02:04:42.279
Rinat Sergeev: But in this case, the space is probably even more,

736
02:04:43.040 --> 02:04:44.760
Rinat Sergeev: More divorced, in that sense, so…

737
02:04:44.760 --> 02:04:50.450
Lada Kesseler: Yeah, and it changes so much, right? So I think we're gonna find out a lot of this stuff in the next year, hopefully, so… yeah.

738
02:04:50.450 --> 02:04:51.350
Rinat Sergeev: That's go back to work.

739
02:04:51.350 --> 02:04:52.080
Lada Kesseler: question?

740
02:04:52.570 --> 02:05:00.080
David Gijsbers: And the next question we had was from Rito. Rito, if you're on the line, do you want to, ask the question?

741
02:05:09.180 --> 02:05:16.219
David Gijsbers: Alright, perhaps Rito stepped away. The question was, how do you use Obsidian in your AI development workflow?

742
02:05:16.560 --> 02:05:19.109
Lada Kesseler: Do you use it for the knowledge library part?

743
02:05:19.650 --> 02:05:27.929
Lada Kesseler: I used to use, not for the knowledge library, but I do use AI with Obsidian quite a lot, and I have things like

744
02:05:28.500 --> 02:05:29.510
Lada Kesseler: I have, like…

745
02:05:29.620 --> 02:05:37.400
Lada Kesseler: notes where, like, I have a folder where I can just dump things that are coming through throughout the day, and I have, kind of, rules where goes where, and AI, kind of.

746
02:05:37.400 --> 02:05:49.999
Lada Kesseler: takes that, organizes it for me, because I'm really bad at organization, but I found that really helps with keeping things maintainable. I don't tend to do it for coding bits, right? So for good… for specifically for… I think the question was around

747
02:05:50.000 --> 02:06:02.510
Lada Kesseler: knowledge library, right? So this is where I have my own repo… I basically just use repositories, and I also have, like, a factory, a skill factory. You can find it on my LinkedIn, it's one of the recent posts, so basically.

748
02:06:02.510 --> 02:06:21.520
Lada Kesseler: keep things like that in the repository, and I kind of sim-link that to the… I simlink the scales to the… to where I need them, so I can still have repository, it's… it's stored where I need to be in the repo, but my agent can have it, and other people's agents can also do the same thing. That was helpful, yeah.

749
02:06:23.370 --> 02:06:27.270
David Gijsbers: Alright, awesome. Well, thank you very much, Lada.

750
02:06:28.260 --> 02:06:31.140
David Gijsbers: Round of applause. Virtual round of applause.

751
02:06:31.250 --> 02:06:36.919
David Gijsbers: And next up, we have our pinch hitter,

752
02:06:37.200 --> 02:06:48.399
David Gijsbers: Zachy, thank you very much for stepping in for our sick presenter. Why don't want to hand over the microphone and the screen share to you so that you can tell us about Agentix.

753
02:06:49.920 --> 02:07:08.189
Zaki Medina: Yes, welcome everyone. This is very last minute. I only found out 2 hours ago I was doing this, so apologies. I don't have much slides prepared, but I will show you something that's very interesting and top of mind for you. I'll give you a quick 2-minute background about myself here.

754
02:07:08.190 --> 02:07:12.030
Zaki Medina: So, as I share my screen here, let me know if you see my screen.

755
02:07:12.360 --> 02:07:21.179
Zaki Medina: So… My company is Revive Healthcare Group. We supply… we are building the first agentic supply chain in healthcare.

756
02:07:21.250 --> 02:07:38.980
Zaki Medina: And, just a quick note about me, I've been, several… I've spent the last 18 plus years as a builder, entrepreneur, wore many hats, probably all the hats that you can think of in the tech world, from building data centers to software dev to…

757
02:07:38.980 --> 02:07:44.670
Zaki Medina: AI, ML, to… yeah, so pretty much everything. I've worked in regulated industries.

758
02:07:44.770 --> 02:08:01.989
Zaki Medina: And, my current organization, we began our journey a few years ago as a wholesaler and a medical distributor, and we've taken that one step further, and we are solving the most difficult problem is medical supplies. So, believe it or not.

759
02:08:02.060 --> 02:08:21.640
Zaki Medina: Hospitals today, 20% of all the procedures in a hospital end up being canceled or rescheduled by a hospital because they don't have medical supplies. It's an ongoing problem, it affects everyone. You know, as I'm… as I have elderly parents, I'm sure some of you do too, you bear the brunt of that.

760
02:08:21.640 --> 02:08:41.550
Zaki Medina: As you go in. So, our goal was to solve the problem, and to 10x when we solve it, meaning we want to 10x the problem. So, we've been around, we've created our own platform, it's being deployed in several health systems, we can solve 98% of the fulfillment gaps in the supply chain.

761
02:08:41.580 --> 02:09:00.100
Zaki Medina: across U.S. and Canada. Now, what's interesting is we are… we are building Agentic. We use Agentic, and we build Agentic as well, so we eat our own dog food. Very small team, but today I will show you my personal operating system on how I build this out. I have no slides prepared.

762
02:09:00.100 --> 02:09:07.110
Zaki Medina: But I'm sure some of you have heard about… probably, let's do this, let's… how many of you have heard about,

763

02:09:07.540 --> 02:09:08.790
Zaki Medina: Open claw.

764
02:09:09.140 --> 02:09:11.989
Zaki Medina: or CloudBot, or MoldBot.

765
02:09:12.350 --> 02:09:25.269
Zaki Medina: And, you know, I think it's top of mind for some of you. Yes, David, you've probably heard about it. So, I consider myself a very, experienced user of this, and I will kind of lay out

766
02:09:25.340 --> 02:09:44.879
Zaki Medina: how I've built my own thing. I'm not gonna go into code or anything, but I'm going to show you. So, as a CTO running a healthcare AI startup, I have a lean team of engineers and a few other divisions, because I also wear the COO hat. And, when you're building for healthcare, you have to build fast.

767
02:09:45.000 --> 02:09:56.740
Zaki Medina: And it can't be broken. It's kind of a fallacy, as you think about it. So, I chose to build it, this way. I have used OpenClaw.

768
02:09:56.740 --> 02:10:08.920
Zaki Medina: I am using something else now, that is proprietary. It's similar to OpenClaw, but there is a version of OpenClaw called Pi, which is built on, Go.

769
02:10:09.000 --> 02:10:20.739
Zaki Medina: And then there is a third version of it, let's call it that, and let's call it in stealth mode. But for all purposes, this is open claw. If you are living under a rock, open claw.

770
02:10:20.980 --> 02:10:32.659
Zaki Medina: Took the world by storm in the last 30 days. It's, you know, it's an open-source agent platform. It crossed 100K GitHub stars, in less than a month.

771
02:10:32.790 --> 02:10:39.540
Zaki Medina: Its founder, exited for $100 million, and probably got a multi-billion dollar package in one month.

772
02:10:39.550 --> 02:10:56.769
Zaki Medina: from OpenAI, right? So, just FYI, again, I don't know if he got that kind of cash, but I'm pretty sure if Meta and OpenAI was after him, and he signed up with OpenAI, I'm pretty sure it was in that realm. So, this is…

773
02:10:56.770 --> 02:11:07.080

Zaki Medina: So, I'm not going to talk about what is open claw overview. Those exist. This is more step-by-step, and I'm literally going to build the slides. I didn't want to generate something on

774
02:11:07.080 --> 02:11:16.100
Zaki Medina: AI and then show it to you guys. I'd rather build it with you guys in the next 40 minutes. I do have my other environment I'm looking at. I don't want to show my environment, because it has

775
02:11:16.170 --> 02:11:31.570
Zaki Medina: data that should not be shown, because I work in healthcare, and obviously I'm in stealth… kind of in stealth mode, but I will show you my operating system. And for those of you, you can leverage this operating system as a major personal productivity hack. So.

776
02:11:31.820 --> 02:11:41.830
Zaki Medina: So I'll talk about my journey. My journey started with, basically, like, like, I literally had only one,

777
02:11:42.730 --> 02:11:54.499
Zaki Medina: Let's get some shapes here. I literally started with one agent, you know? One agent. It wasn't doing much. And most people start with this one assistant. You probably created something off of codecs.

778
02:11:54.500 --> 02:12:04.849
Zaki Medina: sorry, with ChatGPT, you've created a program, or you've created… in Gemini, you've created a gem. That's typically what most people do, they'll have one assistant.

779
02:12:04.860 --> 02:12:27.140
Zaki Medina: And one assistant… when I started out, yes, I had one assistant. It would write code, check email, draft compliance reports, right? Review pull requests. It worked for about a week, but then it died. And the problem was not capability, it was context contamination. So one… one assistant, or one agent, handles multiple different

780
02:12:27.140 --> 02:12:32.959
Zaki Medina: domains. In my case, it was engineering architecture, and marketing copy, and regulatory compliance.

781
02:12:32.960 --> 02:12:56.709
Zaki Medina: it was difficult, because the agent ended up carrying no specialized context. So you need specialized context to make any AI agent work, and to make the domain work very well. So, like, my compliance agent needs to understand IEC, ISO standards, SOC 2, all of that wonderful stuff, but my engineering agent needs access to my GitHub.

782
02:12:56.710 --> 02:13:20.409
Zaki Medina: my deployment patterns, and so on. Asking one agent to context switch between these two is like asking your head of engineering to also run marketing. So, we know what the answer is around that. So, essentially, context bleed, this is a topic, maybe we've invented a new term, destroys quality, right? So, the first instinct is more prompts, better prompts.

783
02:13:20.410 --> 02:13:28.810
Zaki Medina: More prompt engineering. Terrible idea. I used it, I didn't get anywhere. What I needed wasn't more prompts, so I needed an organization.

784
02:13:28.810 --> 02:13:32.299
Zaki Medina: So I decided, listen, I actually don't need…

785
02:13:33.030 --> 02:13:51.989
Zaki Medina: I don't need one agent, I need to move away from one agent to several agents, right? So I started creating, you know, I ended up having about 13 or so agents, you know? 13 agents, and I'll talk a bit about what my 13 agents ended up doing, right? As a whole. So,

786
02:13:53.020 --> 02:13:54.430
Zaki Medina: So, essentially.

787
02:13:54.710 --> 02:14:09.419
Zaki Medina: each of these, you know, I started with one to three agents, yes, two, three domains. I obviously divided my marketing, my engineering, and my compliance, right? As an example, that didn't work. I mean, it worked, but I felt like I could really, really go

788
02:14:09.420 --> 02:14:18.299
Zaki Medina: faster and more, you know? So I added 7 more agents this time. Now I've given it multiple different business functions. Then I went

789
02:14:18.410 --> 02:14:25.219
Zaki Medina: to eventually 13 agents, because I decided, listen, I really want to spend more time in the golf course.

790
02:14:25.220 --> 02:14:41.320
Zaki Medina: than coding, so I decided, let's do this experiment, you know? My CEO is out of town, he's on vacation, and I'm gonna pretend that I'm doing so much work, but it's really my agents, and we're gonna see what happens. So, the first thing I did was

791

02:14:41.360 --> 02:14:50.189
Zaki Medina: Phase 1. I needed 13 agents. When you download OpenClaw, and I'm sorry to make it like an OpenClaw, which is an open source thing.

792
02:14:50.200 --> 02:15:02.239
Zaki Medina: Some of you may be coming on, you know, are familiar with it and probably running it. So, out of my 13 agents, before I go into that, I'm going to talk a bit about what I first did. I first built out

793
02:15:02.320 --> 02:15:21.249
Zaki Medina: what I call is the soul. When you deploy OpenClaw, it gives you 5 workspace files, right? So these five workspace files are essentially… consider this the soul. So you have what is called a soul MD, and then you have what is called a… actually, let's use the proper form here.

794
02:15:21.310 --> 02:15:25.129
Zaki Medina: So, in case somebody is recording this…

795
02:15:30.380 --> 02:15:37.219
Zaki Medina: So, basically, I'm building my second brain, guys, like, kind of in real time, and I could build this in real time if I had another hour.

796
02:15:37.400 --> 02:15:47.249
Zaki Medina: It took me… took me a day, I would say, a day and a half to build this. So I'm just gonna list out all of these, and I'll explain what they all do.

797
02:15:47.390 --> 02:15:51.120
Zaki Medina: It'll give you real practical input, Keith, you know.

798
02:15:51.750 --> 02:15:52.760
Zaki Medina: of this.

799
02:16:02.970 --> 02:16:04.150
Zaki Medina: the agents.

800
02:16:04.340 --> 02:16:18.720
Zaki Medina: While we're doing this, let's keep this interactive. Has anyone used OpenClaw? Has anyone, as I typed this out, has anyone found anything interesting from using CloudBot, MoldBot, or OpenClaw?

801
02:16:19.040 --> 02:16:22.580
Zaki Medina: Or am I the first person to talk about it to you?

802

02:16:25.170 --> 02:16:28.079
Zaki Medina: Okay, no one's… everybody's shy right now.

803
02:16:30.630 --> 02:16:35.039
Zaki Medina: Alright, so first I give… I create these. These are kind of created by default.

804
02:16:35.059 --> 02:16:51.969
Zaki Medina: My soul is my personality and values. This is not a system prompt. It's a character definition. It's the instruction that changes everything. So I asked my, my version of an operating system, I said, listen, you're my assistant.

805
02:16:52.230 --> 02:17:06.360
Zaki Medina: You need to be nice enough that I want to talk to you at 2 AM. I don't want you to be a corporate drone. I don't want you to be a sycophant. Just be good. Here's some hard rules. Never open with great question, I'd be happy to help. Just answer.

806
02:17:06.400 --> 02:17:17.510
Zaki Medina: Have opinions, strong ones, and stop starting… stop hedging those with, it depends. Brevity is mandatory. If the answer fits in one sentence, just give me that. Swearing is allowed.

807
02:17:17.580 --> 02:17:35.010
Zaki Medina: when it lands, don't force it. If I'm about to do something dumb, say so. You know, charm over cruelty. Don't just sugarcoat anything. So I was very blunt with its soul, and it basically somehow ended up taking the personality of the late Norm Macdonald.

808
02:17:35.010 --> 02:17:38.570
Zaki Medina: He's a very famous comedian, so if you guys go back…

809
02:17:38.700 --> 02:17:56.180
Zaki Medina: see Norm Macdonald, it's basically, him on steroids here. Then I have user. So everything about me, my work schedule, my time zone, my communication preferences, project priorities, pet peeves, the AI will read this for every session and adjust this.

810
02:17:56.350 --> 02:18:05.689
Zaki Medina: Then I give it an identity. Now, the identity is nice, because the AI needs to know who, what it is, right? Its name, its avatar, its role, its tagline, and so on.

811
02:18:05.690 --> 02:18:23.900

Zaki Medina: Then I have agents. These are different procedures, operating procedures. Obviously, I have 13 different agents. I have 13 different of these files that are going on. Basically, what does the agent do autonomously? What permissions do they need? How do they handle group chats versus direct messages? Then I give it memory.

812
02:18:23.900 --> 02:18:46.770
Zaki Medina: Or I give it some aspect of memory, right? And this memory file, this is long-term knowledge. It's curated. I purposely keep… kept it lean. I have another agent, like a 14-day agent, that goes and just updates these things and so on. This may seem complicated, but this is literally a couple of hour setup. It's literally 2 hours to set it up. I set it up for my doctor friend.

813
02:18:46.889 --> 02:18:54.210
Zaki Medina: who runs two clinics, and he basically, does not have any admin staff, literally. AI…

814
02:18:54.209 --> 02:19:08.219
Zaki Medina: is running his scheduling, his bookings, his, you know, a bunch of other things, I'm sure, as I showed it to him two weeks ago. So, it's crazy how non-technical user

815
02:19:08.680 --> 02:19:16.920
Zaki Medina: is using it, and how far they've gotten along. So this is, I think 2026 is the era of personal

816
02:19:16.920 --> 02:19:30.980
Zaki Medina: software, where people are building software for themselves, very customized, very, you know, and so on. Anyway, so… so these files are kind of the very important ones. These are with… these are the soul, right? The soul files. Every session starts with this.

817
02:19:30.980 --> 02:19:39.060
Zaki Medina: And so on. So, if you want consistency across your agents, you're going to have to give it something, some kind of…

818
02:19:39.120 --> 02:19:46.039
Zaki Medina: you know, context to what it is, what it should be doing, and so on. And so on.

819
02:19:46.070 --> 02:19:51.050
Zaki Medina: Now, when we first opened this file, the SolMD in OpenClaw workspace.

820
02:19:51.050 --> 02:20:15.980

Zaki Medina: Again, this is not a coding thing, although if we had more time, maybe we could have made it a coding thing, but we… we typically have to play around with things like tone, values, priority order, anti-patterns, what not to do, what to do, what's the emotional range, right? Be helpful is useless, you know? I'm sure a lot of you probably are starting to hit that with Anthropic and Claude, right? Claude, by default, is designed to be helpful.

821
02:20:15.980 --> 02:20:32.249
Zaki Medina: I think is useless. I would say always lead with an answer, then explain if needed, never the reverse. So that's how I've trained my, my large language models to behave, and so on. And the next part, phase two, is this org chart.

822
02:20:32.540 --> 02:20:48.929
Zaki Medina: Now, this assistant, I call it an assistant, but underneath the assistant are 13 agents, right? So, I have, you know, 13 agents running under this, right? This agent. I didn't start with 13, as I mentioned, I started with 3, and then I expanded. My Tier 1.

823
02:20:48.930 --> 02:20:53.630
Zaki Medina: My Tier 1 agents, okay? So, my Tier 1 agents…

824
02:20:56.210 --> 02:20:58.050
Zaki Medina: I have a list of them.

825
02:20:58.340 --> 02:21:03.910
Zaki Medina: I have a few of them. So, I basically have what is called,

826
02:21:04.200 --> 02:21:18.829
Zaki Medina: I hired probably a QM first, quality manager, because initially, I needed someone to deal with the regulatory compliance stuff that was eating into my weekdays. If you're in healthcare, you understand that. Next up, I hired…

827
02:21:19.340 --> 02:21:25.550
Zaki Medina: kind of a chief of staff, you know? Chief of staff. Why? Because AIs are really bad at

828
02:21:25.550 --> 02:21:44.229
Zaki Medina: you know, generating emails, texts, things like that. So I had to create one dedicated, give it a communications degree, and then I made it… get it… gave it access to my Gmail, my Slack, my WhatsApp group, my Telegram. We also use Teams as well, and Outlook, so we have all of that.

829
02:21:44.330 --> 02:21:50.600
Zaki Medina: Then, next up, I gave it, sorry, that was 1, 2, okay, 2…

830
02:21:54.790 --> 02:22:07.160
Zaki Medina: Oh, I forgot my most important agent. I have my commander. So, this is the agent, the manager agent, you know? As an example. And then… I am gonna…

831
02:22:07.320 --> 02:22:09.380
Zaki Medina: My numbering is off sometimes.

832
02:22:10.220 --> 02:22:14.390
Zaki Medina: And then… is this helpful, guys? Give me a thumbs up if you think this is helpful.

833
02:22:14.880 --> 02:22:19.110
Zaki Medina: We can… we can talk about other things. Talk about the Seahawks.

834
02:22:19.290 --> 02:22:21.700
Zaki Medina: And the Patriots.

835
02:22:21.950 --> 02:22:25.640
Jess Wolfe: I'd rather talk about this than the Seahawks and the Patriots.

836
02:22:25.640 --> 02:22:34.339
Zaki Medina: Okay, I appreciate that. I'm on the fence with both, you know? So, I have family on both sides, so…

837
02:22:34.970 --> 02:22:53.529
Zaki Medina: Awesome. So, okay, so now I decided, alright, I need my engineering, right? So I got my CTO, CTO, I'll call it a CTO, although I am the CTO, but let's say this is my engineering guy, makes architecture decisions, then I have Aurora. Okay, let's do this.

838
02:22:53.530 --> 02:22:55.259
Zaki Medina: Sorry, give me one second.

839
02:22:55.610 --> 02:22:59.189
Zaki Medina: I want to make this proper, because I will come back and refer to this.

840
02:23:00.070 --> 02:23:02.599
Zaki Medina: So let's call it Commander.

841
02:23:03.710 --> 02:23:11.139
Zaki Medina: Or Captain… And I mimicked this after Captain Picard from the Enterprise, because I am that nerd.

842
02:23:15.350 --> 02:23:21.319
Zaki Medina: And then… And then I have Chief of Staff.

843
02:23:25.190 --> 02:23:28.919
Zaki Medina: So I might be showing you a bit of the future, guys, but it's fine.

844
02:23:31.980 --> 02:23:41.679
Zaki Medina: And then my CTO, Oh, and so we already explained what the commander does, what the…

845
02:23:41.800 --> 02:24:00.379
Zaki Medina: what the chief of staff does, what the CTO does, what the commander does. And then I'm gonna talk a bit about… I might as well do this. My CMO, Chief Marketing Officer. This was basically marketing, content strategy, all of that wonderful stuff.

846
02:24:00.440 --> 02:24:19.960
Zaki Medina: And then I have a product officer, CPO, product strategy, product roadmap, and then I also have a CRO, you know? Cro, so sales, revenue, partnerships, all of that. And then I have, you know, CFO. Yeah, I went a little crazy, I can… you can tell.

847
02:24:22.500 --> 02:24:34.440
Zaki Medina: And then, once I have a CFO, all good. And then, obviously, I'm in the healthcare space. I have a chief medical officer I needed. This was dangerous. This one, I'm still not convinced.

848
02:24:34.800 --> 02:24:43.180
Zaki Medina: you know, C, Chief Med O. You know, CMED O. Okay,

849
02:24:43.360 --> 02:24:56.919
Zaki Medina: So, chief medical officer, so anything clinical, R&D related, because we do have some of that. And again, you can add your domain-specific, if you're in finance, manufacturing, retail, whatever it is, you can have a domain-specific

850
02:24:56.920 --> 02:25:12.599
Zaki Medina: person here. So essentially, my commander is managing this, and then I have my execution tier, right? So this is my C-suite, now I have my execution tier. I actually didn't have many under my execution tier. I basically had,

851
02:25:12.610 --> 02:25:21.450
Zaki Medina: you know, I had a developer that was running under my CTO, you know? Developer personality, and

852
02:25:21.530 --> 02:25:27.730
Zaki Medina: And basically, this is what I ended up building. Then I had was, a project manager.

853
02:25:28.420 --> 02:25:46.040
Zaki Medina: So a project manager would run under PM, I would run the PM under my CPO, as an example, you know? So… and then I would have someone that is content writer or a copywriter, right? Maybe I'll just call it copy.

854
02:25:47.270 --> 02:25:48.870
Zaki Medina: Copywriter.

855
02:25:50.280 --> 02:25:57.730
Zaki Medina: Honestly, if I was prepared, this would have been nicer. Apologies, everyone. Some of you know me, I'm very prepared, but…

856
02:25:58.040 --> 02:26:00.410
Zaki Medina: Okay, and then we have,

857
02:26:00.640 --> 02:26:07.280
Zaki Medina: researcher. I'm sure this is something everyone knows as a researcher. Now, researcher is cloth…

858
02:26:07.380 --> 02:26:22.929
Zaki Medina: across everyone, right? So everybody… everybody plays into this guy, you know? Everybody comes and talks to researcher, you know? So I'm just showing you as an example, but everybody talks to researcher, you know? And obviously.

859
02:26:23.400 --> 02:26:38.519
Zaki Medina: there's one more agent, 14th agent, which does clean up, makes sure that the, the .MD files, the README files, are all nice. Now, when I created this in OpenClaw, and I built this out, the key insight I've learned

860
02:26:38.530 --> 02:26:48.929
Zaki Medina: is an OpenClaw example is very good with Telegram. Telegram is the first one that was built. So if you try OpenClaw, definitely start with Telegram.

861
02:26:49.000 --> 02:26:53.439
Zaki Medina: For me, no one in my family, no one I knew used Telegram.

862
02:26:53.440 --> 02:27:17.359

Zaki Medina: And so, except, some European friends, and anyway, I don't talk to them, because it's too late or too early when I finish work for them, so I'll see them in the summertime. So, other than that, I use Telegram for all my… between me and my agent, my assistant, right? I would have those conversations. And then WhatsApp was for business, Signal, we use all of that wonderful stuff.

863
02:27:17.380 --> 02:27:23.109
Zaki Medina: As well. So, the key insight I learned is use stronger models, like Claude, Opus.

864
02:27:23.200 --> 02:27:41.819
Zaki Medina: GPT-5 for the C-suite. So all of these C-suite guys, I would use, like, the GPT-level models. And these smaller guys here, I would use, like, Cloud Sonnet or Haiku. I would use, you know, maybe some open-source models. I've been playing with some really great Chinese models.

865
02:27:41.820 --> 02:27:56.359
Zaki Medina: Like, Glimm, Kimi, and then DeepSeek. I have, DeepSeek V4 is coming out next week. I've already been beta testing it for them. So, you know, again, just FYI, there's a question now.

866
02:27:56.360 --> 02:28:04.380
Zaki Medina: Zachy, what are your API costs like? So, I'm gonna be very blunt with you guys, my first

867
02:28:04.690 --> 02:28:18.019
Zaki Medina: So, before this open claw existed, I was trying to do this with other tools. My cost was not great. My first month, I ended up spending four grand.

868
02:28:18.230 --> 02:28:36.330
Zaki Medina: And, I learned my lesson, and I actually, figured out what not to do. And what not to do was I actually started with open source models that were cheaper and easier to use, so I stopped using Claude, stopped using OpenAI. But when OpenClaw came around, I decided, listen.

869
02:28:36.330 --> 02:29:00.240
Zaki Medina: I'm going to deploy this, and then in OpenClaw, there is a way, they call it the Claw Router. It's another plugin that you can download, and that Claw Router will make sure that you can hard-set budgets and things. So right now, my budget monthly is about $200 a month, right? To run all 13 agents. It hasn't been a month, but it's getting close to that range right now. So, I'm just giving you kind of a…

870
02:29:00.240 --> 02:29:03.930
Zaki Medina: an ideal spot. So anyway, For you guys.

871
02:29:03.930 --> 02:29:22.930
Zaki Medina: Obviously, I'm showing you this is overkill. Most people, the doctor, my doctor friend, he ran probably two, three agents, right? That's all he needed. He needed a booking agent, he needed someone to do his SEO, and he needed someone to do the, you know, the communications, right? The emails and all of that stuff, right?

872
02:29:23.350 --> 02:29:48.279
Zaki Medina: And then… and then work, and so on. So he just really needed that. Now he apparently told me he's adding a few other agents, like, he asked me, should I fire my accountant? Should I… like, I don't know, I'm not in the position to answer that for him. I said, listen, I still have my accountant, and I think you should too. So, I don't think that's happening anytime soon. So, anyway. So, phase three, this is the boring stuff.

873
02:29:48.280 --> 02:30:12.059
Zaki Medina: So, after I got all of this set up, obviously, agents sitting around idly were pretty worthless, and, you know, so the real power is scheduled automation. And this is where I spend most of my time. So OpenClaw creates this whole thing around, you know, when it creates a job, it stores it, right? It has its own built-in job, you know? I think Chrome jobs, I think you guys

874
02:30:12.060 --> 02:30:19.780
Zaki Medina: If you remember, which is a job scheduler, right? So, if you go to a slash open clause slash crone, you'll see every

875
02:30:19.780 --> 02:30:24.159
Zaki Medina: session that it runs, it's an isolated session. That means it starts fresh.

876
02:30:24.160 --> 02:30:48.029
Zaki Medina: does the work, delivers the results, without polluting the main conversation, and so on. So, I have many, many Chrome jobs, I won't call… so, let's say 7 o'clock. My typical day starts, 7 AM, there's a communication triage. Basically, all of these guys come in, and they are trained, you know, I've trained them well, to go through my Outlook, go through my Gmail.

877
02:30:48.200 --> 02:31:09.130
Zaki Medina: scan everything, look at our tickets, any support tickets, anything that's super urgent, high urgent, I get that up front, right? Then, at 7.15 in my day, I get all my, some of my tech teams are in Europe and other parts, so they're probably already working last several hours, so,

878
02:31:09.340 --> 02:31:16.170
Zaki Medina: I look at the tickets, I look at any blocked items, overdue reviews, and then so on. And then at 7.30,

879
02:31:16.290 --> 02:31:40.679
Zaki Medina: I have my daily co-pilot, my commander comes in and says, listen, here's the briefing, based on your calendar. It provides me a Kanban board of all the things that are going on. I always train it to give me the top 3 things that I need to focus on, that I cannot delegate, and I do that. It already has my emails from… that I missed yesterday, or between yesterday and today, as my day started.

880
02:31:40.680 --> 02:31:47.570
Zaki Medina: already in the draft mode, so I can go and review those and send those out. So I do a bit of that, and then 8 o'clock.

881
02:31:47.570 --> 02:31:53.349
Zaki Medina: I… I… my mornings are engineering, and then my afternoons are operations, so at 8 o'clock in the morning.

882
02:31:53.350 --> 02:31:58.869
Zaki Medina: I look at open PRs, CI failures, issues assigned to me, and so on. And then.

883
02:31:58.870 --> 02:32:18.829
Zaki Medina: The next time I look… I get anything is, like, an evening digest, like, at 7pm my time, as I'm, like, mindlessly scrolling the next episode of Night Manager here. And so, it's giving me the day's accomplishments, pending decisions, and tomorrow preview, right? And so on. So, that's kind of my engineering

884
02:32:18.830 --> 02:32:20.029
Zaki Medina: Side of it.

885
02:32:20.030 --> 02:32:36.789
Zaki Medina: My compliance side, which I hate wearing, is quite interesting. Some of you may work in regulated industries, but this compliance, this quality manager, right, is really going across different Jira projects, Confluence spaces, is creating

886
02:32:36.790 --> 02:32:45.640
Zaki Medina: His goal is… is not daily. Every week, he has to go in and take one or two of the controls, and then, you know.

887
02:32:45.680 --> 02:32:54.480
Zaki Medina: And solve them. And then, obviously, I haven't gone to a monthly and quarterly, as I've only been running this for about a week now.

888
02:32:54.670 --> 02:33:12.350
Zaki Medina: And so on. So, I think the Chrome job thing is really critical as an engineer, my first 3 Chrome jobs all failed, and it took me about an hour to

troubleshoot all of these, and so, I really learned the difference between isolated versus main sessions.

889
02:33:12.350 --> 02:33:23.749
Zaki Medina: So, use isolated sessions for 95% of your jobs, that way they start clean, do the work, deliver the result, no conversation pollution, and they peace out, right?

890
02:33:23.750 --> 02:33:34.159
Zaki Medina: Main only for your main current conversation context, right? When you need context, put it in the workspace file instead of putting in the main. Believe it or not, this

891
02:33:34.160 --> 02:33:42.159
Zaki Medina: kind of housekeeping. Probably in the next 6 months, you'll start seeing… some of you are using Anthropic, and you'll see, Anthropic

892
02:33:42.960 --> 02:34:05.289
Zaki Medina: especially if you're running Cloud Code, or maybe you're running Cloud Code work, you'll see it's compacting, right? It's doing all of that wonderful thing. That's what it's doing. It's actually compacting some of that conversation, removing it from the context window, putting it into a workspace file that's saved for you. You often don't see that workspace file. If you're using Anthropic and Claude. They want to make it really simple and easy.

893
02:34:05.290 --> 02:34:24.150
Zaki Medina: You know, for all the people, you know, professionals that are working there, but for techies like us, we want to see everything, right? So anyway, so typically, what I would ask people to do now is I'm actually deploying this model for every team member, because I want everybody to have a second brain, and to be supercharged, and…

894
02:34:24.150 --> 02:34:37.429
Zaki Medina: You know, so we can continue our high scores in Mario Kart at lunchtime. So anyway, so we're listing out our top 5 recurring tasks, right? I always say, as you build, go through this journey.

895
02:34:37.430 --> 02:34:54.630
Zaki Medina: How do you implement this? Obviously, there's plenty of articles around OpenClaw. You can Google it, you can go to OpenClaw, you'll see a lot of people. You can build your own. Don't fall into the bullshit of buying Mac Minis and MacBooks. You can do that, but you can just spend 20 bucks

896
02:34:54.660 --> 02:34:58.970
Zaki Medina: on Hostinger or OVH Cloud to get your, you know.

897
02:34:59.660 --> 02:35:10.580
Zaki Medina: similar kind of thing. You do need to know how to copy-paste, so if you're a good copy-paste monkey like I am, you'll be fine, you know? So, like my doctor friend, you'll be fine. So, all good.

898
02:35:11.070 --> 02:35:21.070
Zaki Medina: And then, the things that I learned, schedule, never have a Chrome job run at the same time, have them run differently, and so on.

899
02:35:21.140 --> 02:35:25.450
Zaki Medina: Phase 4 is shared state. This is where it gets…

900
02:35:25.520 --> 02:35:44.969
Zaki Medina: interesting, and this is where most open claw guides stop. Agents need to coordinate without human mediation, and so I built a few components, which I'll talk about. I built a Kanban board, to work with my agents, right? So every time a change happens, it's auto-logged.

901
02:35:44.970 --> 02:36:01.160
Zaki Medina: who changed it, when they changed it, why they changed it. The agents identified themselves with an underscore agent field, so I have an audit trail of what's going on, and who basically screwed up what, you know, down the road. Then I have, second, a task router.

902
02:36:01.160 --> 02:36:07.929
Zaki Medina: a task router, I have a separate file called… it's not here, but I have a separate one called…

903
02:36:11.290 --> 02:36:16.400
Zaki Medina: Router and the… routing… sorry, router is someone else.

904
02:36:16.950 --> 02:36:22.670
Zaki Medina: Putting MD. So, basically, what this does is it decomposes. It first classifies

905
02:36:22.670 --> 02:36:38.259
Zaki Medina: Which domain? Is it engineering, compliance, marketing, or product, or sales, like revenue? And then, next, it decomposes it. It breaks down into complex requests, into atomic tasks. Then it creates Kanban-style tasks.

906
02:36:38.260 --> 02:36:42.570
Zaki Medina: with assignee priority and context, then it creates a spawn agent

907

02:36:42.570 --> 02:36:58.629
Zaki Medina: that, you know, and then that spawn agent will have the Kanban task ID, and so on, and then, man, you could probably build a product out of this, and then track and synthesize, and it will collect the results and validate and deliver. So, basically, my spawn prompt is very simple. It's like, hey.

908
02:36:58.710 --> 02:37:07.609
Zaki Medina: you know, implement a biometric auth fallback, as an example, then the Kanban task will come, say, task number.

909
02:37:07.620 --> 02:37:26.619
Zaki Medina: 111113567, blah blah blah, update the task via API, it'll give 3 to 5 instructions. Hey, set the status, add the comments, set… do this, do that, all of that. Now, this was kind of something that I did. My third thing that I did was

910
02:37:26.640 --> 02:37:38.339
Zaki Medina: I have a working model of shared memory between agents. Memory is a big thing, there's lots of billion-dollar startups out there trying to solve memory. So, let me explain what the memory problem is.

911
02:37:38.340 --> 02:37:48.210
Zaki Medina: When every time a spawned agent wakes up, it has zero context about what the other agents did. So if Agent A finished, something, Agent B

912
02:37:48.380 --> 02:38:06.720
Zaki Medina: start something that's related to it. Agent B doesn't know what Agent A did or found, right? So, I created a new folder called Agent Handoffs, and I'm trying to… this is still ongoing. After completing works, agents basically write structured handoff notes.

913
02:38:06.760 --> 02:38:30.379
Zaki Medina: Basically, like in high school or in school, you know? Like, did you finish the work? Did you do your book report? You submit it, right? So it's basically a handoff report that goes in, and the handoff report is what the agent did, key findings, next steps, and then… and then it hands it off, and it positions it in a place where all the agents can view it and use it if they need to, right? And they want to see.

914
02:38:30.390 --> 02:38:39.219
Zaki Medina: Phase 5 is where I've started building quality gates, smart notifications, validation checklists,

915
02:38:39.220 --> 02:38:50.500
Zaki Medina: code, you know, can the code run without errors? So I started using Greptile, and a bit of, OpenAI has something called Artvark to do all my code reviews, so yes.

916
02:38:50.500 --> 02:38:58.359
Zaki Medina: Agents are reviewing other agents' code, and it's going pretty well. It's not so bad. I'm actually not gonna lie, we're in week 3.

917
02:38:58.620 --> 02:39:04.679
Zaki Medina: I have not looked at… I've looked at some code, but not a lot of code, versus January, so…

918
02:39:04.860 --> 02:39:19.199
Zaki Medina: Anyway, so check out Reptile, check out Artwork. I'm nowhere associated with these guys, but, they're, they're doing, God's work, basically. So, anyway, so going into, okay, so what did I cover?

919
02:39:19.200 --> 02:39:21.530
David Gijsbers: Alright, so I think I have 5 minutes left.

920
02:39:21.740 --> 02:39:25.289
Zaki Medina: I want to take the last 5 for,

921
02:39:25.420 --> 02:39:36.220
Zaki Medina: questions. I can give you my honest assessment before we go into questions. Stuff breaks. Kanban server occasionally crashes, I have to restart it.

922
02:39:36.300 --> 02:39:56.159
Zaki Medina: cross-agent coordination is still mostly mediated by me. The handoff system is working, but it's new, it's unproven. I'm running it on a 16GB VPS. No redundancy, no failover, so, you know, we'll see how that goes. When something does break, I end up spending an hour to two hours.

923
02:39:56.220 --> 02:40:10.010
Zaki Medina: troubleshooting, and I spent 2 hours last night troubleshooting a typo, FYI. So, agents, this system is not autonomous, it's augmented. I'm still the decision maker, I'm still the quality gate of last resort.

924
02:40:10.090 --> 02:40:13.430
Zaki Medina: Shit fails at 3 AM, it just stops working.

925
02:40:13.590 --> 02:40:25.540
Zaki Medina: So, you know, we're still in the early days. Agents make me faster, not unnecessary, I still feel. And prompt injection still remains a real concern.

926
02:40:25.540 --> 02:40:36.350

Zaki Medina: Lots of researchers have come in and tried to demonstrate that. That is still something no one has solved. I try to vet every skill manually, but this was my kind of quick reference guide, so…

927
02:40:36.390 --> 02:40:39.370
Zaki Medina: I'll take questions now, for 5 minutes.

928
02:40:40.030 --> 02:40:55.639
Lara Hill - SoftEd: Zachy, this has been great. Thank you so much for jumping in last minute. This has been fantastic. I just want to read out a couple of questions I've seen in the chat, and I think multiple people had the same question, which is, do you run OpenClaw on a standalone machine?

929
02:40:56.420 --> 02:41:10.780
Zaki Medina: No, I don't. I actually don't run it on a standalone machine. I run it on a very… I started with a 16GB VPS, so you can look up Hostinger, you can look at OVH Cloud, I… they have really good

930
02:41:10.780 --> 02:41:17.770
Zaki Medina: Hostinger is my favorite. They have a one-click open-cloud deployment, literally. You deploy it. I just don't like how…

931
02:41:17.770 --> 02:41:31.949
Zaki Medina: they… they lull you in for one, you know, lull you in with a free subscription, and then they give you, like, no, you need to buy one year, two years. Like, dude, in two years, we don't know what's gonna happen. Aliens could show up, I don't know, like, it could be…

932
02:41:31.950 --> 02:41:38.800
Zaki Medina: Craziness. So, that's the only thing I didn't like about these companies. So, I use VPS.

933
02:41:38.860 --> 02:41:39.430
David Gijsbers: I do.

934
02:41:39.430 --> 02:41:55.249
Zaki Medina: I do have… when I did start off, I took my daughter's MacBook Air, and I did deploy it all on MacBook Air, and I ran it in clamshell mode, and it was good, but it was unnecessary, you know what I mean? So…

935
02:41:56.860 --> 02:42:00.310
Lara Hill - SoftEd: Okay, great. Thank you so much. Does anyone else have any questions?

936

02:42:03.640 --> 02:42:11.220
Lara Hill - SoftEd: Zachy, I would love if you could hang out in the chat for a few minutes, and anyone can private message you, and

937
02:42:12.150 --> 02:42:14.940
Lara Hill - SoftEd: David G, did you see any other questions come in?

938
02:42:16.750 --> 02:42:20.930
David Gijsbers: I… I think he covered the one about the… what do you…

939
02:42:21.540 --> 02:42:27.360
David Gijsbers: let… what actions do you allow agents to take on your behalf without human in the loop? Sorry.

940
02:42:27.530 --> 02:42:29.749
David Gijsbers: I think that was… I think that got covered.

941
02:42:30.540 --> 02:42:32.520
Zaki Medina: Yeah, yeah, yeah.

942
02:42:36.590 --> 02:42:49.500
Zaki Medina: Awesome. So guys, as you guys begin your journey, feel free to follow me on LinkedIn. If you have any questions, drop me a line. And Lara knows I actually, also teach some courses.

943
02:42:49.500 --> 02:43:02.789
Zaki Medina: For, SoftEd, you know, that's my therapy, teaching is my therapy here. I love to give back to the community as much as I can. So we have a great… a bunch of great courses on GCP, on Azure.

944
02:43:02.790 --> 02:43:10.979
Zaki Medina: As well as some security course on Agentix, not including prompt injection. Everything else we can definitely mitigate.

945
02:43:13.850 --> 02:43:14.970
Lara Hill - SoftEd: Thanks, Zachy.

946
02:43:16.970 --> 02:43:24.219
David Gijsbers: Thanks, Laura. Laura, we… going to David Vidra now, or… Yeah, okay. Yes. Awesome. So,

947
02:43:24.680 --> 02:43:36.030

David Gijsbers: you know, I think as we were structuring this presentation, we started at the strategy level, we went into the engineering management level, and we got down into the

948
02:43:36.270 --> 02:43:40.739
David Gijsbers: You know, the developer, and individual contributors

949
02:43:40.860 --> 02:43:48.500
David Gijsbers: Also, the reason why I wanted to have David Vidor speak is that he has the perspective of

950
02:43:48.620 --> 02:43:54.139
David Gijsbers: working in a legacy organization. So, prescriber point, you know, 15-person

951
02:43:54.330 --> 02:44:00.699
David Gijsbers: organization, you know, that's a great story, but, you know, does the use of AI scale

952
02:44:00.830 --> 02:44:06.139
David Gijsbers: And so, David, I'm so excited to have you, be part of this

953
02:44:06.290 --> 02:44:09.279
David Gijsbers: conference, so why don't I hand over the microphone?

954
02:44:09.600 --> 02:44:15.509
David Gijsbers: And the share screen to you, and you can start covering, Your… your presentation.

955
02:44:20.440 --> 02:44:21.969
David Vydra: Okay, can you guys hear me?

956
02:44:22.620 --> 02:44:23.530
David Gijsbers: Sound great.

957
02:44:24.030 --> 02:44:27.619
David Vydra: Alright, I am gonna share my PowerPoint.

958
02:44:29.780 --> 02:44:39.120
David Vydra: And, let's see… aren't there… Start the presentation.

959
02:44:39.120 --> 02:44:39.860

David Gijsbers: this one.

960
02:44:41.550 --> 02:44:44.449
David Vydra: Continue to share with PowerPoint, yes.

961
02:44:51.130 --> 02:44:53.210
David Vydra: No, what is this?

962
02:44:55.220 --> 02:44:57.520
David Vydra: Maybe, maybe I did the wrong one.

963
02:44:59.350 --> 02:45:01.290
David Vydra: Okay, is this good?

964
02:45:03.900 --> 02:45:04.650
David Gijsbers: Yes.

Enterprise Spec-driven Development

965
02:45:04.890 --> 02:45:16.870
David Vydra: Okay, perfect. Okay, so I'm David Wiedra. I think of myself as builder of apps and tools, and I work at SAP.

966
02:45:17.170 --> 02:45:42.119
David Vydra: So this presentation is going to be, basically a… sort of a walk down the memory lane of what I've been doing for the last, probably, 27 years, and as you'll see from my slides, I am purely a backend engineer, you know, I write compilers for, you know, as a hobby, so definitely not as polished as some of the other presenters, but hopefully you'll enjoy the story, and then I'll go fairly quickly

967
02:45:42.120 --> 02:45:49.570
David Vydra: and I really enjoy having an interactive discussion, so we'll leave the questions, you know, for the end. So my story starts about

968
02:45:49.900 --> 02:45:52.949
David Vydra: I think, like, 1998, I… I…

969
02:45:53.090 --> 02:46:03.999
David Vydra: got ahold of this book called, Extreme Programming, and it just completely blew my mind, and I… I became, basically red-pilled, XP, test-driven,

970
02:46:04.000 --> 02:46:22.699

David Vydra: you know, you know, all the way. And then, through the years, I progressed from what I would call classical test-driven to behavior-driven, and now to, spec-driven development. So let's take a look a little bit at the history. So, in classical test-driven, the idea was that,

971
02:46:22.710 --> 02:46:46.669
David Vydra: you practice the discipline of writing a test, make sure it fails, then writing some code, and then you would refactor, you would practice evolutionary design, and that way you would keep your system maintainable. And so this was a very, very tight loop, right? Like, people bragged about, you know, my tests run in, you know, 3 seconds, or 10 seconds, or whatever, you know, very, you know, very, very fast, and so you were…

972
02:46:46.670 --> 02:46:51.500
David Vydra: You were very much in the flow, when you were practicing, you know, test-driven development.

973
02:46:51.810 --> 02:47:03.140
David Vydra: However, as time went on, people realized that, people needed more guidance, as to what is a good test and what is not such a good test, and

974
02:47:03.530 --> 02:47:11.470
David Vydra: the idea of behavior-driven development emerged over the years, and the way I look at it is that there was an outer loop.

975
02:47:11.710 --> 02:47:23.220
David Vydra: Where you practiced… you basically specified the system together with your partners, and then as a developer, you went back to your classical test-driven development. All right.

976
02:47:23.720 --> 02:47:37.319
David Vydra: So, let's dive into behavior-driven. So, from my perspective, the key to behavior-driven development is the collaboration of the triad, the product manager, the developer, and the tester. And,

977
02:47:37.490 --> 02:47:49.579
David Vydra: And, you know, depending on how well this goes, I think that really impacts the quality of the system. Essentially, if you're doing this well, you're building the right thing. You may not be building it right, but you're building the right thing.

978
02:47:50.290 --> 02:48:09.079
David Vydra: Okay, now, how to do behavior-driven development? Over the years, we have shifted from, approaches like Cucumber and putting, you know, our specs into GitHub, to using a wiki, and

979
02:48:09.220 --> 02:48:31.430
David Vydra: The reason for that was that I've never seen product managers actually go and modify, you know, a Gorgon file in GitHub. So I wanted, you know, we wanted to find a place that is equally welcome to all of the three amigos, and so we started practicing, you know, the discovery part of behavior-driven development inside of a wiki.

980
02:48:31.430 --> 02:48:36.250
David Vydra: On the other hand, we also realized that

981
02:48:36.330 --> 02:48:54.809
David Vydra: we probably don't need Cucumber. You know, it's an extra tool, and PMs don't really, you know, they're not gonna modify those files directly, and developers, you know, probably don't need an extra tool, they can just write a test. So that's kind of where we ended up over a few years of experimentation.

982
02:48:55.880 --> 02:49:00.169
David Vydra: And, so this is an example of, you know, how we do.

983
02:49:00.610 --> 02:49:20.370
David Vydra: BDD in the Wiki, and the nice thing about Wiki is it has all kinds of widgets, you can put screenshots, you can put links to Figma, you can basically, you know, use various tables, so it's a very nice environment to do, you know, to basically practice the, especially the discovery side of the BDD.

984
02:49:20.970 --> 02:49:30.320
David Vydra: And then, on the test side, what we started doing, because we have requirements of traceability, is just basically linking

985
02:49:30.530 --> 02:49:48.599
David Vydra: in our case, we use Java, the Java code, directly into the wiki page, and so that establishes a certain amount of traceability. We also had some more complex annotations that can work with our internal tooling, but this is, you know, this is the general idea, is that, you definitely want the tests to be traceable, to

986
02:49:48.890 --> 02:49:53.329
David Vydra: The, you know, the source of the intent, let's put it this way.

987
02:49:54.650 --> 02:50:01.320
David Vydra: Okay, so, one of the things that we found over the years is that

988
02:50:02.060 --> 02:50:26.419
David Vydra: to a large degree, our success depended on the quality of our tests, and the tests that really made a difference between, you know, can we deploy daily, and

oh my god, we're scared to deploy, right? We're scared to do continuous delivery, is basically whether we had a well-written component or integration tests that are traceable to the, to the specifications. This was the discovery that

989
02:50:26.420 --> 02:50:32.819
David Vydra: made us realize that that's a very key practice, and I think some of the earliest speakers also covered this.

990
02:50:34.760 --> 02:50:39.300
David Vydra: Okay, so, moving a few years, and

991
02:50:39.870 --> 02:50:57.699
David Vydra: we got this tool called Copilot handed to us, and basically management said, hey, go, you know, go use this thing and see how you like it. And, frankly, I hated it, and what I really hated about it was that it was hallucinating way too much, and it was just

992
02:50:57.700 --> 02:51:00.000
David Vydra: Causing me all kinds of mental strain.

993
02:51:00.080 --> 02:51:01.730
David Vydra: So,

994
02:51:02.750 --> 02:51:17.160
David Vydra: I wrote this manifesto, if you will, on… I own the testdriven.com website, so I wrote this little, you know, blog post saying, I think this may be the second coming of Test Driven. I think if I…

995
02:51:17.720 --> 02:51:23.510
David Vydra: Just, build my own agent that, only shows me code.

996
02:51:23.750 --> 02:51:40.830
David Vydra: when the tests pass, then that will eliminate the hallucinations, and I'll get my mental health back. And, in fact, it actually worked pretty well. So, with a colleague, we wrote an agent, this was before Cloud Code, and

997
02:51:40.970 --> 02:51:46.640
David Vydra: It worked really well. It eliminated most of the hallucinations,

998
02:51:47.610 --> 02:51:53.270
David Vydra: And, unfortunately, it didn't get a lot of,

999
02:51:53.890 --> 02:52:08.800

David Vydra: uptake within the company, because the reality is, not that many people actually practice, sort of, the, you know, what I call the classic TTD, right? But it wasn't a complete waste. We learned two important things in the process. We learned that

1000
02:52:08.870 --> 02:52:19.769
David Vydra: just giving model the tasks was not enough. You actually needed to give it specs in English. That really made a big difference in the kind of code it generated.

1001
02:52:20.260 --> 02:52:22.780
David Vydra: And the second thing that we learned is that

1002
02:52:23.440 --> 02:52:36.740
David Vydra: even though we did all kinds of tricks to optimize it, it was still too slow for the classic TDD experience, right? Like, you had to wait for it to do its thing, and you basically lost focus, so that just wasn't…

1003
02:52:36.740 --> 02:52:43.950
David Vydra: you know, automating the classic TDD at that time just… just wasn't an option, and I still… I still don't, you know, see it.

1004
02:52:44.020 --> 02:52:50.770
David Vydra: So, around that time, I've learned about this thing called spec-driven development.

1005
02:52:50.870 --> 02:53:02.300
David Vydra: And, there are a lot of opinions of what is spec-driven development, so I'm just gonna share my opinion, since this is my presentation. So, the way I look at it is that

1006
02:53:02.300 --> 02:53:11.629
David Vydra: you still have the same outer loop, right? So we still… we still, practice discovery, and we practice specification as a triad, as three amigos.

1007
02:53:11.820 --> 02:53:28.489
David Vydra: But in the inner loop, you work in slices, so you no longer work, like, a single test, you know, fail, pass sort of loop, but you work… you work on some kind of a chunk, some kind of a slice. But otherwise, at least for me, a lot of the same dynamics are very similar.

1008
02:53:30.030 --> 02:53:38.359
David Vydra: So why work in slices? So, as I've said, I think the reality is that, very few developers, like, really,

1009
02:53:38.780 --> 02:53:42.890
David Vydra: Even practiced, you know, test driven one step at a time.

1010
02:53:43.170 --> 02:53:55.559
David Vydra: And the reality is that right now, LLMs, the good ones, they do take some time to go through their paces and make sure they think and don't hallucinate as much.

1011
02:53:55.890 --> 02:54:00.260
David Vydra: I also have a theory that some planning is actually good, and

1012
02:54:00.770 --> 02:54:15.419
David Vydra: as I remember back, I've had the privilege to work with some very, very good engineers in the pair programming setup, and the ones I remember actually always, you know, wrote down basically some

1013
02:54:15.440 --> 02:54:22.240
David Vydra: some task lists and some pseudocode, before we got into sort of a, you know, proper…

1014
02:54:22.510 --> 02:54:32.490
David Vydra: pair programming using a classic test-driven approach. So I think a little bit of that planning was always there for people who were very experienced and very good.

1015
02:54:34.300 --> 02:54:37.489
David Vydra: So, what is a slice?

1016
02:54:38.200 --> 02:54:48.679
David Vydra: Probably a classical, definition of a slice is you just kind of… a thin slice of your interface that you go from user through business logic to day access.

1017
02:54:48.860 --> 02:54:53.699
David Vydra: In my experience, what I'm calling a slice is

1018
02:54:53.920 --> 02:55:03.710
David Vydra: you know, pretty much anything that's the right chunk for me to work on. So it can be a full feature, although I rarely do that. Most of the time, I work sort of one scenario at a time.

1019
02:55:03.820 --> 02:55:10.850
David Vydra: And obviously a bug or a defect, or a refactoring step. So I basically, you know, treat all of those as…

1020
02:55:11.080 --> 02:55:15.010
David Vydra: slices, and you'll see, why… why I do that.

1021
02:55:15.910 --> 02:55:29.069
David Vydra: And, the tool that I've settled on, after looking at a few tools is called OpenSpec. It's open source, and I really, I really liked, you know, its mission, and I like that it was built

1022
02:55:29.070 --> 02:55:40.089
David Vydra: From the ground up for the… for, you know, for the brownfield development environment, and it's very iterative, and it's very, very, very, you know, lightweight.

1023
02:55:40.550 --> 02:55:47.119
David Vydra: So, basically, using… when I'm using OpenSpec,

1024
02:55:47.370 --> 02:55:55.570
David Vydra: There are 4 files that, you know, guide, guide the agent. So, initially, there's the proposal.

1025
02:55:56.080 --> 02:56:09.170
David Vydra: And then, the next step is you… again, this is all with the help of the agent, by the way, I'll show you. You develop the spec file. After that, you go into the design and the tasks. So…

1026
02:56:10.140 --> 02:56:32.910
David Vydra: OpenSpec comes with, a few, agent commands that you load up, and then a few skills, and then it establishes the cycle where you start a new, they call it change, I call it a slice, and then you continue through those four files, so you do this maybe a couple times, and then you apply, you generate the code, and then you archive, which basically means you…

1027
02:56:33.000 --> 02:56:42.179
David Vydra: Merge the changes into the long-lived specs, and then you create the archive of that particular slice or session.

1028
02:56:43.980 --> 02:56:50.309
David Vydra: So, let's go through how this actually works in practice. So, initially, I would…

1029
02:56:50.660 --> 02:56:59.029
David Vydra: start a new slice, and I would just give it a very short, typically a very short, command, like, you know, do this for me.

1030
02:56:59.030 --> 02:57:23.269
David Vydra: And it will go and it will analyze, you know, the cloud code in this case will analyze my code, and come up with a proposal. And then I spend a little bit of time looking at the proposal, and if it makes sense, I modify it somewhat. If it's completely off-base, I may just kill the session and start fresh. So it depends on the situation. But in my situ… in the project I've been working for the last month, it's been

1031
02:57:23.490 --> 02:57:25.079
David Vydra: Amazingly good at this.

1032
02:57:25.610 --> 02:57:30.609
David Vydra: And so after the proposal, it will generate, a spec file.

1033
02:57:31.110 --> 02:57:45.000
David Vydra: And in this case, it uses sort of the, the given-when, when then, syntax, sort of borrowed from, from Gherkin, not exactly Gherkin, but sort of in that, in that style.

1034
02:57:45.670 --> 02:58:02.209
David Vydra: And then, it will create a design document, and this is the one I pay a lot of attention to, because I want to make sure that, you know, it's not going off the guardrails and doing something, like, either it misunderstood, or it's using, you know, the wrong design pattern that I don't agree with, etc.

1035
02:58:02.210 --> 02:58:12.649
David Vydra: So that's that document, and then, obviously, there's the task list. In this case, you know, it starts without all these X's, but in this case, I took a screenshot after it actually finished it.

1036
02:58:12.910 --> 02:58:17.640
David Vydra: Now, this whole process typically takes me, I would say.

1037
02:58:19.180 --> 02:58:33.390
David Vydra: under 10 minutes, right? Depending on the size of the slice, but typically I work in small slices. And after I do that, and often it's just 3 or 4 minutes, I'm ready to generate my code, I generate my code, and then I go through

1038
02:58:33.750 --> 02:58:36.740
David Vydra: The process of archiving,

1039
02:58:37.000 --> 02:58:55.150

David Vydra: well, of course, before I archive it, I will thoroughly test my code, right? I'll make sure that my tests look good, and typically, these days, I fire up the debugger, and I go through, you know, through my tests, at least with a debugger.

1040
02:58:55.150 --> 02:58:57.200
David Vydra: Because,

1041
02:58:57.200 --> 02:59:12.840
David Vydra: you know, we… like, the project I work on, as David mentioned, Legacy Project, we process over $6 trillion in transactions a year, so, you know, as we say, it's not just lives, it's serious money we're talking about.

1042
02:59:12.840 --> 02:59:26.129
David Vydra: So yeah, I do look at… I do look at the code, but I look a lot more at the tests these days than at the… at the actual code. And then you go through a process of archiving, so I have a nice… I have a nice,

1043
02:59:26.170 --> 02:59:31.199
David Vydra: you know, record of each slice. And by the way, by the way, these are…

1044
02:59:31.330 --> 02:59:36.839
David Vydra: These are slices from OpenSpec, using OpenSpec to build itself.

1045
02:59:37.550 --> 02:59:55.780
David Vydra: Right? And then the specs that are, you know, long-lived specs, are updated, and again, they're also stored in Git. So this completes, you know, one cycle, and again, on average, I've been spending maybe,

1046
02:59:56.820 --> 03:00:09.510
David Vydra: 45 minutes to an hour, on each, on each, you know, each commit, each PR, because, it's, it's been, it's been quite good. Since December, since we got the, the new,

1047
03:00:09.910 --> 03:00:15.489
David Vydra: Opus models, on my little project that I work on right now, it's quite good.

1048
03:00:16.130 --> 03:00:23.020
David Vydra: Alright, so, this has been mentioned before by previous speakers, but…

1049
03:00:24.350 --> 03:00:31.540
David Vydra: Before you get into any kind of agentic development, you should take a very careful look at your organization.

1050
03:00:31.540 --> 03:00:48.470
David Vydra: and see if you're ready for it, right? Because it is powerful, and it's either gonna take you to a very good place, or it's gonna take you to a very bad place. And if you need some help, if you need some upskilling, this is the right… this is the time, this is the year to really get going on that, and…

1051
03:00:48.470 --> 03:00:55.370
David Vydra: you know, I'm glad SelfTech, you know, provides, so many great courses, but, absolutely,

1052
03:00:55.370 --> 03:01:09.459
David Vydra: this is required to start getting benefits from this technology. Make sure that you're in a good place, and that you have… or you have a plan to, you know, to get in a better place. So specifically, I look at…

1053
03:01:09.780 --> 03:01:12.800
David Vydra: two things. I look at, you know.

1054
03:01:13.420 --> 03:01:27.329
David Vydra: domain-driven design is a very strong, important discipline in the kinds of systems that are built, but specifically module design, right? You need… you need to be able to build modular systems, because you need to be able to…

1055
03:01:27.510 --> 03:01:39.289
David Vydra: A fit them into the context, you need to be able to test them in isolation well, etc, and if you're not modular, as Lara was showing, you're gonna get a lot of spaghetti code, and

1056
03:01:39.290 --> 03:01:56.910
David Vydra: you're gonna reach a point of no return, maybe in a matter of months, not years, as typically what happened before. And the other ingredient is continuous delivery. If you're afraid to… if you're building enterprise systems, which is what this talk is about, and you're afraid to deploy.

1057
03:01:57.280 --> 03:02:10.730
David Vydra: you need to ask yourself why. What is it about your testing strategy that makes you… because, you know, software doesn't age like wine, it ages like milk, right? So if your software is sitting there undeployed.

1058
03:02:10.910 --> 03:02:29.309
David Vydra: the big question is why, and most likely, you have to fix your testing approach. And if you have those two, then welcome to spec-driven development. I've really, really enjoyed, especially the last month of my life since I've gotten OpenSpec and got the new models. It's been just…

1059
03:02:29.440 --> 03:02:30.760
David Vydra: Absolute blast.

1060
03:02:31.360 --> 03:02:34.990
David Vydra: And another reminder is…

1061
03:02:34.990 --> 03:02:52.329
David Vydra: you still need to practice evolutionary design, you still need to refactor your systems, because I don't see the agents, doing this work for you, at least not this month, right? Maybe in two months, they'll get there, and then, you know.

1062
03:02:52.420 --> 03:03:05.010
David Vydra: who knows, right? I don't know what's gonna happen in two months, but right now, I still… I still refactor aggressively to make sure that, you know, I can put my name on my code, you know, with pride, and that I, you know, I don't,

1063
03:03:05.490 --> 03:03:08.779
David Vydra: diminish the maintainability of the systems that I work on.

1064
03:03:09.610 --> 03:03:14.800
David Vydra: And then, lastly… oh, no, sorry. One more… one more thing. So…

1065
03:03:15.110 --> 03:03:19.599
David Vydra: So is it, like, are we at 10X, right? Everybody keeps talking about, you know, 10X?

1066
03:03:19.600 --> 03:03:36.739
David Vydra: And, I… it's… developer productivity is very difficult to measure, but I would say, personally, I definitely feel the most excited that I've been about developing software, probably since, again, I got hold of that Extreme Programming book.

1067
03:03:36.740 --> 03:03:54.229
David Vydra: just amazing times, but, the way that I choose to spend my time, because, you know, it writes most of the code for me now, but I spend more time on design, and I spend a lot more time on testing, right? So it was never about the code, right? Software engineering was…

1068
03:03:54.320 --> 03:04:06.970
David Vydra: much more than writing code. That was always true, but now we have fewer and fewer excuses not to become, you know, great software engineers, because the tools are giving us the time back that we can focus on

1069
03:04:07.360 --> 03:04:10.690
David Vydra: Both on design and on good testing practices.

1070
03:04:11.310 --> 03:04:20.809
David Vydra: And people ask me often, you know, how do I prepare for the future with agents? And, in my experience,

1071
03:04:21.180 --> 03:04:26.690
David Vydra: Going back to the fundamentals and reading some classic books is a good idea, because…

1072
03:04:26.720 --> 03:04:44.459
David Vydra: the models have read all these books, right? And you can talk to them at a much higher level of abstraction, right? I can talk to the model, I can say, you know, I want you to use strategy pattern here, or I want you to replace conditional with polymorphism.

1073
03:04:44.460 --> 03:04:51.990
David Vydra: et cetera, et cetera, right? So, I think this is the key. I think the key is to really focus on

1074
03:04:52.110 --> 03:04:55.510
David Vydra: You know, higher level software engineering skills, and…

1075
03:04:55.900 --> 03:05:05.720
David Vydra: Anyway, I think, I think we're gonna be okay, but at least right now, it's a huge amount of fun, I'm really enjoying the entire process.

1076
03:05:06.270 --> 03:05:12.509
David Vydra: So, that's it for my presentation, and I would love to,

1077
03:05:13.550 --> 03:05:20.410
David Vydra: just take questions and, you know, have a quick discussion on this topic. Happy to…

1078
03:05:20.720 --> 03:05:24.050
David Vydra: To see if anybody has any specific questions, or…

1079
03:05:25.130 --> 03:05:27.829
David Vydra: Just, I'm gonna stop sharing this.

1080
03:05:28.140 --> 03:05:28.860

David Vydra: Yep.

1081
03:05:38.620 --> 03:05:40.480
David Gijsbers: And you look like you have a question.

1082
03:05:48.810 --> 03:05:49.540
David Mantica: No?

1083
03:05:50.650 --> 03:05:54.539
David Mantica: Tom has a question. David, what do you think? Mr. Wessel has a question.

1084
03:05:55.820 --> 03:06:05.640
Tom Wessel: Oh, thanks for putting me on the spot. I do not… I was just curious, David, if you have maybe some examples of what you've actually done that maybe, hopefully, are…

1085
03:06:05.810 --> 03:06:10.510
Tom Wessel: cleanse, so there's no client-specific information. Is that something you could share offline?

1086
03:06:11.440 --> 03:06:17.979
David Vydra: I mean, I tried to put what I can into the slides. Yeah, unfortunately, I don't have permission to show my actual work.

1087
03:06:18.470 --> 03:06:26.170
David Vydra: But basically, I would say that, in the last month,

1088
03:06:26.780 --> 03:06:34.510
David Vydra: I wrote, I think, about 25 slices, and it's been, like, 95%

1089
03:06:34.640 --> 03:06:38.540
David Vydra: Code that, the agent generated for me, right?

1090
03:06:41.130 --> 03:06:51.089
David Vydra: And, again, it's just… it's just been an absolute blast. Since the… since the Opus came out, it's just… it's… it's… these are much better models. And,

1091
03:06:51.700 --> 03:07:05.590
David Vydra: Yeah, I mean, that's all I can say, is that if you haven't tried it, you should try it. And again, OpenSpec really helps me to just have the discipline, right, to break things into slices, to really specify them.

1092
03:07:05.590 --> 03:07:13.220
David Vydra: Right? And then also there's, you know, because OpenSpec archives all of that, if I do find some issues, I can go back…

1093
03:07:13.270 --> 03:07:29.060
David Vydra: And, you know, as Lada was showing, we can improve our fundamental documents, we can improve our AgentsMD, or OpenSpec also has, like, a projectMD file, right? So we can, we can improve, you know, improve all of these guide, you know, guide rails,

1094
03:07:29.110 --> 03:07:34.189
David Vydra: And, the other thing that I think is very cool is that,

1095
03:07:35.130 --> 03:07:50.279
David Vydra: it's cheap enough now to run experiments, right? So maybe I have some ideas. I can design it this way, or I can design it that way, right? Well, I can just fire up, you know, two, Git work trees, right? And run, run, you know, run two…

1096
03:07:50.570 --> 03:07:56.949
David Vydra: two slices with different design criteria in parallel, and then be able to choose which one I like better, right?

1097
03:07:57.090 --> 03:08:06.420
David Vydra: So again, to me, it's all about getting time back, and really, really focusing on really good software engineering fundamentals and practices, and .

1098
03:08:07.100 --> 03:08:11.880
David Gijsbers: David, have you been able to scale this… have you been able to scale this across your team?

1099
03:08:12.680 --> 03:08:18.390
David Vydra: So… That is… that is a good question.

1100
03:08:19.250 --> 03:08:24.949
David Vydra: I don't want to get into certain specifics, it's a legacy company, but,

1101
03:08:25.590 --> 03:08:32.679
David Vydra: We have many, many teams, and different teams are at different stages of maturity, right?

1102
03:08:32.880 --> 03:08:38.239
David Gijsbers: Yeah. So, in my case specifically, right now,

1103
03:08:38.880 --> 03:08:41.320
David Vydra: This is mostly,

1104
03:08:41.480 --> 03:08:51.420
David Vydra: a small project… a small project that I'm doing, right? But we definitely have, you know, some other teams I'm in touch with that are in the process of scaling it.

1105
03:08:51.420 --> 03:09:05.540
David Vydra: I would not, you know, I would not be here claiming something that I haven't seen with my own eyes, right? So I think it is very, very early for a large company like us, but we're definitely moving, as fast as we can to…

1106
03:09:05.540 --> 03:09:12.059
David Vydra: You know, to really start taking advantages of these tools and these approaches, and

1107
03:09:12.900 --> 03:09:19.230
David Vydra: Yeah, so that's all I'm gonna say. I think, you know, for a company our size, it's very difficult to…

1108
03:09:19.330 --> 03:09:35.160
David Vydra: say one thing and the other. But I will say that, you know, SAP is investing a lot of money into both the tokens and the tools and the internal practices, so it definitely is one of the…

1109
03:09:35.390 --> 03:09:39.780
David Vydra: Key drivers of our priorities these days, yeah, for sure.

1110
03:09:39.970 --> 03:09:43.370
Carrie Driscoll: I do have a question. I don't know if,

1111
03:09:43.900 --> 03:09:55.940
Carrie Driscoll: if I missed this part, but, like, if you're looking at an Agile team, and you're looking at spec develop… how are we calling it? Specification-driven development? Yeah. With this tool, are you going back to, like.

1112
03:09:56.110 --> 03:10:08.450
Carrie Driscoll: having, BAs and architects write BRDs and, and, BDDs in order, sorry, yeah, large requirement documents in order to then utilize that for these.

1113
03:10:08.560 --> 03:10:09.720
Carrie Driscoll: these tests?

1114
03:10:10.460 --> 03:10:14.089
David Vydra: No, no. I think that.

1115
03:10:14.460 --> 03:10:15.840
Carrie Driscoll: You can do it off stories, and…

1116
03:10:15.840 --> 03:10:19.819
David Vydra: I think we need to… we need to work in small steps, right?

1117
03:10:19.820 --> 03:10:37.900
David Vydra: So, as I put in my slide, the outer loop, in my experience, hasn't changed, right? So we still get together with, you know, product managers, and we basically, you know, I mean, they have a PRD written, of course, right? That's how we work, but then we get together.

1118
03:10:37.900 --> 03:10:48.779
David Vydra: the three Amigas, and we work through the detailed specifications, the edge conditions, you know, all these things, right, that we typically do as a triage, right? From the business point of view, right?

1119
03:10:48.780 --> 03:10:49.390
Carrie Driscoll: Right, yeah.

1120
03:10:49.390 --> 03:11:00.779
David Vydra: And then, again, the differences, from before is where, developers… again, we can… the thing about test-driven is everybody…

1121
03:11:00.780 --> 03:11:18.580
David Vydra: talks about test-driven, very few people actually do it, right? But let's assume… let's assume the positive case, where people actually were practicing test-driven, then there would be, you know, a much tighter loop of sort of developing your system, you know, write a test, write some code, write a test, write some code, right? And then, so to me, the difference is, again.

1122
03:11:19.000 --> 03:11:22.029
David Vydra: I don't work that way anymore, I work in slices.

1123
03:11:22.030 --> 03:11:22.680
Carrie Driscoll: Yep.

1124
03:11:22.680 --> 03:11:34.600

David Vydra: But the spirit is there, right? My slices are pretty small, right? And I still do a lot of testing, and I still do the refactoring. I still do all of that, right? So… so… so to my mind,

1125
03:11:35.370 --> 03:11:43.010
David Vydra: especially for mission-critical systems, this is still the approach that I would recommend. I know that, you know, if you read the…

1126
03:11:43.010 --> 03:11:56.939
David Vydra: you know, the papers, you'll see that some people are running swarms of agents, right, and they're just giving it, like, a very big PRD, and then they go skiing for the weekend, and they come back, and then, you know, the system is built, and…

1127
03:11:57.640 --> 03:11:58.760
David Vydra: Maybe?

1128
03:11:58.760 --> 03:11:59.320
Carrie Driscoll: Amazing.

1129
03:11:59.320 --> 03:12:23.710
David Vydra: I haven't seen that happen, and I would not, you know, I would not, you know, trust this approach with the kind of systems that we work on. But this approach that I'm proposing here is, I think is an evolutionary approach. It's pretty safe, it's easily observable, and therefore, in true agile fashion, you can adjust, right? You can adjust the size of your slices, you can fix your… you can choose your tools, you can choose how you work, right?

1130
03:12:23.920 --> 03:12:31.410
David Vydra: And again, this is basically what I'm doing right now. What I showed you is literally the last month of my life.

1131
03:12:31.410 --> 03:12:33.670
Carrie Driscoll: Very interesting. Thank you.

1132
03:12:33.670 --> 03:12:46.059
David Gijsbers: So, David, one question from the chat. In your view, why is spectra in development gaining traction? Is it connected to the rise of AI slash non-deterministic systems?

1133
03:12:47.110 --> 03:12:51.250
David Vydra: Yeah, so I think that, I think that,

1134
03:12:52.060 --> 03:12:56.580

David Vydra: There's… there's a huge, demand for…

1135
03:12:56.760 --> 03:13:01.079
David Vydra: These tools to write a lot of the detailed code, right?

1136
03:13:01.870 --> 03:13:09.200
David Vydra: I think that, As I've said, it gives us the ability to…

1137
03:13:10.020 --> 03:13:18.269
David Vydra: really focus on outcomes to be more impactful, right? To be more, you know, to have more time to literally speak to our customers and to meet their needs, right?

1138
03:13:18.610 --> 03:13:34.590
David Vydra: And again, in my view, working in slices is an obvious choice because of the nature of these systems right now. They're just not instantaneous, right? So if I wanted to practice classic test-driven and just do one thing at a time, it just…

1139
03:13:34.940 --> 03:13:54.209
David Vydra: I just don't think it will work… I mean, it hasn't worked for me, let's put it this way, right? And I haven't seen it… personally, I haven't seen it in succeed, so… so I think that that is one of the reasons why Spec Driven is… is gaining traction, and other people are calling it by different names, right? I've heard names like Outcome Engineering, I hear names like Igentic Engineering, right?

1140
03:13:54.210 --> 03:13:56.530
David Vydra: But the common theme is,

1141
03:13:56.920 --> 03:14:12.000
David Vydra: you still, you know, you work in slices, you put a lot of guardrails, you really get your testing strategy in a good place, right? Et cetera, et cetera. So all of the themes are common across different names, in terms of what people are calling it, but…

1142
03:14:12.320 --> 03:14:16.920
David Vydra: spec-driven works for me, and so I've adopted it as my next step.

1143
03:14:17.740 --> 03:14:21.349
David Mantica: Awesome. I have one question, Dave. Who is SAP?

1144
03:14:21.350 --> 03:14:21.860
David Gijsbers: Yep.

1145

03:14:22.870 --> 03:14:34.390
David Vydra: Yes, good question. So, yeah, SAP is, I was told we handle about, what, three-quarters of the world's GDP, right? So…

1146
03:14:35.190 --> 03:14:39.220
David Vydra: We build very, very large systems for very important companies.

1147
03:14:39.380 --> 03:14:44.610
David Mantica: I love it, Meg. Thank you so much for being with us, this is fantastic. Go ahead, David G, sorry about that.

1148
03:14:44.610 --> 03:14:54.360
David Gijsbers: I'd make a joke. Alright, thank you very much. So we have a special guest for the next, 10 minutes or so. Danielle, are you on the line?

1149
03:14:54.430 --> 03:14:59.770
Danielle deLuise: Ready to… okay, awesome. Yeah, we'll hand over Danielle from the Scrum Alliance.

1150
03:14:59.770 --> 03:15:05.119
David Gijsbers: She's just gonna share with us a little bit about, the direction that organization's heading.

1151
03:15:05.330 --> 03:15:30.270
Danielle deLuise: Hi, everyone. Danielle DeLuiz, I'm the Chief Product Officer at Scrum Alliance, and we're really happy to be partnering with SoftEd, and I just wanted to talk to you a little bit about just Scrum Alliance today. I know most of you probably know us as the CSM and CSPO home, but we've become far more than just a framework organization. Our mission has evolved to meet many of the exact challenges that's being discussed here today, right? We're helping professionals

1152
03:15:30.270 --> 03:15:54.690
Danielle deLuise: and enterprises thrive in lots of complex environments. And so we're here today because we believe that the success of AI and software engineering is directly tied to the level of agility within the organization. And when we're talking to enterprise leaders, we're hearing so many of the things that David mentioned in that last presentation. They're less focused today on doing projects right, and they're more interested in ensuring along the way that they're doing the right projects.

1153
03:15:54.690 --> 03:16:07.349
Danielle deLuise: And so modern enterprises are demanding professionals who aren't boxed into any narrow role. No news to any of us on this call, I'm sure. They need people who can pivot between strategy and delivery and transformation.

1154

03:16:07.350 --> 03:16:22.150
Danielle deLuise: And at Scrum Alliance, we're building those professional powerhouses through learning journeys and giving them the systems thinking and the leadership skills needed to bridge the gap between, like, a high-level AI strategy and the messy reality that can be the execution.

1155
03:16:22.350 --> 03:16:38.900
Danielle deLuise: I think, there's a common misconception that AI adoption is a technology problem, but I don't think that's true, it's an adaptability problem, right? AI is going to reward organizations that can experiment and learn quickly. So, if your team has a psychological safety to fail.

1156
03:16:38.900 --> 03:16:45.990
Danielle deLuise: If there are the feedback loops in place to refine as you go, and the cross-functional collaboration to connect tech to value.

1157
03:16:46.100 --> 03:16:48.340
Danielle deLuise: That is when you will win with AI.

1158
03:16:48.490 --> 03:17:01.300
Danielle deLuise: And so at Scrum Alliance, we call those agility skills, and they are the underlying capabilities, the prioritization, the transparency, and the iterative problem solving that make professionals plug and play in any context.

1159
03:17:01.300 --> 03:17:09.909
Danielle deLuise: And agility addresses the root cause of slow delivery and misalignment, which can allow AI to actually scale, rather than create chaos.

1160
03:17:10.390 --> 03:17:27.699
Danielle deLuise: And at Scrum Alliance, to support this shift, we've been introducing micro-credentials, which are targeted agility multipliers, so to speak, in areas like AI literacy, change enablement, and value measurement. And they're designed to help professionals move from team facilitators to enterprise enablers.

1161
03:17:27.720 --> 03:17:42.679
Danielle deLuise: Because as, you know, we were talking about in the last session, and I'm sure many sessions to come, AI itself won't transform businesses, but the people working at those businesses will transform it themselves. But only if they have the agility to evolve alongside the technology.

1162
03:17:42.680 --> 03:17:50.180
Danielle deLuise: And so at Scrum Alliance, we are really excited to be new partners with SoftEd, and just thank you for having us today. I'm looking forward to the rest of the sessions.

1163

03:17:51.030 --> 03:17:52.449
David Gijsbers: Thanks so much, Danielle.

1164
03:17:52.970 --> 03:17:56.300
David Gijsbers: Alright, next in queue, we have Ken Pugh.

1165
03:17:56.640 --> 03:18:00.459
David Gijsbers: Kenny, you ready? Oh, look at this… look at this fancy…

1166
03:18:00.660 --> 03:18:08.230
David Gijsbers: screen that he has, he's… he's in… he's… I don't know how he's in presentation mode. He's, like, literally in presentation mode.

1167
03:18:08.230 --> 03:18:12.250
Ken Pugh: That's the beauty of AI.

1168
03:18:12.400 --> 03:18:15.269
David Gijsbers: Ken, tell us about Gherkin for AI testing.

1169
03:18:15.270 --> 03:18:16.690
Ken Pugh: Okay.

Utilizing Gherkin in AI Testing

1170
03:18:16.900 --> 03:18:20.799
Ken Pugh: So that's what I'm going to talk about, utilizing Gherkin for AI testing.

1171
03:18:21.780 --> 03:18:28.369
Ken Pugh: So, why Gurkin for AI? Well, it streamlines the prompting.

1172
03:18:28.810 --> 03:18:33.120
Ken Pugh: And we'll see some of this as I describe things in a little more detail.

1173
03:18:33.470 --> 03:18:38.419
Ken Pugh: And it decreases rework, it's readable.

1174
03:18:38.800 --> 03:18:46.870
Ken Pugh: you can actually read the specs. You saw the give and when then in David's presentation that OpenSpec creates.

1175
03:18:47.460 --> 03:18:51.980
Ken Pugh: And it acts as an executable specification.

1176
03:18:53.720 --> 03:18:54.680
Ken Pugh: Now…

1177
03:18:58.320 --> 03:19:04.450
Ken Pugh: Okay, so I've got one overall rule. There are exceptions to every statement except this one.

1178
03:19:04.720 --> 03:19:17.439
Ken Pugh: And what do I mean by that? When I say you always ought to do something, I don't mean always, always, but usually always. And when I say you should never do something, I don't mean never, ever, but usually never.

1179
03:19:17.700 --> 03:19:27.260
Ken Pugh: So if something I say here isn't going to work for you, feel free to raise your hand with the exception, because we learn by the exception, as well as by the rules.

1180
03:19:28.970 --> 03:19:31.340
Ken Pugh: My, another overall rule.

1181
03:19:31.850 --> 03:19:35.140
Ken Pugh: Context is everything.

1182
03:19:35.610 --> 03:19:46.389
Ken Pugh: How you apply Gherkin, or any process, depends on what your situation is, your organizational thing, and what domain you're operating in.

1183
03:19:46.670 --> 03:20:01.220
Ken Pugh: If you care about… if you just want to get stuff out, and you want to make it look pretty, but don't have a lot of business rules, that's gonna be a different context than if you're a financial organization, where 90% of the programming is business rules.

1184
03:20:04.610 --> 03:20:12.379
Ken Pugh: All of a sudden, my, clicker doesn't want to… there we go. Explicitness beats implicitness.

1185
03:20:13.360 --> 03:20:25.229
Ken Pugh: To be explicit, We'll see, giving very precise, scenarios with lots of data definitions in them.

1186
03:20:25.620 --> 03:20:34.520

Ken Pugh: And… that… As opposed to just a general overall effect. And we'll see an example of that later.

1187
03:20:35.200 --> 03:20:38.370
Ken Pugh: And finally, my one specific rule.

1188
03:20:38.600 --> 03:20:44.340
Ken Pugh: No code goes in until the spec slash test goes on.

1189
03:20:44.510 --> 03:20:50.549
Ken Pugh: If you don't have a spec for what you want to be doing, then any code will get you there.

1190
03:20:52.950 --> 03:20:56.840
Ken Pugh: So, for the equivalence purposes for this session.

1191
03:20:57.140 --> 03:21:08.800
Ken Pugh: Specification-driven development, behavior-driven development, automatic acceptance test-driven development are all the same. They do have slight variations, but

1192
03:21:08.990 --> 03:21:14.510
Ken Pugh: In this case, everything is about… Oops, sorry.

1193
03:21:14.690 --> 03:21:20.969
Ken Pugh: My clicker decides it doesn't want to work today. Specifying behavior and testing behavior

1194
03:21:21.890 --> 03:21:27.109
Ken Pugh: That that behavior that you've specified actually exists in the underlying system.

1195
03:21:27.810 --> 03:21:34.340
Ken Pugh: So, what is a specification? It's the behavior from the exterior point of view of a system.

1196
03:21:34.560 --> 03:21:40.530
Ken Pugh: We have a user that sends an input into the system. Do they get the right output out?

1197
03:21:41.490 --> 03:21:51.569
Ken Pugh: I have a state change. The user is placing an order, he hits the payment button, and now is the order in process.

```
1198
03:21:51.880 --> 03:21:55.830
Ken Pugh: And finally, External interfaces.

1199
03:21:56.090 --> 03:22:00.030
Ken Pugh: I.e, we're talking to servers or service providers.

1200
03:22:01.770 --> 03:22:04.080
Ken Pugh: Do we send them the right information?

1201
03:22:05.650 --> 03:22:08.239
Ken Pugh: So let me start with a couple of definitions.

1202
03:22:09.080 --> 03:22:11.150
Ken Pugh: Acceptance criteria.

1203
03:22:11.580 --> 03:22:12.890
Ken Pugh: General.

1204
03:22:13.910 --> 03:22:17.839
Ken Pugh: Very general. And it outlines a correct behavior.

1205
03:22:18.240 --> 03:22:25.060
Ken Pugh: And acceptance tests specific scenarios that either pass or fail.

1206
03:22:25.760 --> 03:22:30.440
Ken Pugh: You're doing a calculator, acceptance criteria. When I add two numbers
together.

1207
03:22:31.010 --> 03:22:32.919
Ken Pugh: I must get the correct sum.

1208
03:22:33.350 --> 03:22:35.250
Ken Pugh: Acceptance test.

1209
03:22:35.530 --> 03:22:39.460
Ken Pugh: When I add 2 plus 2, I get 4.

1210
03:22:40.420 --> 03:22:43.900
Ken Pugh: then you'd probably go and go, Ken, what else could you get?
```

1211
03:22:44.010 --> 03:22:45.760
Ken Pugh: Well, 1-0.

1212
03:22:46.080 --> 03:22:48.730
Ken Pugh: And you go, 1-0?

1213
03:22:49.030 --> 03:22:52.560
Ken Pugh: How'd you get that? Don't you know we use base 4 here.

1214
03:22:53.120 --> 03:22:56.939
Ken Pugh: Now, if you think something like that would never occur.

1215
03:22:57.450 --> 03:23:03.880
Ken Pugh: A few years back, there was a thing called the Mars Lander, except it crashed on Mars.

1216
03:23:04.570 --> 03:23:12.270
Ken Pugh: And it was designed by U.S. and Europeans. In the post-mortem, and it was literally a post-mortem.

1217
03:23:12.440 --> 03:23:22.549
Ken Pugh: The Europeans asked the Americans, oh, what did you use for the gravitational constant? And the Americans said, well, 3 feet per second per second.

1218
03:23:22.840 --> 03:23:25.530
Ken Pugh: And the Europeans said, eat?

1219
03:23:26.400 --> 03:23:31.520
Ken Pugh: $150 million piled onto the Martian surface.

1220
03:23:32.390 --> 03:23:44.749
Ken Pugh: So, the other thing about acceptance tests is they are implementation independent. Doesn't matter whether your implementation is in Java, or C Sharp, or…

1221
03:23:45.080 --> 03:23:46.570
Ken Pugh: Visual Basic.

1222
03:23:48.110 --> 03:23:51.440
Ken Pugh: So, let's give an example of defining behavior here.

1223
03:23:52.690 --> 03:23:57.700

Ken Pugh: Okay, if we were all together, I'd ask the question, who wants a fast car?

1224
03:23:58.250 --> 03:24:04.720
Ken Pugh: And I… when I do this in my classes, somebody inevitably comes up and gives… and…

1225
03:24:04.850 --> 03:24:10.149
Ken Pugh: comes up with a, an example. So, first the criteria.

1226
03:24:10.560 --> 03:24:15.450
Ken Pugh: We must accelerate from… to the desired speed within some time.

1227
03:24:15.870 --> 03:24:19.619
Ken Pugh: That's our general thing. How about a very specific?

1228
03:24:20.550 --> 03:24:25.870
Ken Pugh: We want to accelerate from 0 to 60 in 5 seconds.

1229
03:24:26.880 --> 03:24:34.420
Ken Pugh: That's a very nice test, very nice spec. But how about the top speed? Is 60 miles per hour our top speed?

1230
03:24:35.340 --> 03:24:42.900
Ken Pugh: How long do we want to be able to be at that top speed? 2 seconds, or should we want to be able to drive at 15 minutes?

1231
03:24:43.100 --> 03:24:48.080
Ken Pugh: So what we do is we can have a requirement, and then

1232
03:24:48.280 --> 03:24:58.850
Ken Pugh: Using one of our DDDs, we take that requirement, and we split it up into multiple tests, and multiple specifications.

1233
03:24:59.620 --> 03:25:08.870
Ken Pugh: So, we have our fast car, and we split it up, things like acceleration, what should be the top speed, and so forth.

1234
03:25:09.330 --> 03:25:16.610
Ken Pugh: So, just a requirement, a simple requirement, may have multiple tests associated with it.

1235
03:25:18.080 --> 03:25:21.190

Ken Pugh: So now let's talk just a moment about context.

1236
03:25:21.870 --> 03:25:24.640
Ken Pugh: Here we have our acceleration over here.

1237
03:25:24.760 --> 03:25:31.099
Ken Pugh: Push down on the pedal to the metal, and we get movement out. 0 to 60 and 5 seconds.

1238
03:25:31.440 --> 03:25:35.910
Ken Pugh: The test that we have is independent of the implementation.

1239
03:25:36.170 --> 03:25:46.240
Ken Pugh: It doesn't matter whether we're using something that has an engine, transmission, driveshaft, and wheels, such as a typical internal combustion engine.

1240
03:25:46.400 --> 03:25:53.340
Ken Pugh: or… We're doing it with an electric vehicle, a controller with some engines and some wheels.

1241
03:25:53.890 --> 03:25:56.170
Ken Pugh: We push pedal to the metal.

1242
03:25:57.040 --> 03:26:02.820
Ken Pugh: We turn on the stopwatch, and we reach, when we reach 60, did we get there in 5 seconds?

1243
03:26:02.970 --> 03:26:05.550
Ken Pugh: They have the same external behavior.

1244
03:26:06.070 --> 03:26:07.879
Ken Pugh: Or, do they really?

1245
03:26:08.680 --> 03:26:14.610
Ken Pugh: When you push pedal to the metal in an ICE, it goes…

1246
03:26:15.020 --> 03:26:19.019
Ken Pugh: When you push pedal to the metal, and an electric car, it goes…

1247
03:26:20.750 --> 03:26:24.059
Ken Pugh: So, maybe if we want the same external behavior.

1248
03:26:24.280 --> 03:26:29.630
Ken Pugh: We need to have a noise generator, maybe tuned to be a Porsche, or whatever you want to sound like.

1249
03:26:30.220 --> 03:26:38.990
Ken Pugh: So, the idea is that we're defining our behavior in external terms independent of implementation.

1250
03:26:40.330 --> 03:26:42.739
Ken Pugh: So let's give a sample behavior here.

1251
03:26:43.270 --> 03:26:53.559
Ken Pugh: Here's a sample business rule. Is the customer rating as good and the order total is less than or equal to, $10? Well, I'll let you read that.

1252
03:26:54.470 --> 03:26:56.419
Ken Pugh: And I have a question for you.

1253
03:26:57.240 --> 03:27:02.739
Ken Pugh: What is a discount for a good customer and a $50.01 order total?

1254
03:27:03.070 --> 03:27:05.700
Ken Pugh: I'll just give you a moment to think about that.

1255
03:27:06.140 --> 03:27:16.739
Ken Pugh: What did you come up with?

1256
03:27:17.060 --> 03:27:19.650
Ken Pugh: Well… 1%?

1257
03:27:20.800 --> 03:27:22.250
Ken Pugh: 5%!

1258
03:27:22.510 --> 03:27:25.550
Ken Pugh: I've done this, and some people come up with 6%.

1259
03:27:25.940 --> 03:27:32.830
Ken Pugh: Why the variation? Well… This business rule was deliberately ambiguous.

1260
03:27:33.150 --> 03:27:39.740

Ken Pugh: Now, I know that you probably never get an ambiguous business rule in your organization.

1261
03:27:40.370 --> 03:27:44.580
Ken Pugh: But, if you do, it's difficult to program against.

1262
03:27:44.770 --> 03:27:52.690
Ken Pugh: And AI will have the same difficulty in trying to write code for an ambiguous business rule as a human being.

1263
03:27:53.320 --> 03:28:09.480
Ken Pugh: So now, how can we help to clarify this business rule? Well, let's make up a give and win then. Given our total is $50.01 and the rating is good, when the discount is computed, then the percent is 1%.

1264
03:28:09.970 --> 03:28:15.010
Ken Pugh: We make sure with our customer that we interpreted it correctly, and we're good to go.

1265
03:28:15.630 --> 03:28:23.080
Ken Pugh: But chances are, we want to add a couple of more give and wins there. Oh, how about the $10 and 0%?

1266
03:28:23.350 --> 03:28:35.039
Ken Pugh: Or $10.01 and 1%. Have we got all of that right? And as David was talking with the triad, the customer, developer, and tester makes sure that everybody is on the same page.

1267
03:28:35.790 --> 03:28:44.349
Ken Pugh: In particular, because we're going to give this as our specification to AI, and we want to make sure we're giving them the right information.

1268
03:28:45.910 --> 03:28:49.960
Ken Pugh: So, let me differentiate between a specification and a test.

1269
03:28:50.450 --> 03:28:59.429
Ken Pugh: A specification. Give our total is $10.01, the rating is good. When the discount is computed, percent is 1%.

1270
03:29:00.230 --> 03:29:01.679
Ken Pugh: What is a test?

1271
03:29:02.320 --> 03:29:04.809

Ken Pugh: Given. When?

1272
03:29:04.990 --> 03:29:09.420
Ken Pugh: And now, we check that the percent is 1.

1273
03:29:09.810 --> 03:29:12.680
Ken Pugh: And we do that internally.

1274
03:29:13.070 --> 03:29:20.950
Ken Pugh: So… Specification and test, they're just two different slight variations on the same thing.

1275
03:29:22.620 --> 03:29:28.840
Ken Pugh: This is what we want, and the test is, we check that we got what the VIN says.

1276
03:29:30.620 --> 03:29:33.370
Ken Pugh: Now, let's take a look at this in a little more…

1277
03:29:33.500 --> 03:29:41.279
Ken Pugh: detail here. Here's our Gherkin for computer discount. We could come up with a lot of those little scenarios, but…

1278
03:29:41.530 --> 03:29:50.629
Ken Pugh: We could just come up with a table. Table that looks like something in the language of business, because the language of business is Excel.

1279
03:29:51.250 --> 03:29:52.060
Ken Pugh: F.

1280
03:29:52.670 --> 03:30:11.029
Ken Pugh: Business looks at it, and we all agree. And by the way, the testers look at it, and for those in testing, we come up with all the equivalence classes and the breakpoints. And so, we're going to have a business rule that is tested in its entirety.

1281
03:30:11.030 --> 03:30:14.319
Ken Pugh: As far as we can do it with a reasonable amount of time.

1282
03:30:15.250 --> 03:30:22.000
Ken Pugh: So, that's the Gherkin that we're going to be using as a prompt for our AI.

1283

03:30:23.100 --> 03:30:28.289
Ken Pugh: Let's do a little bigger thing here with a little flow and a context associated with it.

1284
03:30:28.580 --> 03:30:33.820
Ken Pugh: So here's the Tiktron company. They want to present online events for a chart.

1285
03:30:34.210 --> 03:30:39.630
Ken Pugh: As an event manager, I want to create events, sell tickets, and receive the proceeds.

1286
03:30:40.420 --> 03:30:48.600
Ken Pugh: So here's our flow. We're gonna create events, we're gonna sell tickets, we're gonna hold the event, and we're gonna get our money.

1287
03:30:49.370 --> 03:30:51.080
Ken Pugh: Standard workflow.

1288
03:30:51.990 --> 03:30:56.769
Ken Pugh: So we can describe this behavior at a high level.

1289
03:30:57.300 --> 03:31:00.299
Ken Pugh: Or, as we'll see in a little more detailed level.

1290
03:31:00.410 --> 03:31:09.430
Ken Pugh: So here's a scenario. Create an event. Given an event does not exist, when the producer creates it, then it becomes available for ticket sales.

1291
03:31:10.490 --> 03:31:24.519
Ken Pugh: Oh, how about selling the tickets? Given an event is available for ticket sales. When a purchaser purchases a ticket, then the purchaser receives the ticket in email, and the purchaser is charged for that ticket.

1292
03:31:25.490 --> 03:31:27.929
Ken Pugh: Now, this is a high-level description.

1293
03:31:28.480 --> 03:31:29.390
Ken Pugh: If…

1294
03:31:29.920 --> 03:31:39.060
Ken Pugh: The application that you're creating is similar to previous ones. AI probably knows about that, and this is all that you need.

1295
03:31:39.340 --> 03:31:49.499
Ken Pugh: But chances are, you're doing something custom, and it will have not experienced before. So, explicitness will beat implicitness.

1296
03:31:50.120 --> 03:32:08.229
Ken Pugh: So first, let me just show a context diagram here, since we talked about it. Here we got a purchaser and an event producer, and they deal with our online ticket system, and we know we're going to have an email server and a credit card processor to take payments, and some sort of a database or persistent storage.

1297
03:32:09.330 --> 03:32:15.169
Ken Pugh: So that's our context, then we're going to create things in… in that context.

1298
03:32:16.030 --> 03:32:19.869
Ken Pugh: So here is our detailed behavior.

1299
03:32:20.160 --> 03:32:21.969
Ken Pugh: Here's create an event.

1300
03:32:22.070 --> 03:32:29.210
Ken Pugh: Given it does not exist, We now create it. Adele is going to be…

1301
03:32:29.350 --> 03:32:37.850
Ken Pugh: have about 100 tickets, and only $5 each. Really good deal. I suggest you, get a ticket as soon as possible.

1302
03:32:39.280 --> 03:32:44.389
Ken Pugh: Once the event has been created, it's now going to be available for sale.

1303
03:32:45.480 --> 03:32:54.329
Ken Pugh: Notice we specified in this, more detail than just creating an event, all of the attributes in an event, from…

1304
03:32:54.900 --> 03:32:56.870
Ken Pugh: The customer's point of view.

1305
03:32:57.870 --> 03:33:00.109
Ken Pugh: From the external behavior point of view.

1306
03:33:00.950 --> 03:33:03.960

Ken Pugh: Here's another behavior, a little more detail.

1307
03:33:04.480 --> 03:33:05.950
Ken Pugh: Let's sell a ticket!

1308
03:33:06.930 --> 03:33:13.100
Ken Pugh: Given that we have that event available for sale, we're gonna purchase it.

1309
03:33:13.380 --> 03:33:16.300
Ken Pugh: Oh, we're gonna get an email.

1310
03:33:16.410 --> 03:33:25.309
Ken Pugh: And a credit card number. Notice we don't get into all the details of the dialogue, these are just going to be the inputs a purchaser would have to put in.

1311
03:33:26.010 --> 03:33:28.530
Ken Pugh: And now, what should occur?

1312
03:33:29.710 --> 03:33:34.709
Ken Pugh: Then, the purchaser receives a ticket in their email.

1313
03:33:35.910 --> 03:33:42.810
Ken Pugh: The purchaser gets charged for the ticket, for that credit card, and the number of tickets decrease.

1314
03:33:43.800 --> 03:33:57.159
Ken Pugh: Now that we get into something this detailed, we'd always go back with our triad and go, have we covered it? Is there any other detail that we should add here that we should tell our implementer about?

1315
03:33:58.790 --> 03:34:04.009
Ken Pugh: So… Let's look at a few more details, though.

1316
03:34:05.790 --> 03:34:07.470
Ken Pugh: Domain terms.

1317
03:34:08.400 --> 03:34:11.159
Ken Pugh: One of the key aspects of

1318
03:34:11.760 --> 03:34:16.010
Ken Pugh: Understanding a problem is understanding the domain terms.

1319
03:34:16.400 --> 03:34:18.239
Ken Pugh: So, here's an example.

1320
03:34:18.340 --> 03:34:20.559
Ken Pugh: Here's a domain term ticket count.

1321
03:34:20.890 --> 03:34:25.059
Ken Pugh: And it represents how many tickets are available, or maybe being sold.

1322
03:34:25.400 --> 03:34:30.069
Ken Pugh: So we have that minus 1 is not a valid number.

1323
03:34:30.330 --> 03:34:34.709
Ken Pugh: Zero is okay, because that's how many we might have left when we're sold out.

1324
03:34:35.010 --> 03:34:45.569
Ken Pugh: And we are going to have a maximum number of 1,000, and if we're over a thousand… excuse me, 10,000, that would be not valid.

1325
03:34:46.900 --> 03:34:52.030
Ken Pugh: Now we have a domain ticket count that might be used in our program.

1326
03:34:52.510 --> 03:34:54.899
Ken Pugh: Let's look like another domain term.

1327
03:34:55.570 --> 03:34:57.870
Ken Pugh: Here's their main term, percentage.

1328
03:34:58.510 --> 03:35:07.350
Ken Pugh: Once again, minus 1 is not valid, 0 to 100 is valid, and 101 is not valid.

1329
03:35:08.190 --> 03:35:09.690
Ken Pugh: And, of course.

1330
03:35:09.840 --> 03:35:18.739
Ken Pugh: 200, or 1,000, or 2,000, like some politicians like to use for discounts, of course, would not be valid at all.

1331
03:35:19.850 --> 03:35:23.230
Ken Pugh: So now we described our domain terms.

1332
03:35:23.450 --> 03:35:24.730
Ken Pugh: That then…

1333
03:35:25.130 --> 03:35:34.730
Ken Pugh: We tell AI, these are the types of things that we are dealing with. If you haven't seen them before, this is what we mean by percentage.

1334
03:35:35.650 --> 03:35:40.700
Ken Pugh: And then we can add our business rules. Here's our business rule for discount.

1335
03:35:40.970 --> 03:35:47.550
Ken Pugh: We're gonna give some discounts, 0% for 1, 5% for 2,

1336
03:35:47.650 --> 03:35:51.090
Ken Pugh: And… 10% if you got 6 or more.

1337
03:35:51.580 --> 03:35:57.400
Ken Pugh: And then our maximum number of tickets you can buy at any time is gonna be 100, or what have you.

1338
03:35:58.990 --> 03:36:03.299
Ken Pugh: So we've got domain terms and business rules.

1339
03:36:03.990 --> 03:36:08.310
Ken Pugh: And… As I mentioned before, when you get

1340
03:36:08.890 --> 03:36:20.029
Ken Pugh: Probably for some financial institutions, 50, 60, 70% of the code is all about business rules, and they have to be absolutely correct.

1341
03:36:20.210 --> 03:36:26.549
Ken Pugh: And that's why we need to have a scenario that shows what is correct.

1342
03:36:28.750 --> 03:36:31.919
Ken Pugh: And once again, then, we can also look at these.

1343
03:36:32.190 --> 03:36:37.269
Ken Pugh: As our triad, we get together and we go, have we covered all the possibilities?

1344
03:36:37.540 --> 03:36:52.320
Ken Pugh: And, especially those who have a tester perspective go, well, what happens if we try and create an event, and we put the date that's 2 weeks ago? Well, obviously, that should not be allowed, but let's make sure that it is

1345
03:36:52.630 --> 03:37:00.980
Ken Pugh: And how about trying to sell a ticket after the event's over? Well, once again, we should make sure that events are not available after that.

1346
03:37:01.770 --> 03:37:02.880
Ken Pugh: Oh!

1347
03:37:03.010 --> 03:37:20.840
Ken Pugh: What if the purchaser's purchase information is invalid? Should we just reject the order as a whole and say, come back later? Should we give them an opportunity? And so forth. So now we can start to have discussions on what the behavior should be on each of those.

1348
03:37:21.200 --> 03:37:33.119
Ken Pugh: Oh, and the worst one is, a purchaser does not receive a ticket. Oh, check your spam mail. You just put a message up. If you didn't get a ticket in one minute, check your spam.

1349
03:37:33.750 --> 03:37:40.139
Ken Pugh: That seems to be the number one excuse for not receiving something.

1350
03:37:40.580 --> 03:37:42.449
Ken Pugh: Even happened to me today.

1351
03:37:43.200 --> 03:37:44.240
Ken Pugh: Self!

1352
03:37:45.620 --> 03:37:51.139
Ken Pugh: Those are scenarios. They describe the behavior that we want.

1353
03:37:51.820 --> 03:37:54.640
Ken Pugh: And being pretty explicit about them, too.

1354
03:37:55.200 --> 03:37:59.170
Ken Pugh: So, how do we execute them and turn them into the test?

1355
03:37:59.860 --> 03:38:05.989

Ken Pugh: Well… We're going to give our Gherkin files as input.

1356
03:38:06.290 --> 03:38:13.540
Ken Pugh: With clawed code, I just put them in the source code, and tell Claude Code.

1357
03:38:14.120 --> 03:38:18.090
Ken Pugh: Here's your source file, here is your feature files.

1358
03:38:18.550 --> 03:38:20.179
Ken Pugh: Make it so.

1359
03:38:22.320 --> 03:38:27.349
Ken Pugh: And it doesn't. Sometimes not as good as other times, but a lot it doesn't.

1360
03:38:27.460 --> 03:38:28.330
Ken Pugh: Now…

1361
03:38:29.460 --> 03:38:36.770
Ken Pugh: Claude actually takes these… these Gherkin files and treats it two different ways, so you can be a little more specific.

1362
03:38:37.470 --> 03:38:50.000
Ken Pugh: One, I wanted to actually read the Gherkin file, and what's called glue code, which code is going to be connected to production code, and I say, create the GLUE code for me.

1363
03:38:50.810 --> 03:38:59.620
Ken Pugh: Alternatively, what Claude does is simply take the Gherkin file and translate it into unit tests in the language that you want.

1364
03:39:01.310 --> 03:39:03.389
Ken Pugh: That works a bit faster.

1365
03:39:03.960 --> 03:39:23.739
Ken Pugh: I prefer the former, because now I know I can take a Gergen file, make a few alterations to it, without changing any code, and make sure that the application is still working as desired. That worked for one ticket. I'm going to give you that same thing and ask for 8 tickets.

1366
03:39:23.970 --> 03:39:30.730
Ken Pugh: Or whatever… whatever your person with a tester perspective thinks should be done.

1367
03:39:31.540 --> 03:39:34.450
Ken Pugh: And now, here's the other possibility.

1368
03:39:34.680 --> 03:39:37.059
Ken Pugh: AI is the specifier.

1369
03:39:37.590 --> 03:39:41.210
Ken Pugh: You go, okay, I've given you this Gherkin file.

1370
03:39:41.820 --> 03:39:52.210
Ken Pugh: what other variations would you do? Would you say? Hmm, okay. I've done the hard work, I've broken it up into the pieces and everything.

1371
03:39:52.480 --> 03:39:54.329
Ken Pugh: Give me…

1372
03:39:54.510 --> 03:39:55.990
Rinat Sergeev: door, door-to-door.

1373
03:39:57.780 --> 03:40:04.430
Ken Pugh: Two or What's happening? Two or three different variations of that.

1374
03:40:04.810 --> 03:40:05.570
Ken Pugh: Okay.

1375
03:40:06.150 --> 03:40:10.590
Ken Pugh: So… We start with the Gherkin.

1376
03:40:10.710 --> 03:40:13.909
Ken Pugh: And we may let,

1377
03:40:14.520 --> 03:40:26.560
Ken Pugh: our AI actually create more Girkin for us, which is now readable, and we can check to see if that meets our expectations.

1378
03:40:28.840 --> 03:40:33.780
Ken Pugh: Now, I created a open source code called Gherkin Executor.

1379
03:40:34.250 --> 03:40:43.139
Ken Pugh: which takes the Gherkin, which Cucumber and some of the other BDD frameworks read, and I added what I call the data statement.

1380
03:40:43.570 --> 03:40:53.260
Ken Pugh: And the data statement gives a bit more information, i.e, it gives the data types for each of the table fields.

1381
03:40:53.710 --> 03:40:58.829
Ken Pugh: Now… The data statement gives more information to the AI.

1382
03:40:59.230 --> 03:41:07.830
Ken Pugh: Do we need it? Well, if it's having a hard time interpreting your domain, maybe so.

1383
03:41:08.750 --> 03:41:18.279
Ken Pugh: So, let me give an example of this. Here I have a domain, a, when we buy a ticket, and we have an event number, tickets, email, and so forth.

1384
03:41:19.210 --> 03:41:37.079
Ken Pugh: And now I have a data statement that basically says the event is just a string, or maybe an alphanumeric string. The number of tickets is a ticket count, so nobody should be able to enter less than… less than zero, a negative number, or greater than 100.

1385
03:41:37.230 --> 03:41:47.179
Ken Pugh: We have an email address, which should be a standard data type forever and ever, and that will ensure that at least are formatted correctly.

1386
03:41:47.550 --> 03:42:02.230
Ken Pugh: And finally, a credit card number, data type, which will make sure that when this appears on the screen, it will check to make sure that it's at least correctly formatted, not necessarily that it's a real credit card.

1387
03:42:03.520 --> 03:42:09.190
Ken Pugh: And we could do the same thing for our business rules. Here was our discount,

1388
03:42:09.300 --> 03:42:15.270
Ken Pugh: Business rule for our tickets. We'd say the number of tickets is going to be a ticket count.

1389
03:42:15.380 --> 03:42:20.330
Ken Pugh: And… The discount percentage is going to be a percentage.

1390
03:42:20.640 --> 03:42:21.620

Ken Pugh: Anne?

1391
03:42:22.060 --> 03:42:30.829
Ken Pugh: If we try to enter a business rule, or get a percentage that we gave it, oh, 101, it should be rejected.

1392
03:42:34.530 --> 03:42:41.489
Ken Pugh: All right, so the question is, how big of the steps, how big of the pieces do we need to do?

1393
03:42:42.360 --> 03:42:46.150
Ken Pugh: Here's one… couple of ways to break it into smaller steps.

1394
03:42:46.730 --> 03:42:52.660
Ken Pugh: And it's all part of my, phrase, act globally, think globally.

1395
03:42:54.280 --> 03:42:59.500
Ken Pugh: So let's decompose this flow, sort of like the slices that David was talking about.

1396
03:43:00.050 --> 03:43:10.009
Ken Pugh: Let's take that sell ticket, and we're gonna break it into searching for an event, creating an order, paying for the order, and sending the email.

1397
03:43:10.650 --> 03:43:23.040
Ken Pugh: We can have small scenarios for each one of those. And then, of course, paying for the order. Do we pay with a valid credit card? Let's see if we have an order that has a discount applied to it. Did that work?

1398
03:43:23.340 --> 03:43:38.580
Ken Pugh: and paying it with an invalid. And let's see whether it asks for another credit card, or if it keeps… you get 10 credit cards in, and you should finally reject it, because somebody's trying to,

1399
03:43:39.490 --> 03:43:42.110
Ken Pugh: Try a lot of invalid credit cards.

1400
03:43:42.370 --> 03:43:47.949
Ken Pugh: So… That's one way to decompose things into smaller steps.

1401
03:43:50.260 --> 03:43:54.230
Ken Pugh: Now, here's design… here's my concept of design.

1402
03:43:54.730 --> 03:44:00.689
Ken Pugh: Design composes… decomposes behavior into smaller behaviors.

1403
03:44:00.850 --> 03:44:07.100
Ken Pugh: And then, each of those smaller behaviors contributes to producing the larger behavior.

1404
03:44:07.760 --> 03:44:10.530
Ken Pugh: It's a matter of how do you break things up?

1405
03:44:10.720 --> 03:44:22.950
Ken Pugh: That is truly what design, at least on a large scale, is about. And of course, we might have multiple levels. We have a big level, and then smaller pieces, and smaller, and smaller pieces in everything.

1406
03:44:23.480 --> 03:44:32.059
Ken Pugh: So… AI, if it's having trouble, if you give it a big piece, and it has trouble understanding.

1407
03:44:32.770 --> 03:44:38.419
Ken Pugh: Break it into smaller pieces. Breaking into the individual steps in a flow.

1408
03:44:39.490 --> 03:44:51.539
Ken Pugh: And give it just one single step at a time, or maybe give it a couple of steps at a time as separate scenarios, so that it understands the interactions between the steps in a flow.

1409
03:44:52.710 --> 03:44:55.280
Ken Pugh: This is one of those times when

1410
03:44:56.060 --> 03:45:00.790
Ken Pugh: How much, how explicit, and how small do I need to be?

1411
03:45:01.900 --> 03:45:05.509
Ken Pugh: If you're too big, Bring it up.

1412
03:45:08.170 --> 03:45:20.459
Ken Pugh: And we can also decompose on the context. For example, instead of having a big context here, we could say, oh, we're going to have an inventory portion, and a payment portion, and a search portion.

1413
03:45:20.790 --> 03:45:27.649

Ken Pugh: And now, each of those scenarios would be against each one of those pieces.

1414
03:45:27.830 --> 03:45:41.019
Ken Pugh: In essence, you're designing the system into modules that have smaller behaviors that are going to be composed together to provide one of those bigger behaviors.

1415
03:45:41.540 --> 03:45:49.370
Ken Pugh: And in F… you are… somewhat designing the system, and then AI is going to produce the system.

1416
03:45:51.620 --> 03:45:58.150
Ken Pugh: So, in conclusion, What's the goal of writing Gherkin?

1417
03:45:59.030 --> 03:46:05.379
Ken Pugh: It's to replace misunderstanding with shared understanding.

1418
03:46:06.850 --> 03:46:09.550
Ken Pugh: When we just had all humans together.

1419
03:46:10.070 --> 03:46:15.480
Ken Pugh: We have the customer, the developer, and the tester

1420
03:46:16.050 --> 03:46:21.710
Ken Pugh: We replace a misunderstanding among them with a shared understanding.

1421
03:46:21.910 --> 03:46:26.750
Ken Pugh: And with AI, We now replace

1422
03:46:26.880 --> 03:46:34.359
Ken Pugh: That misunderstanding that AI may interpret us improperly with our shared understanding.

1423
03:46:36.170 --> 03:46:49.730
Ken Pugh: So, if you want to see examples, somebody asked for examples, go to attdd-bd on GitHub, and you'll see a test recorder from Claude.

1424
03:46:49.970 --> 03:46:52.280
Ken Pugh: This is actually a full stack

1425
03:46:52.660 --> 03:46:58.410
Ken Pugh: application, a UI, some logic, some database created by Claude.

1426
03:46:58.650 --> 03:47:05.209
Ken Pugh: Add… You can see the feature files which Claude used to create the application.

1427
03:47:05.500 --> 03:47:11.970
Ken Pugh: I did not change a line of code in this application, so if you don't like the code, blame Claude.

1428
03:47:12.290 --> 03:47:18.190
Ken Pugh: But in all passes the tests that are specified in the feature file.

1429
03:47:18.350 --> 03:47:30.839
Ken Pugh: And what this application does is it actually is a manual test thing, that you… you can create a manual test run… you run against an application, and it will record all the times that you ran it.

1430
03:47:31.580 --> 03:47:33.769
Ken Pugh: I did this 3 years ago.

1431
03:47:34.070 --> 03:47:41.349
Ken Pugh: Manually creating this application, I took the feature files from that application and gave it to Claude.

1432
03:47:41.950 --> 03:47:46.239
Ken Pugh: So… And then, we've got bowling from Claude.

1433
03:47:46.590 --> 03:47:56.590
Ken Pugh: the old-fashioned bowling program described in Gherkin, and Claude created the code from it. And there's some other examples in there as well.

1434
03:47:57.010 --> 03:48:04.100
Ken Pugh: Currently, I'm… I'm doing a… a web-based thing, for events, and Claude…

1435
03:48:04.430 --> 03:48:13.730
Ken Pugh: Seems to be doing a fine job of it, and once again, I'm not looking at the code, I am analyzing the behavior at the outside level.

1436
03:48:15.540 --> 03:48:17.519
Ken Pugh: So, if you want to contact me.

1437
03:48:17.900 --> 03:48:20.470

Ken Pugh: What I like to do is immersive training.

1438
03:48:20.630 --> 03:48:30.690
Ken Pugh: It used to be just in behavior-driven development and acceptance test-driven development, but now, since the new buzzword has come out, it's also in specification-driven development.

1439
03:48:30.800 --> 03:48:35.500
Ken Pugh: You can tag me at… contact me at Ken… at KenPugh.com.

1440
03:48:36.310 --> 03:48:40.780
Ken Pugh: So, connect with me on LinkedIn, or just, check out my website.

1441
03:48:42.220 --> 03:48:43.160
Ken Pugh: Oh.

1442
03:48:43.630 --> 03:48:46.059
Ken Pugh: This is not an ending, but a beginning.

1443
03:48:47.050 --> 03:48:48.600
Ken Pugh: Showed you how to…

1444
03:48:48.980 --> 03:48:58.610
Ken Pugh: Some examples of using Gherkin as prompts, well, actually, they're input, because they're actually part of your source code.

1445
03:48:58.860 --> 03:49:02.919
Ken Pugh: And having Claude simply write code against them.

1446
03:49:03.240 --> 03:49:05.490
Ken Pugh: At what level do you want to do this?

1447
03:49:05.720 --> 03:49:11.299
Ken Pugh: Maybe you start at that highest outline level, and if you're not getting the code you want.

1448
03:49:11.580 --> 03:49:22.420
Ken Pugh: Become more and more explicit, and if that's not producing it, or it's producing some… some clot is taking a while, then break it into smaller chunks.

1449
03:49:22.580 --> 03:49:25.229
Ken Pugh: smaller chunks, and just have claw.

1450
03:49:25.680 --> 03:49:28.519
Ken Pugh: Or AI do it one at a time.

1451
03:49:29.130 --> 03:49:33.809
Ken Pugh: All right, thank you. And if you got any questions, I'll be sitting right here.

1452
03:49:40.020 --> 03:49:49.549
David Gijsbers: Thanks, Ken. I actually did have a question. What are some of the typical barriers that you see within organizations that would prevent them from

1453
03:49:49.750 --> 03:49:51.020
David Gijsbers: Adopting these.

1454
03:49:52.310 --> 03:49:59.069
Ken Pugh: The same barriers I've seen preventing them 10 years ago from adopting BDD. Yeah.

1455
03:49:59.990 --> 03:50:02.250
Ken Pugh: It's,

1456
03:50:02.800 --> 03:50:17.690
Ken Pugh: I, I have given… I… I don't know how many times I have given this, my, my, couple-of-day workshop, and everybody goes, this is really great, but we don't have time to do this.

1457
03:50:17.890 --> 03:50:19.100
Ken Pugh: Right.

1458
03:50:19.290 --> 03:50:29.019
Ken Pugh: And it's like… and, you have time to look at an application that produces the wrong business rule, and then fix it.

1459
03:50:29.220 --> 03:50:31.240
Ken Pugh: So,

1460
03:50:31.660 --> 03:50:40.259
Ken Pugh: And the other… and I think the other, thing is the separation of… we have a testing group.

1461
03:50:40.410 --> 03:50:53.910

Ken Pugh: And we have a development group, and the testers don't even get involved until the developers have done something. And to actually sit down ahead of time and have testers and developers get together

1462
03:50:54.090 --> 03:50:55.090
Ken Pugh: Bob.

1463
03:50:55.240 --> 03:50:56.780
Ken Pugh: is… is foreign.

1464
03:50:57.220 --> 03:51:03.610
Ken Pugh: The ones that have done it, they go, wow, why weren't we doing this before? And the other ones…

1465
03:51:04.700 --> 03:51:07.989
Ken Pugh: Don't… Want to try it, for whatever reasons.

1466
03:51:11.030 --> 03:51:18.039
David Gijsbers: Do you think it's mostly cultural reasons, or is it, you know, just…

1467
03:51:19.800 --> 03:51:21.859
David Gijsbers: This is always the way we've done it.

1468
03:51:22.260 --> 03:51:24.390
David Gijsbers: It's just too hard to change behavior.

1469
03:51:25.200 --> 03:51:27.160
Ken Pugh: Okay.

1470
03:51:27.340 --> 03:51:30.259
Ken Pugh: I'll give you a couple of reasons, because…

1471
03:51:30.420 --> 03:51:47.050
Ken Pugh: the developers go, oh, that Gherkin stuff? Oh, that's so… no, no, no, that's not code, that isn't what we want. You know, if you guys produce it, yeah, we're gonna… we're gonna go and interpret it and put it into our own code itself.

1472
03:51:47.140 --> 03:51:57.900
Ken Pugh: So I'm gonna say there's… is that a cultural thing? They also complain that, the…

1473
03:51:57.920 --> 03:52:11.359

Ken Pugh: customers, don't have time to say completely… just like that business rule, filling out the things, they don't have time to do that. And the customers go, you know what I meant.

1474
03:52:13.060 --> 03:52:18.869
Ken Pugh: So, I… I… it's both maybe a cultural and a time-based.

1475
03:52:21.710 --> 03:52:23.939
David Gijsbers: Do people not view it as their job?

1476
03:52:28.550 --> 03:52:47.760
Ken Pugh: I would view it as their job, but I'm not part of their corporation, so… I… I really don't, I really don't know. I do know that a lot of developers do, I will call it, when I'm, on LinkedIn, when I even mention Gherkin, it's like, Gherkin, oh, that's so ugly!

1477
03:52:49.340 --> 03:52:59.450
Ken Pugh: And it's like, what? This is… in fact, I… I try and convince developers when I can, look at your unit tests.

1478
03:52:59.970 --> 03:53:19.009
Ken Pugh: look at the Gherkin, and I grant you, there is Gherkin out there that is really bad. So there's… and they maybe look… and if the… if the really bad Gherkin is what they're using, yeah, I can see them rejecting it. But the Gherkin that you saw in my examples here is, I'm going to call it, minimalistic Gherkin.

1479
03:53:19.080 --> 03:53:20.190
David Gijsbers: And… Right.

1480
03:53:20.460 --> 03:53:25.379
Ken Pugh: And, when they use that, It's very easy to implement.

1481
03:53:25.510 --> 03:53:35.950
Ken Pugh: If you use some of the bad Gherkin, it takes a long… it takes much longer to implement. And that's what they go. Why should we spend all this time

1482
03:53:36.460 --> 03:53:41.210
Ken Pugh: Well, you know, implementing this when it's faster to just write unit tests.

1483
03:53:43.100 --> 03:53:51.980
David Gijsbers: Yeah, David Vidra put a comment in the chat, AI gives us the best reason to actually learn modern software engineering, which…

1484
03:53:52.770 --> 03:53:57.800
David Gijsbers: Maybe, like, the, the headline… For the next…

1485
03:53:58.290 --> 03:54:02.330
David Gijsbers: conference that we do like this. So… .

1486
03:54:02.360 --> 03:54:03.270
Ken Pugh: Awesome.

1487
03:54:03.960 --> 03:54:05.639
Ken Pugh: I, I, I…

1488
03:54:05.740 --> 03:54:17.109
Ken Pugh: That… that is a perfect… that's a perfect thing, is, I mean, the perfect comment, because in essence, you should have been doing this before, and now, when you do this.

1489
03:54:17.400 --> 03:54:22.659
Ken Pugh: By the way, the developer will still be a part of this, but

1490
03:54:23.170 --> 03:54:28.860
Ken Pugh: They'll be looking at the code itself and making sure that it seems a little proper.

1491
03:54:29.040 --> 03:54:35.539
Ken Pugh: But… You don't have to… and the testers will still be doing some exploratory testing on this.

1492
03:54:35.890 --> 03:54:36.460
David Gijsbers: Yep.

1493
03:54:36.460 --> 03:54:40.870
Ken Pugh: on this standard stuff here. This is the way we want it to work.

1494
03:54:42.200 --> 03:54:46.120
Ken Pugh: It is either working or not working, and so…

1495
03:54:46.360 --> 03:54:51.120
David Gijsbers: I'm… I'm thinking of the… of the business results, right? At the…

1496
03:54:51.520 --> 03:54:56.390

David Gijsbers: End of the day, you're eliminating rework,

1497
03:54:59.400 --> 03:55:11.959
David Gijsbers: you know, if you get good at this, wonder how much faster you are in throughput. Do you have any kind of thoughts around those business metrics that might help?

1498
03:55:12.250 --> 03:55:13.330
David Gijsbers: to sell it.

1499
03:55:15.150 --> 03:55:19.109
Ken Pugh: Well, I'll tell you,

1500
03:55:19.780 --> 03:55:34.479
Ken Pugh: I always… I have this standard one, and we're talking about just using it for regular teams, and I had one team that said, we took this up, and all of a sudden, our team happiness factor increased.

1501
03:55:34.480 --> 03:55:35.090
David Gijsbers: Hmm.

1502
03:55:35.250 --> 03:55:36.040
David Gijsbers: Hi. Yup.

1503
03:55:36.040 --> 03:55:42.119
Ken Pugh: Because a tester was no longer getting everything at the end. They were testing throughout the sprint.

1504
03:55:42.280 --> 03:55:46.000
Ken Pugh: The lead developer and the tester were much happier.

1505
03:55:46.150 --> 03:55:54.599
Ken Pugh: Fewer production defects, fewer test environment defects, fewer everything. We're… we're happy. We don't know why we weren't doing this before.

1506
03:55:54.870 --> 03:56:10.879
Ken Pugh: Now, as far as measuring it on a productivity basis, I… I don't have… since I'm an independent consultant, I can't measure my own productivity, so I don't have productivity standards

1507
03:56:11.310 --> 03:56:16.110
Ken Pugh: from people, other than getting feedback, in fact, I will tell you that,

1508
03:56:16.660 --> 03:56:25.460
Ken Pugh: One place that is now doing just the equivalent of this, they have found

1509
03:56:26.070 --> 03:56:32.050
Ken Pugh: That, that doing this, you know, things are getting out.

1510
03:56:32.730 --> 03:56:38.540
Ken Pugh: Much more rapidly. I can't… I don't want to quote a number, but we'll call it much more rapidly.

1511
03:56:38.680 --> 03:56:39.550
Ken Pugh: So…

1512
03:56:40.580 --> 03:56:44.140
David Gijsbers: Yeah, I was just relating in my mind back to the…

1513
03:56:44.480 --> 03:56:51.060
David Gijsbers: Swarmio session, and then they, you know, they used the number of pull requests

1514
03:56:51.550 --> 03:56:56.589
David Gijsbers: As the, as one of their measures, and, what was the other one?

1515
03:57:00.380 --> 03:57:02.010
Jess Wolfe: There's batch size, there's.

1516
03:57:02.010 --> 03:57:02.920
David Gijsbers: That's right, yeah.

1517
03:57:02.920 --> 03:57:05.499
Jess Wolfe: It does, cycle time as well.

1518
03:57:06.560 --> 03:57:07.200
David Gijsbers: Yeah.

1519
03:57:07.730 --> 03:57:14.660
David Gijsbers: So, you know, I think, you know, bringing together those engineering metrics, and…

1520
03:57:14.790 --> 03:57:19.160
David Gijsbers: A behavior change like this might be a good way to…

1521
03:57:20.070 --> 03:57:24.349
David Gijsbers: Kind of measure the type of productivity increases that you're talking about.

1522
03:57:25.390 --> 03:57:32.500
Ken Pugh: And I'm going to add one other thing, and, you know, if you're creating an application through prompt.

1523
03:57:33.810 --> 03:57:47.830
Ken Pugh: where is the actual specification of exactly how this thing works? Now, some… I have seen instances where, you know, the AI can produce that spec, sort of, but then you…

1524
03:57:47.830 --> 03:57:54.080
Ken Pugh: It's not sometimes in the detail needed to say, oh, how does that business rule work?

1525
03:57:54.110 --> 03:57:56.820
Ken Pugh: So, the…

1526
03:57:56.820 --> 03:58:00.060
David Gijsbers: That's an interesting thought. Can you teach Claude Gherkin?

1527
03:58:01.210 --> 03:58:04.500
Ken Pugh: I… I give Kawad Gherkin, okay?

1528
03:58:04.500 --> 03:58:09.369
David Mantica: Yeah, Claw can output Gherkin as well. I know ChatGTP can output Gherkin, too.

1529
03:58:09.560 --> 03:58:10.190
Ken Pugh: Yep.

1530
03:58:11.460 --> 03:58:12.460
Ken Pugh: So…

1531
03:58:13.390 --> 03:58:23.639
Ken Pugh: And that's how I want to see my specs. But I don't want… I've seen… I've seen Claw on a… well, actually, I've used Copilot for this. I've seen them produce

1532
03:58:23.720 --> 03:58:39.400

Ken Pugh: scenarios, but they don't show any data in them. They're… they're a very… that high-level scenario, and I'm going, that's not what I want. That's… I want to see the detailed scenario with the data so I can see what the data flow through the scenario is.

1533
03:58:39.520 --> 03:58:45.010
Ken Pugh: And I haven't tried clogged yet to see if it's gotten able to create that.

1534
03:58:45.640 --> 03:58:46.350
David Gijsbers: Awesome.

1535
03:58:48.380 --> 03:58:54.499
David Gijsbers: Alright, well, let's open her up to the floor, see if there's any other… questions. I'll give people…

1536
03:58:54.640 --> 03:58:56.129
David Gijsbers: 15 seconds.

1537
03:58:59.640 --> 03:59:01.589
David Mantica: I thought you should give him 18 seconds.

1538
03:59:01.590 --> 03:59:02.420
David Gijsbers: 18?

1539
03:59:07.180 --> 03:59:11.920
David Gijsbers: How are you speaking with your microphone muted? That's what I would like to know.

1540
03:59:16.100 --> 03:59:18.120
David Mantica: Is Ken's microphone actually muted?

1541
03:59:18.460 --> 03:59:20.120
David Gijsbers: No, I thought you… you…

1542
03:59:20.120 --> 03:59:24.240
David Mantica: Oh, I'm magic. I got… I got back-end controls who you don't want…

1543
03:59:25.680 --> 03:59:33.130
David Gijsbers: Alright, awesome. Well, thanks so much, Ken. Priyanka, are you, on the line.

1544
03:59:33.730 --> 03:59:34.929

Priyanka Malkoti: Yep, I am.

1545
03:59:35.250 --> 03:59:38.599
David Gijsbers: Alright, awesome. Well, why don't we hand it over to you, and

1546
03:59:39.420 --> 03:59:41.589
David Gijsbers: You can just start a couple minutes early.

1547
03:59:43.770 --> 03:59:44.300
Priyanka Malkoti: Nope.

1548
03:59:45.110 --> 03:59:48.810
Priyanka Malkoti: Oh… Get my screen set up.

1549
03:59:54.670 --> 03:59:55.770
Priyanka Malkoti: Alright.

1550
03:59:56.610 --> 03:59:58.139
Priyanka Malkoti: Can you all see my screen?

1551
03:59:58.450 --> 03:59:59.590
David Gijsbers: Yeah, it looks great.

1552
03:59:59.810 --> 04:00:00.769
Dev Panajkar: Yes, we can.

How AI reshapes small and medium businesses

1553
04:00:01.880 --> 04:00:11.180
Priyanka Malkoti: Alright, so we are in hour 3. If you're still fully present, you deserve a batch. If you are half-present, I'll try to have the other half.

1554
04:00:11.470 --> 04:00:19.490
Priyanka Malkoti: So, let's start with the part everyone loves. AI running workflow sounds incredible, because it is.

1555
04:00:19.860 --> 04:00:26.600
Priyanka Malkoti: When AI executes well, a 12-person company suddenly operates like a 100-person company.

1556
04:00:28.340 --> 04:00:44.049

Priyanka Malkoti: I am not as… we all have been going through these sessions, and most of them have been technical about what the approach should be, and… and I think Zaki even showed us a live demo of how to set up those agents and how to make them work for you.

1557
04:00:44.780 --> 04:00:58.049
Priyanka Malkoti: So, you know, as we're talking about mostly SMBs here, I want to focus mostly on SMBs, right? When AI executes well, a 12% company can operate like a 100% company.

1558
04:00:58.270 --> 04:01:05.270
Priyanka Malkoti: And it's… it's because the same person has to wear multiple hats, and that works incredibly well.

1559
04:01:05.750 --> 04:01:07.860
Priyanka Malkoti: It… it is supporting…

1560
04:01:08.150 --> 04:01:23.179
Priyanka Malkoti: The customer is 24-7, leads get qualified instantly, invoices go out automatically, marketing campaigns personalize themselves. That's not optimization only. That's operational transformation.

1561
04:01:23.350 --> 04:01:27.570
Priyanka Malkoti: And that's the opportunity everyone wants to chase.

1562
04:01:28.000 --> 04:01:40.739
Priyanka Malkoti: But here is the uncomfortable question, and I think during the course of this, this event today, we have got some of those questions answered, or still are there.

1563
04:01:41.810 --> 04:01:46.330
Priyanka Malkoti: That is… if your AI sends the… I mean.

1564
04:01:46.470 --> 04:01:53.259
Priyanka Malkoti: I remember Zaki talking about one of his friends asking him if he should fire his accountant.

1565
04:01:53.390 --> 04:02:00.099
Priyanka Malkoti: Is that something we would want to do? If your AI is sending the wrong invoice to 400 customers overnight.

1566
04:02:00.200 --> 04:02:04.779
Priyanka Malkoti: what do we call it? Is that innovation, or is that a business incident?

1567
04:02:05.090 --> 04:02:11.470
Priyanka Malkoti: The risk is not AI execution. The risk is AI execution without reliability.

1568
04:02:12.050 --> 04:02:16.350
Priyanka Malkoti: And today, the gap between capability and reliability

1569
04:02:16.770 --> 04:02:20.610
Priyanka Malkoti: That's where most real business damage happens.

1570
04:02:20.900 --> 04:02:29.840
Priyanka Malkoti: So this isn't about prompts, it isn't about tools. It is not about which model is best. It is about this.

1571
04:02:30.190 --> 04:02:39.039
Priyanka Malkoti: Our software engineering determines whether AI becomes a compounding leverage or repeated operational chaos.

1572
04:02:39.400 --> 04:02:44.520
Priyanka Malkoti: Especially for small and mid-sized businesses that cannot afford chaos.

1573
04:02:45.540 --> 04:02:54.659
Priyanka Malkoti: Okay, so let's do a quick experiment, since I can't see all of you, so if you can just raise your hand through the reaction or the chat.

1574
04:02:55.290 --> 04:03:01.289
Priyanka Malkoti: Raise your hand if you've seen an impressive AI demo in the last 6 months.

1575
04:03:04.330 --> 04:03:07.130
Priyanka Malkoti: All right, great. Thanks, David. G.

1576
04:03:08.320 --> 04:03:09.330
Priyanka Malkoti: Nope.

1577
04:03:09.560 --> 04:03:11.199
Priyanka Malkoti: Keep your hand up.

1578
04:03:11.650 --> 04:03:17.179
Priyanka Malkoti: If that same demo is running a mission-critical workflow today.

1579
04:03:20.760 --> 04:03:21.510
Priyanka Malkoti: Correct.

1580
04:03:21.620 --> 04:03:27.070
Priyanka Malkoti: That's… That's a… that's the drop, that is the AI maturity gap.

1581
04:03:27.360 --> 04:03:34.890
Priyanka Malkoti: Forrester reports that only 10-15% of AI pilots actually scale long-term.

1582
04:03:35.010 --> 04:03:40.040
Priyanka Malkoti: Not because the models fail, because the systems around them are not trusted.

1583
04:03:40.240 --> 04:03:54.670
Priyanka Malkoti: We trust AI to draft. We don't trust it to execute. And that trust gap is not a model problem. It is an engineering problem. Because the moment AI touches revenue.

1584
04:03:54.780 --> 04:03:57.440
Priyanka Malkoti: Customers, billing, or contracts?

1585
04:03:57.630 --> 04:03:59.339
Priyanka Malkoti: The state changes.

1586
04:03:59.630 --> 04:04:03.959
Priyanka Malkoti: And for SMB, that stake is ImageA.

1587
04:04:06.760 --> 04:04:10.110
Priyanka Malkoti: So, this is where the shift really happens.

1588
04:04:10.380 --> 04:04:14.559
Priyanka Malkoti: Right? We're talking about two different ways how we handle AI.

1589
04:04:14.710 --> 04:04:17.400
Priyanka Malkoti: One is assistance, the other is delegation.

1590
04:04:17.900 --> 04:04:20.030
Priyanka Malkoti: Assistance is safe.

1591

04:04:20.160 --> 04:04:25.279
Priyanka Malkoti: AI drafts an email, you review it, you click send. If it is wrong, you catch it.

1592
04:04:26.000 --> 04:04:28.740
Priyanka Malkoti: Delegation, on the other hand.

1593
04:04:29.500 --> 04:04:38.730
Priyanka Malkoti: is when everything is AI. AI draft, AI sends, AI updates, CRM, AI triggers billing, everything.

1594
04:04:39.270 --> 04:04:43.120
Priyanka Malkoti: Now what happens? The customer becomes the QA.

1595
04:04:43.600 --> 04:04:48.649
Priyanka Malkoti: And when the customer is QA, They are always very honest.

1596
04:04:49.400 --> 04:05:00.200
Priyanka Malkoti: We've already seen what happens with that in the real world. I'm sure you must have all heard about Air Canada's chatbot giving the customer incorrect refund information.

1597
04:05:00.790 --> 04:05:05.120
Priyanka Malkoti: The airline was actually held legally responsible, because it was their chatbot.

1598
04:05:05.410 --> 04:05:10.479
Priyanka Malkoti: The AI didn't malfunction. It did exactly what the system allowed it to do.

1599
04:05:10.840 --> 04:05:23.689
Priyanka Malkoti: We've seen AI support agents, they're closing tickets too aggressively, AI invoices, they mishandle some edge cases, they're not escalating at the right time. The pattern is the same.

1600
04:05:24.280 --> 04:05:39.409
Priyanka Malkoti: the AI didn't go rogue. The boundaries were unclear, right? And that's not a model failure. That's a system design. Now, as David Wee was talking about the spec-driven development, right?

1601
04:05:39.570 --> 04:05:54.000
Priyanka Malkoti: there might be something that we need to have, which is called the control-driven development, and it kind of closely relates to the test-driven development as well. So, a lot of those specs that we can work around.

1602
04:05:54.880 --> 04:05:55.850
Priyanka Malkoti: No.

1603
04:05:56.420 --> 04:05:58.440
Priyanka Malkoti: Coming back to the SMBs.

1604
04:05:58.770 --> 04:06:05.989
Priyanka Malkoti: This is something very interesting, right? There is… there is a myth that SMBs are behind in AI. They're actually not.

1605
04:06:06.360 --> 04:06:13.220
Priyanka Malkoti: According to Intuit, there are nearly 89% of SMBs which are AI active.

1606
04:06:13.640 --> 04:06:28.830
Priyanka Malkoti: In many cases, they adopt faster than enterprises, as we were looking at Zaki, and how he's been able to… he can make those changes in a day. That's why the adoptation is easier. At the same time, when David Wee was talking about

1607
04:06:28.930 --> 04:06:44.200
Priyanka Malkoti: solely working on it, and how a bigger enterprise, that change takes longer. Because, of course, because of the scale, because of the number of applications, and the interdependency, it is not easy to bring those changes to big enterprises.

1608
04:06:44.920 --> 04:06:46.420
Priyanka Malkoti: In many cases.

1609
04:06:46.750 --> 04:07:00.230
Priyanka Malkoti: the SMBs, they adopt faster, because they don't have layers. They don't have extra headcount. They don't have buffer. AI isn't innovation for them. It is survival leverage.

1610
04:07:01.820 --> 04:07:08.270
Priyanka Malkoti: But here's the part that changes everything. They also have the smallest margin for error.

1611
04:07:08.850 --> 04:07:18.460
Priyanka Malkoti: When a Fortune 500 company can absorb a $50,000 automation mistake for an SMB, that might be payroll.

1612

04:07:18.890 --> 04:07:21.630
Priyanka Malkoti: That might be vendor payments. So…

1613
04:07:21.780 --> 04:07:28.420
Priyanka Malkoti: Reliability engineering in SMB AI system is not a nice-to-have.

1614
04:07:28.630 --> 04:07:30.720
Priyanka Malkoti: It's financial protection.

1615
04:07:31.160 --> 04:07:34.889
Priyanka Malkoti: And that changes how we should design.

1616
04:07:35.770 --> 04:07:36.490
Priyanka Malkoti: Got it.

1617
04:07:37.270 --> 04:07:48.540
Priyanka Malkoti: So, here is about the abandonment curve. So, Gartner projects that around 40% of agentic initiatives will be abandoned by 2027.

1618
04:07:49.140 --> 04:07:57.150
Priyanka Malkoti: And it's not because agents don't work, but because they are deployed without clear ownership, player boundaries.

1619
04:07:57.420 --> 04:07:59.270
Priyanka Malkoti: Clear failure paths.

1620
04:07:59.630 --> 04:08:09.899
Priyanka Malkoti: So… Open question, right? When your AI makes a bad decision that can cost you real money.

1621
04:08:10.480 --> 04:08:11.770
Priyanka Malkoti: Who owns it?

1622
04:08:12.470 --> 04:08:14.480
Priyanka Malkoti: Do you think it's engineering?

1623
04:08:14.820 --> 04:08:21.600
Priyanka Malkoti: ops… product, like… The vendor, the model.

1624
04:08:23.030 --> 04:08:31.359
Priyanka Malkoti: We're living in a world where the LLMs are changing… Faster than we can adapt.

1625
04:08:31.650 --> 04:08:41.869
Priyanka Malkoti: Right? But when something breaks, teams are arguing, was it the prompt? Was it the model? Was it the API? Was it the integration?

1626
04:08:43.120 --> 04:08:47.279
Priyanka Malkoti: That ambiguity… Is expensive, right?

1627
04:08:47.710 --> 04:08:55.069
Priyanka Malkoti: And in SMBs, ambiguity moves directly to impact. There is no 3-month review committee.

1628
04:08:55.300 --> 04:09:01.629
Priyanka Malkoti: there is immediate response. Go ahead, Toby, do you want to answer that question? If AI fails.

1629
04:09:01.630 --> 04:09:12.329
Toby Rao: No, no, I had a question, actually, a curiosity. So, I see 3 reasons, integration, complexity, unclear ownership, weak or… but, you know, it's so dynamic, I mean.

1630
04:09:12.470 --> 04:09:16.149
Toby Rao: like, a month and a half ago, I didn't even know about Claude. Two weeks.

1631
04:09:16.150 --> 04:09:16.630
Priyanka Malkoti: Lord.

1632
04:09:16.630 --> 04:09:22.949
Toby Rao: robot cursor, and now, like, in my organization, Claude is

1633
04:09:23.210 --> 04:09:26.660
Toby Rao: being used so much, right? So, can…

1634
04:09:27.090 --> 04:09:38.139
Toby Rao: a technology advancement that has come, and we were using an older technology. Could that also be a reason for abandoning an agent that was created?

1635
04:09:38.680 --> 04:09:39.419
Toby Rao: Just kidding.

1636
04:09:39.420 --> 04:09:47.240

Priyanka Malkoti: Possibly could be. See, think about it. As I said, like, until 6 months back, OpenAI was the go-to

1637
04:09:47.350 --> 04:10:01.659
Priyanka Malkoti: GPT for everyone. Everyone, like, anyone, they go hand in hand. AI and ChatGPT was just going hand-in-hand. Everybody was talking about it. And then you had Claude, or Anthropy came by and said, okay, let me

1638
04:10:01.720 --> 04:10:17.270
Priyanka Malkoti: I already know what the issues with OpenAI are, and I'm going to come up with a model which is smarter, because most of those founders were part of the OpenAI team, right? Yeah. So this is going to change. It is not always going to be which L&M we are using.

1639
04:10:17.570 --> 04:10:24.189
Priyanka Malkoti: You see, when we were talking about whether this event had just started, we were talking about garbage in, garbage out.

1640
04:10:24.350 --> 04:10:27.439
Priyanka Malkoti: the… when we talk about data. So…

1641
04:10:27.570 --> 04:10:42.570
Priyanka Malkoti: that is, I think, what you feed the system is what is going to come out. So, LLMs are going to change. All these… these… they're going to be one API today, one agent tomorrow. That is going to change. We…

1642
04:10:42.650 --> 04:10:58.499
Priyanka Malkoti: have to understand how our existing system is behaving. What are the challenges that it already has? What are the loopholes around? And build around it. Make sure that the agents that we are introducing

1643
04:10:58.750 --> 04:11:15.040
Priyanka Malkoti: They know the system, and that is something that, you know, the kind of five questions that will come up later in the slides, that what we need to ask ourselves when we are designing these AI systems, because they are critical. They are critical to the whole agent tech AI design.

1644
04:11:15.470 --> 04:11:16.380
Priyanka Malkoti: End.

1645
04:11:16.550 --> 04:11:23.890
Priyanka Malkoti: It… small businesses get the impact a lot more than, as I just said, compared to the enterprises.

1646

04:11:23.890 --> 04:11:45.489
Priyanka Malkoti: And if we even take… and, you know, we've been talking about slicing the problem, and coming up with solutions, and how, in bigger organizations, it is difficult to bring that change. So even if we take a small part of the entire organization as an SMB, and work on any system that we want to improve, or we want to enhance, or make more efficient.

1647
04:11:45.490 --> 04:12:00.820
Priyanka Malkoti: and we start there, that could guide us as to how that can be scaled. So, the abandonment is not because of how the technology is changing, how the LLMs are changing, it is about how you are designing your system.

1648
04:12:01.570 --> 04:12:02.460
Priyanka Malkoti: So…

1649
04:12:02.840 --> 04:12:03.350
Toby Rao: Thank you, Leon.

1650
04:12:03.350 --> 04:12:05.760
Priyanka Malkoti: Relying into… yeah, alright, thanks.

1651
04:12:07.120 --> 04:12:10.030
Priyanka Malkoti: So…

1652
04:12:10.390 --> 04:12:27.829
Priyanka Malkoti: the engineer mindset shift, right? We train developers to write deterministic logic. If X is this, then do Y. But AI systems don't return certainty. They return probability. And that changes everything, because now we are not just writing features.

1653
04:12:28.070 --> 04:12:35.600
Priyanka Malkoti: We are encoding business judgment. If your AI auto-approves refunds, you're defining financial policy.

1654
04:12:36.040 --> 04:12:41.970
Priyanka Malkoti: If your AI updates deal stages, you are influencing revenue forecasting.

1655
04:12:42.780 --> 04:12:47.849
Priyanka Malkoti: If your AI is sending customer emails, you're shaping brand trust.

1656
04:12:47.960 --> 04:12:50.549
Priyanka Malkoti: That's responsibility expansion.

1657
04:12:50.840 --> 04:12:55.280
Priyanka Malkoti: And most engineering training hasn't caught up to that shift yet.

1658
04:12:56.710 --> 04:12:59.909
Priyanka Malkoti: So, whenever we are designing.

1659
04:13:00.070 --> 04:13:14.360
Priyanka Malkoti: an AI-based system, there are a few questions that help us design that better. What decisions can this AI system make autonomously? What happens when the AI agent is

1660
04:13:14.630 --> 04:13:21.519
Priyanka Malkoti: uncertain, and this is… on this one, I think Lara, she's the one who started, the,

1661
04:13:21.660 --> 04:13:39.680
Priyanka Malkoti: the discussion, when it was talking about how it fails, that is important. Like, when it is uncertain, how are we going to determine what does it do when it is uncertain? It doesn't… what are the next steps? That helps us make those system designs.

1662
04:13:40.760 --> 04:13:50.749
Priyanka Malkoti: at what confidence level does it escalate? What actions require dual confirmation? So, in the case of the Air Canada.

1663
04:13:50.840 --> 04:13:56.369
Priyanka Malkoti: It was a request for refund, and that the policy

1664
04:13:56.440 --> 04:14:11.480
Priyanka Malkoti: mentioned that, you know, it cannot be done after the event has happened. So, if that confirmation would have been there, if that check would have kept in place, then probably that mishap wouldn't have happened.

1665
04:14:12.100 --> 04:14:15.420
Priyanka Malkoti: Can I replay and audit its decisions?

1666
04:14:15.700 --> 04:14:29.320
Priyanka Malkoti: That is extremely important. You know, when you… especially if you plan to have a lot of dependency on your agents, you want to see if it failed anywhere, if you can go back and replay

1667
04:14:29.920 --> 04:14:37.189

Priyanka Malkoti: what decisions it made. Because that, again, comes back to when we talk about, the control

1668
04:14:38.060 --> 04:14:48.839
Priyanka Malkoti: the SDD, which was a spec-driven, and then the control-driven. So the specs are going to determine how it needs to work, and the controls are going to help it make that decision.

1669
04:14:51.130 --> 04:14:55.180
Priyanka Malkoti: Alright. Now, because… when those…

1670
04:14:55.510 --> 04:15:13.249
Priyanka Malkoti: Decisions aren't defined. Incidents look like this, like, an AI auto-approves refunds it was only meant to recommend. A sales agent sends an email it wasn't confident about. A billing discrepancy appears, and no one can trace it, because we were not auditing or replaying the decisions.

1671
04:15:13.360 --> 04:15:23.599
Priyanka Malkoti: Pipeline numbers drift silently after a rules update. Something happened, and we saw what happened with CloudShare last year.

1672
04:15:23.790 --> 04:15:30.029
Priyanka Malkoti: Most AI incidents are not model failures. They are system decisions that were never explicitly made.

1673
04:15:30.390 --> 04:15:33.960
Priyanka Malkoti: And silence in design becomes noise in production.

1674
04:15:36.780 --> 04:15:37.560
Priyanka Malkoti: Here.

1675
04:15:37.770 --> 04:15:52.009
Priyanka Malkoti: But here is the real opportunity, right? AI will run more workflows. That is not optional. That is here with us. It's gonna stay. SMVs will continue adopting. That's not optional.

1676
04:15:52.480 --> 04:15:54.870
Priyanka Malkoti: The only real choice is this.

1677
04:15:55.230 --> 04:16:01.140
Priyanka Malkoti: Will those systems be engineered deliberately, or will they emerge accidentally?

1678

04:16:01.840 --> 04:16:08.899
Priyanka Malkoti: If AI is going to act like a critical team partner inside small and mid-sized businesses.

1679
04:16:09.100 --> 04:16:11.940
Priyanka Malkoti: It must be engineered like one.

1680
04:16:12.890 --> 04:16:15.069
Priyanka Malkoti: It needs to have clear authority.

1681
04:16:15.170 --> 04:16:19.449
Priyanka Malkoti: Clear boundary, clear monitoring, and clear accountability.

1682
04:16:19.950 --> 04:16:25.110
Priyanka Malkoti: Because in SMB environments, reliability is not scalability.

1683
04:16:25.390 --> 04:16:28.070
Priyanka Malkoti: Reliability is survival.

1684
04:16:28.650 --> 04:16:33.840
Priyanka Malkoti: And for all the engineers in this room, This is the opportunity, because…

1685
04:16:34.060 --> 04:16:38.299
Priyanka Malkoti: Software engineering is now becoming business engineering.

1686
04:16:38.740 --> 04:16:46.710
Priyanka Malkoti: And the people who understand that shift early, they will be the one who will build the companies that scale safely.

1687
04:16:48.150 --> 04:16:55.410
Priyanka Malkoti: that is most of what I wanted to talk about, so… and if you have made it through almost 3 and…

1688
04:16:56.280 --> 04:17:02.889
Priyanka Malkoti: 3 hours and 15 minutes with me, I'm pretty sure you all deserve a coffee. David, is that part of it? Virtual coffee?

1689
04:17:06.440 --> 04:17:07.320
Priyanka Malkoti: Alright.

1690
04:17:08.290 --> 04:17:09.140

David Mantica: Dave, you're on mute!

1691
04:17:09.140 --> 04:17:10.200
Priyanka Malkoti: Has any questions?

1692
04:17:10.760 --> 04:17:11.970
David Mantica: David G, you're on mute.

1693
04:17:12.910 --> 04:17:21.359
David Gijsbers: I thought I hit the unmute button, but, I didn't. Yes, free coffee, everyone, if you have a pot at home.

1694
04:17:22.890 --> 04:17:28.480
David Gijsbers: So I wrote down, software engineering is business engineering. I think that is…

1695
04:17:28.650 --> 04:17:33.519
David Gijsbers: Awesome, and in a nutshell, a lot of what we talked about today.

1696
04:17:33.820 --> 04:17:42.330
David Gijsbers: I know you're all championing at the bit for… oh, sorry, before we, move on, anybody have any questions for Priyanka?

1697
04:17:43.660 --> 04:17:46.119
Sterling Fulton: We measure from the state of born, every few months.

1698
04:17:46.120 --> 04:17:51.420
Jess Wolfe: No questions, really. I just want to applaud you on the presentation. It was really well done.

1699
04:17:52.040 --> 04:18:10.169
Priyanka Malkoti: Thank you, thank you. That's what we do partly as well. So, the AI part is the interesting part, and the designing part is the creative part, and that is what I was talking about earlier, how, the UI design is very… if it is intuitive.

1700
04:18:10.170 --> 04:18:15.080
Priyanka Malkoti: It… it likes… people like to see it. It gets people engaged. Thanks, Jess.

1701
04:18:16.220 --> 04:18:16.990
David Gijsbers: Alright.

1702

04:18:17.340 --> 04:18:29.220
Dev Panajkar: Priyanka, this is Deb. A quick question, please. You know, you did rightly say that for, small and medium businesses, the, the,

1703
04:18:29.220 --> 04:18:41.689
Dev Panajkar: a revenue curve or the impact curve is much shorter compared to bigger companies being impacted by a mistake done by chatbots or, you know, AI agents and so on.

1704
04:18:41.690 --> 04:18:55.109
Dev Panajkar: What other aspects do you think differentiate small and medium companies with respect to big companies, purely from an adoption standpoint with respect to AI agents or agentic AI and so on?

1705
04:18:56.130 --> 04:19:15.660
Priyanka Malkoti: So, as I was saying later as well, is that this small size helps them adopt faster, right? I know what is working for me, I know how my business works, it is a working model. I also know the challenges that I face, right? So, when we are talking about

1706
04:19:15.660 --> 04:19:18.490
Priyanka Malkoti: This pegdriven development.

1707
04:19:18.520 --> 04:19:20.049
Priyanka Malkoti: I know what my specs are.

1708
04:19:20.090 --> 04:19:30.630
Priyanka Malkoti: I know how… what I need to focus on. I know how my workflow works. So that makes it a lot easier for small to mid-sized businesses to adopt these agents.

1709
04:19:30.810 --> 04:19:35.600
Priyanka Malkoti: Whereas when we're talking about big enterprises, like for Chase.

1710
04:19:35.850 --> 04:19:44.170
Priyanka Malkoti: Taking… by the time a legacy application has been modernized, it already becomes a legacy architecture.

1711
04:19:44.510 --> 04:20:00.210
Priyanka Malkoti: Because that is the challenge. That is the challenge that enterprises face, and I think that is something that David… we was talking about as well, when he was talking about the scaling being the challenge for enterprises. So, I think that ways, they're pretty fast.

1712

04:20:00.210 --> 04:20:12.549
Priyanka Malkoti: They are not solely dependent on one LLM or one agent or the other. They know what is working best for them in terms of efficiency, in terms of revenue, and they are very quick to adapt.

1713
04:20:12.850 --> 04:20:20.350
Priyanka Malkoti: So, I think that that is something that works for them, but at the same time, you know, if something goes wrong, then the impact is broader as well.

1714
04:20:20.670 --> 04:20:22.610
Priyanka Malkoti: Thank you for that question.

1715
04:20:23.170 --> 04:20:23.760
Dev Panajkar: Makes sense.

1716
04:20:23.760 --> 04:20:43.209
Priyanka Malkoti: I know this is probably… my presentation was probably the least technical, because I knew everybody was going to talk about it, and when we… you have already seen a lot of live demos, it is not something that… I thought you needed a break from that, so it is mostly about looking at, you know, what are the challenges that this can bring.

1717
04:20:45.480 --> 04:20:57.620
David Gijsbers: All right, thanks. Well, why don't we, head into our last session? Dev, you're… you're on the screen and you've got the microphone, so we'll hand it over to you for our last session. Thanks so much. Thanks, Priyanka.

1718
04:20:57.620 --> 04:20:58.600
Dev Panajkar: Fantastic.

1719
04:21:00.160 --> 04:21:01.050
Sterling Fulton: You make a lot more…

1720
04:21:01.050 --> 04:21:03.370
Dev Panajkar: Let me know if you can see my screen.

1721
04:21:03.630 --> 04:21:04.530
Sterling Fulton: More than one foot.

1722
04:21:05.300 --> 04:21:09.529
Sterling Fulton: for a while, but it makes your stuff different, and can lead to altitude sickness.

1723

04:21:09.530 --> 04:21:10.969
Dev Panajkar: Are you able to see my screen?

1724
04:21:10.970 --> 04:21:11.390
David Gijsbers: Yes.

Turning AI Ideas Into Impact: Delivering Value with Purpose

1725
04:21:12.360 --> 04:21:24.740
Dev Panajkar: Fantastic. Many organizations are excited about AI, right? You know, we just heard about so many presentations speak the same thing. Small and medium companies, large companies, and so on.

1726
04:21:24.740 --> 04:21:33.939
Dev Panajkar: They're investing, piloting, and building prototypes. The energy is real. But here's the uncomfortable truth, right? Most AI initiatives fail.

1727
04:21:34.030 --> 04:21:41.060
Dev Panajkar: A recent report from Harvard Business Review pegs the number of failure as high as 90%.

1728
04:21:41.290 --> 04:21:42.290
Sterling Fulton: And…

1729
04:21:42.290 --> 04:21:55.910
Dev Panajkar: I just read there as well that most AI initiatives don't fail or don't fall because the technology doesn't work. They fail because the organization doesn't know how to turn innovation into impact.

1730
04:21:56.110 --> 04:22:09.460
Dev Panajkar: And that's what this talk is about. Hi, I'm Dev Panachkar. As mentioned, after demo failures and so on, momentum for AI initiatives fail. Ownership is unclear, budgets burn.

1731
04:22:09.510 --> 04:22:18.610
Dev Panajkar: So today, I want to walk you through a practical way to solve this. And by the end of this presentation, you will understand 3 key things.

1732
04:22:18.690 --> 04:22:31.850
Dev Panajkar: why most AI efforts stall, what good AI innovation actually looks like, and most importantly, how to build an idea-to-impact engine that scales across enterprise. So let's get to it.

1733
04:22:32.300 --> 04:22:36.360

Sterling Fulton: What does AI innovation look like in most organizations today?

1734
04:22:36.870 --> 04:22:54.679
Dev Panajkar: As described earlier, it usually starts with excitement and energy, and then comes the workshops, committees, vendors, pilots, and so on. None of that is bad, until the process becomes a product. It gets expensive, slow, and heavy.

1735
04:22:54.840 --> 04:22:57.959
Dev Panajkar: There are a lot of activities, but little adoption.

1736
04:22:58.070 --> 04:23:03.430
Dev Panajkar: Eventually, companies start believing, hey, we've tried AI, but it didn't work.

1737
04:23:03.700 --> 04:23:07.219
Dev Panajkar: But AI didn't fail. The system around it did.

1738
04:23:07.430 --> 04:23:15.770
Dev Panajkar: What's missing is a repeatable way to evaluate, fund, implement, and scale the right ideas.

1739
04:23:17.410 --> 04:23:29.240
Dev Panajkar: So, if the trap is that AI innovation becomes expensive, slow, and overly complicated, then the natural question is, what does good AI innovation actually look like?

1740
04:23:29.400 --> 04:23:37.209
Dev Panajkar: And what I want to be clear here is that good AI innovation isn't about more pilots or more ideas.

1741
04:23:37.310 --> 04:23:39.290
Dev Panajkar: Well, ideas are cheap.

1742
04:23:39.420 --> 04:23:41.390
Dev Panajkar: But impact is expensive.

1743
04:23:41.500 --> 04:23:49.260
Dev Panajkar: What works is a system that moves from idea, validation, ownership, implementation, to outcomes.

1744
04:23:49.380 --> 04:23:54.229
Dev Panajkar: It replaces chaos with clarity and bureaucracy with speed.

1745
04:23:54.540 --> 04:24:09.770
Dev Panajkar: The winners in AI aren't the most creative, they are the most consistent in turning ideas into value. So let's start with the foundation, because AI innovation doesn't need more intent, it needs structure.

1746
04:24:10.130 --> 04:24:13.990
Dev Panajkar: So, if we agree that AI innovation needs a system.

1747
04:24:14.390 --> 04:24:18.029
Dev Panajkar: Then we have to start with something most organizations skip.

1748
04:24:18.250 --> 04:24:21.589
Dev Panajkar: A clear vision and mission for AI innovation.

1749
04:24:21.710 --> 04:24:24.820
Dev Panajkar: Because without this, AI becomes random.

1750
04:24:24.940 --> 04:24:33.290
Dev Panajkar: So here's the vision, to enable responsible, scalable AI innovation that delivers measurable business impact.

1751
04:24:33.660 --> 04:24:45.810
Dev Panajkar: That sentence matters, because it includes 3 key words most companies don't operationalize. Responsible, scalable, measurable. Responsible. What should we build?

1752
04:24:46.190 --> 04:24:48.919
Dev Panajkar: you know, scalable. Can we grow?

1753
04:24:49.050 --> 04:24:55.890
Dev Panajkar: measurable. Does it move outcomes? Now, the mission is what makes the vision real.

1754
04:24:59.360 --> 04:25:18.379
Dev Panajkar: to provide a structured approach, governed approach, that transforms AI ideas into operational outcomes across the organization, not hype, not curiosity, adopted trusted working AI. With that foundation, repeatability becomes possible, and that's what we are going to look at next.

1755
04:25:18.830 --> 04:25:25.219
Dev Panajkar: This is the turning point of the entire presentation, because most organizations right now have plenty of intent.

1756
04:25:25.380 --> 04:25:27.680
Dev Panajkar: The issue is not desire, of course.

1757
04:25:27.840 --> 04:25:30.190
Dev Panajkar: The issue is the absence of a system.

1758
04:25:30.350 --> 04:25:37.050
Dev Panajkar: And without a system, AI innovation becomes one of two things. Either it becomes chaos.

1759
04:25:37.200 --> 04:25:47.589
Dev Panajkar: thousand ideas with no path to impact, or it becomes paralysis, governance so heavy that nothing moves. A system is what prevents both.

1760
04:25:47.940 --> 04:25:54.849
Dev Panajkar: The key message I want you to hold onto is that innovation is not just strategic, it's also tactical.

1761
04:25:54.990 --> 04:25:59.010
Dev Panajkar: Strategic innovation connects to vision and competitive advantage.

1762
04:25:59.220 --> 04:26:02.409
Dev Panajkar: Tactical innovation connects to real workflows.

1763
04:26:02.540 --> 04:26:16.900
Dev Panajkar: A system connects opportunity, alignment, execution, and impact. And innovation must be both strategic and tactical. It has to match the leadership goals and how work happens daily. If that bridge is missing.

1764
04:26:17.180 --> 04:26:25.400
Dev Panajkar: AI stalls, and that bridge is idea management and innovation system. So now, let's look into what that bridge connects.

1765
04:26:27.020 --> 04:26:31.070
Dev Panajkar: What I want to talk to you through today is a simple but powerful idea.

1766
04:26:31.470 --> 04:26:38.909
Dev Panajkar: It's that innovation isn't a department. It's a bridge. And idea management is a system that makes that bridge work.

1767
04:26:39.110 --> 04:26:46.500

Dev Panajkar: This diagram elucidates the integrated relationship between strategy, org planning, technology, and innovation.

1768
04:26:46.640 --> 04:26:53.179
Dev Panajkar: And how these elements must stay in constant motion if an org wants to remain competitive and adaptable.

1769
04:26:53.310 --> 04:26:57.969
Dev Panajkar: At the center, highlighted in yellow, is Idea Management and Innovation.

1770
04:26:58.070 --> 04:27:04.609
Dev Panajkar: And that placement is intentional, because innovation is not an isolated activity happening off to the side.

1771
04:27:04.770 --> 04:27:08.029
Dev Panajkar: It is embedded into the business systems ecosystem.

1772
04:27:08.210 --> 04:27:10.759
Dev Panajkar: This is where discovery meets strategy.

1773
04:27:10.910 --> 04:27:17.969
Dev Panajkar: Where ideas that start out as rough, abstract concepts become the seeds of real transformation.

1774
04:27:18.230 --> 04:27:34.309
Dev Panajkar: So hopefully you understand that these integrated systems are designed for innovation and can truly drive impact only when they are strategic and tactical, ones that align with goals and opportunities, but also support an organization's technical planning.

1775
04:27:34.550 --> 04:27:37.320
Dev Panajkar: Only then we can innovate with AI.

1776
04:27:37.640 --> 04:27:47.920
Dev Panajkar: So what we are going to do next is go one layer deeper, because before we can scale AI innovation, you need to understand the architecture it depends on.

1777
04:27:49.910 --> 04:27:58.030
Dev Panajkar: So, as I said, before we talk about scaling AI innovation, we need to talk something about what most leaders sometimes underestimate.

1778
04:27:58.170 --> 04:28:00.899

Dev Panajkar: AI is not one thing, it's a stack.

1779
04:28:01.170 --> 04:28:07.199
Dev Panajkar: And if you don't understand the stack, you end up funding AI ideas that are impossible to deliver.

1780
04:28:07.340 --> 04:28:13.080
Dev Panajkar: Or you end up underestimating the cost, the risk, and the operational complexity.

1781
04:28:13.190 --> 04:28:15.670
Dev Panajkar: This slide breaks AI into 5 layers.

1782
04:28:16.090 --> 04:28:19.459
Dev Panajkar: Every AI initiative could touch more than one layer.

1783
04:28:20.020 --> 04:28:21.929
Dev Panajkar: Let's walk through them quickly.

1784
04:28:22.160 --> 04:28:23.679
Dev Panajkar: the energy layer.

1785
04:28:24.410 --> 04:28:32.719
Dev Panajkar: This powers the data centers and chips. It becomes a major cost and sustainability factor. Next is the infra layer.

1786
04:28:32.910 --> 04:28:39.089
Dev Panajkar: This is what makes AI scalable and reliable. It connects models, data, and applications.

1787
04:28:39.310 --> 04:28:49.949
Dev Panajkar: The chip layer is what makes AI computationally possible. It executes the heavy lifting, it enables fast training and inference, and directly impacts performance.

1788
04:28:50.230 --> 04:29:02.190
Dev Panajkar: Model layer is a brain. It learns patterns from data, makes predictions, and improves over time. And finally, of course, is the app layer. This is where the user experiences AI.

1789
04:29:02.290 --> 04:29:05.240
Dev Panajkar: It's where AI output becomes action.

1790
04:29:05.730 --> 04:29:10.959
Dev Panajkar: And it determines usability, adoption, and of course, ultimately business value.

1791
04:29:11.120 --> 04:29:20.370
Dev Panajkar: When organizations say we want AI, they're usually thinking about the app layer to the model layer, but enterprise success depends on the entire stack.

1792
04:29:20.620 --> 04:29:23.190
Dev Panajkar: That's why innovation needs a system.

1793
04:29:23.350 --> 04:29:34.069
Dev Panajkar: Because the system forces ideas to be evaluated, not just on excitement, but on feasibility, readiness, and impact across all these five layers.

1794
04:29:34.260 --> 04:29:46.710
Dev Panajkar: And now that we have grounded ourselves in the architecture, you know, we can move to the… into the reality. AI ideas are everywhere. The challenge is what we do with them.

1795
04:29:51.310 --> 04:30:04.930
Dev Panajkar: The good news is that great ideas live with employees, partners, customers, and everyone connected to the business. The best opportunities start with friction and pain points. Where is the work slow? Where are people stuck?

1796
04:30:05.140 --> 04:30:08.979
Dev Panajkar: AI creates value by removing these pain points.

1797
04:30:09.250 --> 04:30:19.759
Dev Panajkar: To achieve this, we must activate crowds by mobilizing the wisdom of the organization. We need to engage communities, increase collaboration, and we then drive results.

1798
04:30:19.960 --> 04:30:24.279
Dev Panajkar: Because the goal isn't more ideas, the goal is business impact.

1799
04:30:24.390 --> 04:30:29.099
Dev Panajkar: The challenge is building the system that turns those ideas into outcomes.

1800
04:30:29.350 --> 04:30:34.380
Dev Panajkar: This slide is the truth that most organizations learn the hard way.

1801
04:30:35.270 --> 04:30:36.839
Dev Panajkar: Ideas are easy.

1802
04:30:36.940 --> 04:30:38.879
Dev Panajkar: Impact comes with a system.

1803
04:30:39.140 --> 04:30:49.599
Dev Panajkar: Because in today's age of Gen AI, everyone has ideas, right? You can ask ChatGPT for 50 AI use cases, and it'll spit it out in a few seconds.

1804
04:30:49.750 --> 04:30:54.829
Dev Panajkar: You can brainstorm, Things in a workshop, you can run a hackathon.

1805
04:30:55.100 --> 04:31:03.820
Dev Panajkar: But none of that guarantees business value. What creates value is the path from idea to impact, and that's what this system does.

1806
04:31:06.930 --> 04:31:09.740
Dev Panajkar: This is a 5-step, eye-to-eye model.

1807
04:31:09.860 --> 04:31:13.880
Dev Panajkar: Idea, investigate, initiate, Implement, and impact.

1808
04:31:14.040 --> 04:31:29.670
Dev Panajkar: In step one, idea, we capture AI ideas across the organization, from employees, customers, partners. In step two, we investigate. So basically, before we commit resources, we validate value, feasibility, and risk.

1809
04:31:29.950 --> 04:31:36.609
Dev Panajkar: In step 3, initiate, we approve, fund, and assign ownership, and decide the execution path.

1810
04:31:36.960 --> 04:31:40.479
Dev Panajkar: We know that if there is no ownership, there is no outcome.

1811
04:31:40.680 --> 04:31:57.219
Dev Panajkar: Step 4 is implement. This is where AI becomes real. We embed it into our workflows, we integrate it into our systems, we manage adoption and change, and Step 5 is the important one, impact. This is where we measure outcomes, learn and scale success.

1812
04:31:57.570 --> 04:32:02.969

Dev Panajkar: And here's a key idea, right? This isn't a one-time journey. This is a repeatable engine.

1813
04:32:03.070 --> 04:32:11.480
Dev Panajkar: Because enterprise AI doesn't win through one big project. It wins through a pipeline of validated, prioritized, and implemented capabilities.

1814
04:32:11.820 --> 04:32:21.520
Dev Panajkar: So the next part of the deck is going to walk through these steps one by one. And we'll start at the beginning. How do we build structure for ideation?

1815
04:32:22.860 --> 04:32:26.500
Dev Panajkar: So let's start with step one, idea intake.

1816
04:32:26.740 --> 04:32:34.150
Dev Panajkar: Because if you want AI innovation at scale, you need to treat ideas like an enterprise asset, not random suggestions.

1817
04:32:34.850 --> 04:32:38.940
Dev Panajkar: And most organizations get this wrong in one of two ways.

1818
04:32:39.040 --> 04:32:57.580
Dev Panajkar: Either they have no intake at all, and AI ideas stay trapped inside individuals or teams, or they create an intake process that is very heavy and bureaucratic, that people stop submitting AI ideas. So what does good intake look like? It's structured, but lightweight.

1819
04:32:57.740 --> 04:33:00.440
Dev Panajkar: This slide shows 4 essentials.

1820
04:33:01.410 --> 04:33:08.259
Dev Panajkar: During the intake process, you need a clear way for everyone and anyone in the organization to submit AI ideas.

1821
04:33:08.490 --> 04:33:14.550
Dev Panajkar: Second, within intake, we should align those AI ideas into the organization strategy.

1822
04:33:14.660 --> 04:33:20.270
Dev Panajkar: This is critical, because you don't want an AI pipeline full of random ideas.

1823
04:33:20.380 --> 04:33:24.109

Dev Panajkar: You want a pipeline that supports the company's priorities.

1824
04:33:24.250 --> 04:33:28.540
Dev Panajkar: Third, we need to collaborate and ensure that ideas are worked on together.

1825
04:33:28.770 --> 04:33:39.019
Dev Panajkar: AI innovation improves when ideas are refined. And fourth, within collaborate are platforms. You need tools that make this easy, not spreadsheets, not email threads.

1826
04:33:39.080 --> 04:33:57.359
Dev Panajkar: A platform creates visibility, transparency, and momentum. Now, here's a key takeaway. Idea intake isn't about collecting volume. It's about creating a clean entry point into a system that can deliver outcomes. Because if step one is messy, everything downstream becomes chaos.

1827
04:33:57.470 --> 04:34:05.900
Dev Panajkar: So… so once we have captured ideas, the next step is the most important gate in the process, investigation.

1828
04:34:07.160 --> 04:34:15.999
Dev Panajkar: This is the most underestimated step in AI innovation… investigation. Most organizations either rush through it, or skip it entirely.

1829
04:34:16.270 --> 04:34:25.940
Dev Panajkar: Investigation isn't about slowing innovation down, it's about making sure we are solving the right problem in the right way before we scale the wrong thing.

1830
04:34:25.950 --> 04:34:43.559
Dev Panajkar: And this is where we validate value. We test feasibility, we surface risk before it becomes expensive, and most importantly, this is where human judgment matters more than algorithms. AI can analyze, AI can simulate, but humans decide

1831
04:34:43.599 --> 04:34:49.039
Dev Panajkar: whether something is worth building. Investigation is the difference between experimentation

1832
04:34:49.150 --> 04:34:56.670
Dev Panajkar: An execution that actually delivers impact. When we get this step right, everything downstream moves faster.

1833
04:34:57.939 --> 04:35:06.499

Dev Panajkar: This slide represents a moment where AI ideas either earn the right to scale, or get stopped early, cheaply, and intentionally.

1834
04:35:06.770 --> 04:35:16.499
Dev Panajkar: On the left is a decision flow. First, we own and scope the idea. Someone is accountable, and we are explicit about what's in and what's out.

1835
04:35:16.800 --> 04:35:28.849
Dev Panajkar: And then we validate assumptions. Not just technical feasibility, but process, impact, risk, cost, and return. And finally, we decide. Move forward, refine, or stop.

1836
04:35:29.099 --> 04:35:32.679
Dev Panajkar: Killing an idea here is not failure, it's leadership.

1837
04:35:33.750 --> 04:35:48.350
Dev Panajkar: In the middle is what I call the innovation sweet spot. A real impact only happens when these four things overlap. Desirability, viability, feasibility, and integrity. Miss any one of these, and scale becomes risk instead of value.

1838
04:35:48.590 --> 04:36:05.029
Dev Panajkar: Let's look at the right. AI gives us powerful validation tools at this stage. Predictive market analysis to forecast demand, agentic workflows to scan competitors and trends, synthetic user simulations to stress test ideas before we spend real money.

1839
04:36:05.110 --> 04:36:11.369
Dev Panajkar: But… but, and this is critical, AI informs decisions, and humans make them happen.

1840
04:36:11.890 --> 04:36:16.379
Dev Panajkar: That's why the real key to success is mentioned at the bottom of this slide.

1841
04:36:16.619 --> 04:36:20.619
Dev Panajkar: Human oversight, critical thinking, context.

1842
04:36:20.880 --> 04:36:28.160
Dev Panajkar: Investigation is where judgment meets speed, so everything that follows can move faster with confidence.

1843
04:36:29.189 --> 04:36:42.219

Dev Panajkar: This slide explains why so many AI initiatives never make it past experimentation. On the left of the funnel, ideas are abundant, and… but as ideas move forward, something important happens. The funnel narrows.

1844
04:36:42.400 --> 04:36:47.220
Dev Panajkar: Not because innovation is failing, but because discipline is being applied.

1845
04:36:47.480 --> 04:36:52.679
Dev Panajkar: Investigation filters ideas based on value, feasibility, and risk.

1846
04:36:53.060 --> 04:37:01.250
Dev Panajkar: But initiate, this highlighted step, is where leadership shows up. This is the moment an idea owns the right to consume real resources.

1847
04:37:01.610 --> 04:37:04.979
Dev Panajkar: Initiation answers 3 non-negotiable questions.

1848
04:37:05.210 --> 04:37:06.790
Dev Panajkar: Are we approving this?

1849
04:37:06.930 --> 04:37:10.450
Dev Panajkar: Are we funding this? And who's accountable for outcomes?

1850
04:37:10.820 --> 04:37:28.759
Dev Panajkar: Until these questions are answered, nothing downstream matters. This step turns possibility into intent. It transforms an idea into a product initiative. It creates focus by saying yes to a few things, and just as importantly, no to many others.

1851
04:37:28.990 --> 04:37:34.079
Dev Panajkar: When initiation is done well, implementation becomes execution, not chaos.

1852
04:37:34.230 --> 04:37:37.619
Dev Panajkar: And impact becomes measurable, not accidental.

1853
04:37:38.950 --> 04:37:48.500
Dev Panajkar: Once an idea has been investigated and approved, the next risk isn't technology. The next risk is starting without readiness.

1854
04:37:48.710 --> 04:37:52.779
Dev Panajkar: And that's why initiation is not a single decision, it's a journey.

1855
04:37:53.000 --> 04:37:58.609
Dev Panajkar: This path shown here shows exactly how a validated idea becomes execution-ready.

1856
04:37:58.880 --> 04:38:09.020
Dev Panajkar: It starts with a product intake, wherein ideas officially enter the organizational product roadmap, and may become part of the next PI planning cycle.

1857
04:38:09.210 --> 04:38:16.359
Dev Panajkar: Next is Engineering Connect. This is where we align tech stack choices and infra constraints.

1858
04:38:16.470 --> 04:38:25.379
Dev Panajkar: Then comes AI FinOps review. This step, by far, is the most critical, because we validate cost, risk, and scalability before we build.

1859
04:38:25.669 --> 04:38:32.210
Dev Panajkar: We do not want to build AI that creates surprise bills or operational drag, which becomes tech debt later.

1860
04:38:32.550 --> 04:38:50.330
Dev Panajkar: After that, we log the solution blueprint. Architecture decisions are made deliberately here. ARB approval ensures governance along with security, compliance, and risks. These are addressed early. And finally, provision. When we reach this point, teams are ready to build.

1861
04:38:52.439 --> 04:38:56.349
Dev Panajkar: This is… this is where most AI initiatives quietly fail.

1862
04:38:56.770 --> 04:39:01.330
Dev Panajkar: Not because the model doesn't work, but because the organization never

1863
04:39:01.599 --> 04:39:09.870
Dev Panajkar: changes how work actually gets done. Implementation is not just about building AI, it's about embedding AI into workflows.

1864
04:39:10.210 --> 04:39:29.969
Dev Panajkar: Right? Implementation is where planning turns into behavior. We finalize priorities, we decide how work will be delivered, we align teams, budgets, and milestones. Then we build, test, and deploy. This is where change management matters most, because if adoption fails, so does impact.

1865

04:39:30.150 --> 04:39:33.849
Dev Panajkar: Successful implementation makes AI invisible.

1866
04:39:35.330 --> 04:39:38.479
Dev Panajkar: This is the moment where strategy leaves this room.

1867
04:39:38.840 --> 04:39:43.809
Dev Panajkar: And execution takes over. Up until now, we have been deciding what to do.

1868
04:39:43.990 --> 04:39:48.199
Dev Panajkar: Implementation is where we decide how work actually happens.

1869
04:39:48.350 --> 04:40:01.599
Dev Panajkar: On the left, you see the classic SDLC, right? Requirements, design, build, test, maintain. But AI changes the nature of delivery. Models evolve, data changes, feedback never stops.

1870
04:40:01.700 --> 04:40:09.530
Dev Panajkar: That's why most successful AI implementations shift from an agile, you know, towards an agile and iterative model.

1871
04:40:09.650 --> 04:40:10.989
Dev Panajkar: Shown on the right.

1872
04:40:11.120 --> 04:40:16.810
Dev Panajkar: Instead of building once and hoping we get it right, we learn continuously. We develop

1873
04:40:16.980 --> 04:40:21.330
Dev Panajkar: Test, implement, and review. And we improve, again and again.

1874
04:40:21.550 --> 04:40:27.700
Dev Panajkar: Implementation isn't complete when the system launches. It's complete when people adopt it.

1875
04:40:28.650 --> 04:40:34.669
Dev Panajkar: That's why this phase explicitly includes change management, training, and transition.

1876
04:40:34.850 --> 04:40:41.889
Dev Panajkar: Implementation succeeds when AI stops feeling like a project and starts feeling like part of the job.

1877
04:40:45.020 --> 04:40:45.840
Dev Panajkar: Right?

1878
04:40:46.510 --> 04:40:58.959
Dev Panajkar: This is where AI finally earns the place in the business at impact. Impact is not activity, it's measurable change, with examples like revenue increases, or cost reduces.

1879
04:40:58.960 --> 04:41:06.880
Dev Panajkar: Or decisions are improved, or risk lowered, and so on. If we can't point to a real outcome, we didn't innovate, we just experimented.

1880
04:41:07.070 --> 04:41:15.579
Dev Panajkar: At this stage, leaders ask difficult questions. Did this actually move the business? Can we repeat it? Can we scale it responsibly?

1881
04:41:15.980 --> 04:41:23.360
Dev Panajkar: When teams see real results, trust grows, adoption increases, better ideas surface upstream.

1882
04:41:23.480 --> 04:41:32.799
Dev Panajkar: The system feeds itself. AI innovation isn't about more ideas, it's about turning the right ones into impact, and doing it again and again.

1883
04:41:34.240 --> 04:41:51.089
Dev Panajkar: This is where AI stops becoming a project and starts becoming a capability. On this slide is what I call the AI realization matrix. It's not a checklist, but a lens. It's a way for leaders to evaluate readiness, assess impact, and decide where to invest next.

1884
04:41:51.470 --> 04:42:10.139
Dev Panajkar: On the right, you see at the center is the Gen AI value realization, and it sits between two non-negotiables, a responsible AI approach built on trust, governance, and intentional design, and a human-led approach, where AI amplifies human judgment instead of replacing it.

1885
04:42:10.530 --> 04:42:21.609
Dev Panajkar: Impact unfolds in stages. We start with, of course, a clear value hypothesis, we define use cases that matter, we apply proven patterns, select the right tooling.

1886
04:42:21.860 --> 04:42:37.850

Dev Panajkar: We move into solutioning, evaluate costs, deploy and learn in the real world, and then scale adjacent value responsibly. This is how AI stops being a one-off success and becomes a repeatable engine of value.

1887
04:42:38.440 --> 04:42:47.849
Dev Panajkar: When AI is… when impact is measured this way, leaders stop asking, did AI work? And instead, they start asking, where do we apply it next?

1888
04:42:49.920 --> 04:42:55.920
Dev Panajkar: Innovation without measurement isn't innovation, it's just an activity. This slide explains

1889
04:42:56.440 --> 04:42:59.810
Dev Panajkar: Answers a simple yet uncomfortable question.

1890
04:43:00.260 --> 04:43:14.870
Dev Panajkar: How do we know innovation is actually working? On the left are signals that tell us whether our innovation system is healthy, flow health, are ideas moving, or are they stuck? Are we seeing volume, acceptance, and velocity?

1891
04:43:15.080 --> 04:43:26.599
Dev Panajkar: what's the portfolio quality? Are we doing… are we only doing safe incremental work, or are we placing intentional bets that can change the game? What are the ROI trends?

1892
04:43:27.110 --> 04:43:34.629
Dev Panajkar: Not just cost, but realized value over time. These metrics don't punish teams, they reveal where friction lies.

1893
04:43:35.010 --> 04:43:36.699
Dev Panajkar: On the right, you see…

1894
04:43:37.060 --> 04:43:53.789
Dev Panajkar: you know, AI successes that can't be measured only on one dimension alone. We look at, you know, there are four of them that we look at together. Does the AI actually work? Does it become outcomes that leaders care about? Do people rely on it or work around it?

1895
04:43:53.920 --> 04:44:09.199
Dev Panajkar: Can we run it sustainably, securely, and at scale? When any one of these is missing, impact is fragile. When all four are present, AI becomes dependable. Metrics don't slow innovation, they give it direction.

1896
04:44:10.480 --> 04:44:15.170

Dev Panajkar: This slide makes a lot of people uncomfortable, and that's intentional.

1897
04:44:15.380 --> 04:44:20.410
Dev Panajkar: AI experiments are easy to start, but AI impact is hard to sustain.

1898
04:44:20.580 --> 04:44:25.920
Dev Panajkar: And confusing the two is the fastest way to burn money, credibility, and trust.

1899
04:44:26.290 --> 04:44:35.720
Dev Panajkar: Before an AI initiative earns the right to scale, leaders need to ask… need to be able to ask and answer 6 very simple questions shown here.

1900
04:44:36.630 --> 04:44:44.299
Dev Panajkar: If you can't answer these questions, you are not failing. You are just not ready. And readiness is something you can build intentionally.

1901
04:44:45.800 --> 04:44:54.769
Dev Panajkar: Well, we have talked about systems, we have talked about discipline, we have talked about how ideas move from inspiration to execution, and it all comes down to this.

1902
04:44:55.180 --> 04:45:09.499
Dev Panajkar: AI innovation is not about more ideas. Ideas are everywhere. What's rare is follow-through. It's about choosing the right ones and building the courage, the structure, and the leadership to turn them into impact.

1903
04:45:09.840 --> 04:45:19.049
Dev Panajkar: The work… this work isn't about technology. It's about how organizations decide, commit, and act, again and again.

1904
04:45:19.200 --> 04:45:22.840
Dev Panajkar: Because when ideas are seen, shaped, and scaled.

1905
04:45:23.030 --> 04:45:29.790
Dev Panajkar: That doesn't change… that just doesn't change businesses, they change what people believe in. Thank you.

1906
04:45:30.650 --> 04:45:33.759
David Gijsbers: Ev, thank you very much, that was awesome.

1907
04:45:33.880 --> 04:45:38.130
David Gijsbers: We do have some, questions in the chat,

1908
04:45:38.320 --> 04:45:43.479
David Gijsbers: One, first one is, can you elaborate a little bit more on the AI metrics?

1909
04:45:43.770 --> 04:45:49.310
David Gijsbers: This question's related to the innovation metrics, the slide on innovation metrics.

1910
04:45:49.310 --> 04:45:57.150
Dev Panajkar: Okay, this one. Agreed. You know, this is a… this is a good slide. I love this slide because on the left, it talks about

1911
04:45:57.200 --> 04:46:11.389
Dev Panajkar: portfolio-level metrics, or, you know, you know, things that are… that talk about innovation. And on the right, these speak a bit more about AI-specific, you know, metrics, right? And I truly, you know.

1912
04:46:11.390 --> 04:46:20.939
Dev Panajkar: like these, I didn't specify all the detailed ones, you know, across the four ones. For example, for AI model performance.

1913
04:46:20.940 --> 04:46:33.449
Dev Panajkar: People talk about accuracy, F1 score, blue or rogue, inference latency, response time, and things like that. For business impact, of course, we talk about ROI, cost savings.

1914
04:46:33.460 --> 04:46:46.219
Dev Panajkar: Efficiency gains, revenue growths, time to value. With respect to adoption and trust, there's always… we talk about our adoption rates, usage frequency, session length.

1915
04:46:46.320 --> 04:46:59.609
Dev Panajkar: you know, CSAT, you know, with respect to user feedback and so on. Operational length, operational health and data metrics, we talk about data quality or drift.

1916
04:46:59.670 --> 04:47:18.360
Dev Panajkar: model drift, and so on. So, you know, there are much, many more that people use, but I focus on these when I speak to, you know, companies to implement AI holistically, and also for internal consumption, we use a lot of metrics to ensure that we are doing things correctly.

1917
04:47:19.760 --> 04:47:26.240

David Gijsbers: Dad, we've got a request to repeat the, quote, metrics do not slow innovation, it gives it

1918
04:47:27.610 --> 04:47:29.720
David Gijsbers: direction, clarity…

1919
04:47:31.230 --> 04:47:35.319
Dev Panajkar: Exactly, yes. I will look for that quote and post it down here.

1920
04:47:35.540 --> 04:47:39.890
David Gijsbers: Alright, awesome. Alright, second… second question is,

1921
04:47:40.680 --> 04:47:44.600
David Gijsbers: Can you give an example of FinOps? How to use it effectively?

1922
04:47:45.210 --> 04:47:57.159
Dev Panajkar: Oh my god, this is an amazing topic, thank you for that question. You know, for me, right, I'm a firm, firm believer that all organizations or companies should integrate FinOps

1923
04:47:57.160 --> 04:48:05.249
Dev Panajkar: AI FinOps early into their eye-to-eye model. As explained on the initiate slide, let me see where the initiate…

1924
04:48:05.260 --> 04:48:06.859
Dev Panajkar: Slide is.

1925
04:48:08.820 --> 04:48:12.849
Dev Panajkar: You know, this one here, right? AF In Ops Review.

1926
04:48:12.850 --> 04:48:26.849
Dev Panajkar: You know, let me give an example and do a bit deeper with an example. We usually, you know, this is akin to, you know, when we used, when we moved to cloud, from our on-prem environments.

1927
04:48:26.850 --> 04:48:35.700
Dev Panajkar: We usually often default to cloud APIs for AI workloads, because they're easy, right? But easy can sometimes be expensive quickly.

1928
04:48:35.700 --> 04:48:55.159
Dev Panajkar: You know, just a few weeks ago, I was working with a colleague on an early AI project that incorporated audio translations and transcribing, and we were

using Google Speech-to-text. And it usually costs around 2 cents or 1.6 cents per minute as a cloud cost.

1929
04:48:55.290 --> 04:49:03.850
Dev Panajkar: The project that we're working on would have processed thousands of hours of audio. And can you imagine the variable cost?

1930
04:49:03.970 --> 04:49:14.620
Dev Panajkar: for this, that would have destroyed the project margins completely. And this is where an early assessment of AI FinOps, the, you know, trajectory costs are extremely important.

1931
04:49:14.640 --> 04:49:32.999
Dev Panajkar: Because in this instance, we worked on the infra and decided to use a hybrid approach, and we leveraged the on-prem compute power instead of purely cloud power, right? So, it's extremely important to focus on AI FinOps costs.

1932
04:49:33.140 --> 04:49:48.000
Dev Panajkar: The point I'm driving at is that at the early onset of the project, you experiment and pay for speed, right? Speed and convenience using cloud APIs and infra, but as we scale, we have to audit our costs.

1933
04:49:48.000 --> 04:49:55.840
Dev Panajkar: Because any compute costs or any GPU costs can have a drastic impact on the overall project cost.

1934
04:49:55.850 --> 04:50:01.569
Dev Panajkar: You have to leverage the hybrid model, you have to leverage price to, price-to-performance metrics.

1935
04:50:01.660 --> 04:50:17.739
Dev Panajkar: In short, by moving inference to the edge, we were able to cap our cost while, you know, maintaining burst capacity via the cloud failover. So, you know, that's a perfect example I can give. It's easy to get carried away by

1936
04:50:17.760 --> 04:50:25.339
Dev Panajkar: moving everything to AI and on the cloud, and it can have a drastic negative impact on costs.

1937
04:50:30.060 --> 04:50:33.509
David Gijsbers: Sterling, would you like to come off mute and ask your question?

1938
04:50:36.920 --> 04:50:43.530
Sterling Fulton: Yes, yeah, the question that I had was about capturing,

1939
04:50:43.660 --> 04:50:48.559
Sterling Fulton: Capturing the time that's saved for AI investment.

1940
04:50:48.560 --> 04:51:04.410
Sterling Fulton: What happens, or what type of ROI could you talk about so that, when a company is looking, for AI, or to, or to, how can I say this? When a company is looking

1941
04:51:04.410 --> 04:51:15.950
Sterling Fulton: to, utilize AI, there are savings, time savings, that that company's going to incur. Is that something that you're taking into account?

1942
04:51:17.130 --> 04:51:26.929
Dev Panajkar: Oh, absolutely. There are various facets, of metrics, KPIs, performance metrics that we need to look into.

1943
04:51:27.100 --> 04:51:38.250
Dev Panajkar: And, you know, there are technical, you know, metrics, and there are non-technical and business-oriented metrics that I mentioned we need to look into. And of course, you know, a time save.

1944
04:51:38.250 --> 04:51:57.030
Dev Panajkar: saving is a huge thing. Early, a lot of companies moved resources from on-site to offshore, or near shore, and so on. This is the same thing with respect to, you know, AI agents. You know, a lot of companies sometimes focus even on GPU time. You know, they're able to,

1945
04:51:57.030 --> 04:52:14.730
Dev Panajkar: make deals with, you know, GPU vendor providers to get a cheaper rate for an off-time, you know, compute execution. So there are various factors you can look into, and yeah, you know, that particular example is a sure one.

1946
04:52:17.560 --> 04:52:25.269
David Gijsbers: So, what, what are you hearing about organizational resistance?

1947
04:52:27.110 --> 04:52:42.180
Dev Panajkar: I think one of the most important org resistance, you know, I can give a very personal example. My, you know, I've got a son and a daughter. My son is a commci major, focusing on AI, and my daughter is humanities major, environment sciences.

1948
04:52:42.180 --> 04:52:56.740

Dev Panajkar: She absolutely thinks AI is a wrong approach, because she says people, you know, kids, students in colleges are using it, you know, to write papers and not spending true time, you know, thinking and so on.

1949
04:52:56.890 --> 04:53:06.710
Dev Panajkar: So, a lot of times, this particular example is, of course, very important from organizational planning as well.

1950
04:53:06.710 --> 04:53:31.080
Dev Panajkar: so long as we are able to be open with respect to culture, saying, you know, it's absolutely okay to experiment. You know, some people, you know, may say that, hey, we want to experiment on a particular LLM or a different LLM. It's absolutely okay. Earlier, we had only a limited number of LLMs, but nowadays, with agents and tools, specific tools for specific problems.

1951
04:53:31.190 --> 04:53:50.909
Dev Panajkar: You know, there's a plethora of things that we can experiment, and letting employees and team members experiment is the best thing to start with. Only thing is, of course, as Priyanka rightly mentioned, implementing it into production, you need to have guardrails.

1952
04:53:50.910 --> 04:54:02.720
Dev Panajkar: governance guardrails, FinOps guardrails, and so on, because if a wrong thing happens, it can truly have a bottom line disaster, or even a PR disaster for the company.

1953
04:54:04.230 --> 04:54:10.420
David Gijsbers: What about the cost of tokens for AI-native software engineers? Like…

1954
04:54:10.760 --> 04:54:15.009
David Gijsbers: How would… how would companies even start budgeting for something like that?

1955
04:54:15.440 --> 04:54:30.790
Dev Panajkar: No, absolutely correct. You know, earlier, if you looked at late 2025 and early 2026, if you look at, board, you know, street, discussions.

1956
04:54:30.870 --> 04:54:48.840
Dev Panajkar: there are… every line item for the company has to have an AI, you know, word embedded in there, because the street wants to know how much… how much are you investing on AI, and what is the, you know, specific initiatives you are focusing on. So,

1957
04:54:48.840 --> 04:55:13.019

Dev Panajkar: it is imperative for companies to, you know, carve out budgets, both training, organizational budgets, experimentation budgets, and then determine, as I mentioned in my presentation, you need to have specific use cases to be identified. Those are pain points that you want to address, and have, you know, internally

1958
04:55:13.020 --> 04:55:15.559
Dev Panajkar: internal teams to build AI agents.

1959
04:55:15.560 --> 04:55:32.279
Dev Panajkar: And of course, at the end, go, you know, to Agentic AI to solve those pain points. And those are the use cases, business cases, that, you know, you have to take small wins, and then, you know, have them part of your project management roadmap.

1960
04:55:35.010 --> 04:55:44.289
David Gijsbers: Alright, this is the last call. It's been… it's been quite an afternoon. Anybody got any more questions, either in the chat.

1961
04:55:45.260 --> 04:55:50.059
David Gijsbers: David Vidra expects to spend $500K per year in tokens, because.

1962
04:55:50.060 --> 04:55:52.610
David Mantica: Yeah, David, you gotta really… yeah, he got a really big.

1963
04:55:52.610 --> 04:55:54.220
David Gijsbers: I mean, that's the…

1964
04:55:54.220 --> 04:55:56.600
David Mantica: That's actually the honest truth, I mean…

1965
04:55:56.700 --> 04:56:01.759
David Mantica: This is an area that people don't dig into enough. It depends on the vendor you pick.

1966
04:56:02.090 --> 04:56:15.529
David Mantica: And it depends on their strategy and their licensing costs, but you're gonna have to dig into your service level agreement, look at the token costs that they're doing, and the tokens they're giving you as part of your licensing fee. But that's not out of the realm

1967
04:56:15.880 --> 04:56:17.240
David Mantica: of possibilities.

1968

04:56:19.640 --> 04:56:20.589
Dev Panajkar: I'm just going through…

1969
04:56:20.590 --> 04:56:26.039
David Mantica: This will be 20… 2027 will be the year people snap about token fees.

1970
04:56:27.170 --> 04:56:41.249
Dev Panajkar: No, good point. And David, I'm going through the chat here, and, you know, David Mantica, you have mentioned a couple of good points there. It's absolutely correct that it's mostly the fear mindset.

1971
04:56:41.250 --> 04:56:50.380
Dev Panajkar: That we need to work on. And every organization is going through its own, its own level of,

1972
04:56:50.570 --> 04:57:14.279
Dev Panajkar: adoption, right? Every organization, every department within every company goes through it differently. You know, a lot of times, you know, digital organizations like e-comm and so on are easy to adapt, you know, versus supply chain and so on, sometimes, you know, tend to lag, or ERP organizations or domains, you know, tend to lag.

1973
04:57:14.280 --> 04:57:16.470
Dev Panajkar: in AI implementation adoption.

1974
04:57:16.960 --> 04:57:21.170
David Mantica: I got to do a number of executive AI sessions

1975
04:57:21.470 --> 04:57:35.669
David Mantica: A little bit this year and a little bit last year, and holy moly. Number one, it was all fear. It was all very… I only went… I only did one session where the CEO was more of an aggressive, alright, let's see what we can do. Most of it was protection.

1976
04:57:35.770 --> 04:57:37.350
David Mantica: Fear, control.

1977
04:57:41.130 --> 04:57:44.120
David Mantica: Pretty cool. And nowadays… You got some.

1978
04:57:44.120 --> 04:57:44.560
Dev Panajkar: It doesn't end up.

1979

04:57:44.560 --> 04:57:46.260
David Mantica: Yes, go ahead, you first.

1980
04:57:46.920 --> 04:58:05.940
Dev Panajkar: sorry, you know, because I'm a board member as well, a lot of boards are focusing on, AI knowledge for board members as well. It's become mandatory for, you know, governance. There are, you know, some companies have, have, have, started internal committees. That's great.

1981
04:58:05.940 --> 04:58:21.689
Dev Panajkar: on AI as well, because you, you know, earlier it used to be cloud, now it's AI, but that… that awareness is very important, and I think it coming from the top, always sends a good message down the line that the company's serious about AI, and so on.

1982
04:58:22.370 --> 04:58:25.649
David Mantica: There's, like, 2 or 3 slides in your deck that could really become

1983
04:58:25.980 --> 04:58:30.799
David Mantica: You know, stuff that gets viral, depending on how you position it.

1984
04:58:30.940 --> 04:58:34.799
David Mantica: Especially, I think it was the second-to-last one, the last one, I was like, yeah, that's…

1985
04:58:35.080 --> 04:58:39.330
David Mantica: something I see a lot, just because… because we work with a ton of customers. I mean, we've been…

1986
04:58:39.440 --> 04:58:43.959
David Mantica: We started doing conferences in, in, 2023.

1987
04:58:44.150 --> 04:58:49.960
David Mantica: And, you know, that doesn't seem like it's very long, but we probably trained about 200,000 people now, and…

1988
04:58:50.280 --> 04:58:59.879
David Mantica: Those couple slides there, man, really mapped up to some of the stuff that we're seeing, and it gives people a great mindset of how to approach it at the transformation level.

1989
04:59:00.850 --> 04:59:06.550
David Gijsbers: So, yeah, the intention is, oh, Lars or handsome, to share the deck.

1990
04:59:06.720 --> 04:59:16.789
David Gijsbers: So, just as I kind of wrap it up here, I just wanted to say thanks to Lara and Dave Manteca. When they first suggested this, I was like.

1991
04:59:16.860 --> 04:59:29.530
David Gijsbers: Yeah, let's do it. Let's see who, you know, who can, contribute. I really like the idea of bringing it all the way from, from the strategic level with, you know, dev and

1992
04:59:29.610 --> 04:59:43.279
David Gijsbers: and Steve Elliott's presentations down into the code level with Ken Pugh. So, appreciate everybody. You know, this is the first one that we did on AI in the enterprise, but it's definitely not going to be the last one.

1993
04:59:43.340 --> 04:59:51.920
David Gijsbers: If you have presentations that you'd like to share with the community, please let me know, and, you know, we can take a look and see how we,

1994
04:59:52.050 --> 04:59:54.510
David Gijsbers: Maybe costs are wider than that.

1995
04:59:54.730 --> 05:00:04.040
David Gijsbers: For, for the next, agenda. But I'd like to just personally thank everyone for their participation. I would like to personally thank

1996
05:00:04.190 --> 05:00:12.380
David Gijsbers: all of the speakers, and all of our sponsors, they made it happen. So thanks, everybody, and, have a good evening.

1997
05:00:14.520 --> 05:00:15.770
David Mantica: Thank you all!

1998
05:00:16.730 --> 05:00:17.890
David Mantica: Thank you.

1999
05:00:17.890 --> 05:00:19.530
Sasan Afsoosi: Thank you, G, thank you.

2000
05:00:19.530 --> 05:00:20.990
Lara Hill - SoftEd: Thanks, everyone!

2001
05:00:20.990 --> 05:00:22.730

```
Toby Rao: Can't wait for the next one!

2002
05:00:22.900 --> 05:00:26.700
Dev Panajkar: It's coming! Bye-bye.

2003
05:00:26.700 --> 05:00:27.480
David Mantica: Bye-bye.
```