EPISODE 1431

[INTRODUCTION]

[00:00:00] JM: National Instruments develop software and hardware for engineering in a wide variety of domains, from aerospace to government technology to application testing. The interface between hardware and software presents a variety of difficult engineering challenges. Luke Schreier is a senior vice president in National Instruments and joins the show to discuss the engineering and management of the company.

[INTERVIEW]

[00:00:21] JM: Luke, welcome to the show.

[00:00:21] LS: Yeah, thanks, Jeff. It's an honor to be here. Thanks for having me.

[00:00:26] JM: You work at NI, formerly National Instruments, and you work on mostly the aerospace, defense and government side of things. And I guess I'd like to start by just discussing the primary product lines of NI and what you guys build for the average customer.

[00:00:49] LS: Yeah, so if your listeners aren't familiar with NI, NI has been around for about 45 years, largely a test and measurement company, but we do some work in design as well. So, what that means is we'll help customers kind of design test and validate, average everyday devices that you're encountering, whether that's a mobile phone, or an electric vehicle, or some of the focus you described for me is more on the aerospace and defense side. So, if there's new commercial aircraft, or if there's defense systems, we're going to make the hardware and software that are going to help our customers be able to make sure that those devices were built correctly.

So, that might mean validating that the design was the right design in the first place. That might mean making a measurement on the product as they roll off the assembly line. Or that might mean making measurements on stuff that's deployed in the field to make sure that it's still working right. The specific products, we'll make instruments like if there's some electronics heads in the audience, some multimeters, or oscilloscopes, or basic voltage current measuring devices, but will also make application software that allows you to program those devices. NI LabVIEW is probably one of our more well-known pieces of software that allows people to do graphical programming and design those systems. But then there's also, you know, test management software or embedded software validation suites. So, it can range from very simple measurements on simple devices all the way to really complex systems, test some of the most complex devices ever built.

[00:02:19] JM: Describe the process of developing kind of an integrated hardware and software product line that allows this kind of testing infrastructure.

[00:02:32] LS: Yeah, well, we usually work back of course, from what our customers are wanting to test. So, if I'm just thinking about something as complicated as a new fighter jet, some navy or Air Force might be building, there's going to be subsystems there for electronics, or communications, or there's going to be the physical skin of the jet. There's going to be the engine. There's going to be all these different things that need to be validated, they're going to work correctly. So, we'll take a look at that. And we have over decades set, right, what are all the different things you want to measure there? What are the analog signals that you're going to want to digitize, and you're going to want to be able to do some analysis on inside some kind of computing platform?

So, that leads you to building measurement devices of different frequencies, or of different channel accounts that can be synchronized in certain ways, and combined with other measurement types to sort of be able to paint the full picture of the system. We do that in different hardware platforms. One that's open standard called PXI, it's PCI extensions, or PCI Express extensions for instrumentation. But there's lots of other form factors like that, that we will make card modular instruments plug into. That's kind of on the hardware side.

And then on the software side, we'll take a look and say, “Well, how do you want to aggregate all that data? What are the kind of driver interfaces to those different instrument technologies? What's the operator going to be doing? Is it an engineer? Is it a technician? Is it something that's going to be in a lab? Or is it going to be deployed?” Start to architect kind of the software stack that's going to talk to those instruments. And our approach kind of from the beginning has been to say, let's make tools and capabilities that allow our customers to build as much of that themselves. We will provide that kind of technology building blocks, but we don't do a lot of the turnkey, here's your test system, call us if you need help. We try to provide a really open hardware and software stack that our customers can piece together just the right mix of capabilities to be able to test to whatever extent they want to.

But if you think about what that requires, when the fighter jet example I brought up, versus a mobile device or consumer electronics equipment, it could be very different, radically different mix of technologies involved there. Our goal as a company trying to make a good business out of this is to find all the commonalities there, and then deploy those as efficiently as possible to our customers.

[00:04:52] JM: There is an inherent challenge in testing a testing instrument. How do you validate that something is actually giving the correct reading, if it is itself the piece of testing infrastructure?

[00:05:09] LS: Yeah, it's an insightful question. Because if you're going to rely on the measurement device to be able to validate that something works, you've got to really rely on the way you designed the measurement device. And so, we'll certainly have rules of thumb about the amount of uncertainty, the absolute accuracy, the way that we're going to validate the equipment that we deliver. But frankly, there's an inherent challenge to trying to make your systems open and interoperable extensible by the customers, we can specify things to a certain degree, but we want to make them as software defined as possible. We want to make it so that our customers can derive even more value out of them. And sometimes, that modification can change some of the performance characteristics.

So, a lot of times, we have to be really clear on, here's what I was tested to when we designed it. And here's if you want to use it just in that way, this is what will guarantee and you can send it back for calibration, you can do all the things that you would need to when you think about hardware maintenance. But we also want to give the users the ability to tweak elements of that, and maybe, in some cases, change the firmware, reprogram the firmware on some of our instruments to put in different signal processing. And then we have to be really comfortable in the way we've architected all those systems, so that they know what they can rely on for us and what sort of calibratable, but then also take ownership for the additional work they put in to make it their own.

And of course, in my customer base, where it's a lot of times, you know, highly sensitive data or classified uses of the program, that's kind of service level agreement works really well. Our customers understand what they need to manage if they're going to go in and start tweaking. But on the other side, we sell to a lot of hobbyists and casual users that probably don't want to do that. And they just want to know, “Hey, what does the spec sheet say that this thing will do?” And we've tested it multiple orders of magnitude, more accuracy than that, to confirm that that's what it's going to do repeatedly when it rolls off our assembly lines.

[00:07:06] JM: What's the interface between LabVIEW and the measurement tools themselves?

[00:07:16] LS: So, what we design, whenever we build a piece of hardware, we will write a driver for it. We’ll explain how the communication is going to go across whatever bus, whatever computer bus, or interface it's connected to, we're going to write a driver that makes that possible. That driver will have programming interfaces for every kind of programming language you can imagine. If you take it for LabVIEW, for instance, we're going to provide a set of the function calls that are going to be most commonly used. We'll do example programs to help you get up and running quickly. We'll probably also do interactive panels to be able to query the instrument or be able to use it in an interactive way. But that's not just limited to LabVIEW. That would be for Python, or that would be in any kind of programming environment that a customer would want to use. And that's been one of our kind of core principles is to try to make it as open and interoperable, whether someone wants to use our software, which is great. Or if they have a specific language they're comfortable in, they can use that as well.

[00:08:11] JM: Can you take me through the development of a particular hardware module that you've worked on and just describe the end to end process of developing that system?

[00:08:29] LS: Yeah, so if I'll take one of the topics of interest lately, take the software defined radio, we have this device called the Universal Software Radio Peripheral, that is a receiver and a transmitter that's connected up over a bus, either through PCI Express or can be over

Ethernet to be able to connect it to the computing platform. We'll look at that at the beginning and say, “Alright, what are the frequency ranges? What's the dynamic range that we need to be able to measure signals at?” Obviously, it's a transmit and it's a receive. So, what ultimately are we going to try to enable the customer to digitize? So, how much of a custom analog front end are we going to need? Or in the case of most software defined radios, how light can we make that front end? And how much can we put into the firmware. We'll put one of Xilinx RF systems on a chip on there, which has FPGA computing platform but also arm kind of processing. And it'll have analog to digital converters, digital to analog converters for the transmit and the receive side. And then we'll either use the kind of native bus interface that's a part of that, or we'll use one of our own to be able to communicate that back over Ethernet or over PCI Express going to the host, to the PC that's ultimately going to receive the data.

We’ll, of course, have to do a whole bunch of mechanical design, what's the form factor you want that thing to sit in? Is it going to sit on someone's desk? Are we going to deploy it in some size, weight and power optimized environment? How sensitive do we need to make each piece of the signal processing chain there? How much are we going to rely on the hardware to do versus the software to do? So, how much do we need to ramp up large firmware development efforts to? And in the case of software defined radios, that's kind of where the trend is. Let's try to do as much as we can, in software instead of kind of hard, electronic-zing things. So, we leave ourselves more flexibility.

And then it's traditional electronics design, using the design tools. It's manufacturability assessments, and it’s a lot of integration testing. When we bring all those things together, are we getting the analog performance that we want on the transmit and receive chains? Can we put real world signals through this and make sure that it's going to meet the requirements? And then also, can it, in a lot of cases, stream the data off to another location? Can it be synchronized with other devices, and that sort of gets into what the end application of this building block of this software defined radio going to be? Because a lot of times they can get to be really complex.

So, we do, not just the product design, but we're also designing for the system uses of it, and trying to make that as easy to use or user friendly in the way that some of that capability scales. But then also, we got to get these things out on time. We've got to hit price targets. So, there's a lot of that assessment and reassessment that's going on. And then increasingly too, we want to sustain these things for years or again, in the case of my customers, sometimes decades, so we think about what are the key points of lifecycle management that we need to build in from the beginning. And I think that's really where the opportunities to make things more software defined, have been extremely important to our customers across all industries, because they'd like to be able to put as much in software as they can, and make the hardware design as lightweight as possible, shortens our development cycles, makes it easier for them to maintain, ultimately makes it easier to afford. So, that's a lot of the calculus that's going into that design process right now.

[00:11:58] JM: The amount of engineers and product designers involved in this must be pretty considerable. Can you give a sense for how engineering and team management works on a project like this?

[00:12:14] LS: Yeah, so much like probably every other part of the modern engineering world, we're setting up as many agile processes and teams with right subject matter expertise. And we're setting up as much architecture reuse as we can. I talked about bus interfacing earlier. As a company, we don't need to have 15 different ways of moving data over PCI Express over Ethernet. So, there's dome centers of excellence that we pull from across all the different modules were developing and platforms we're supporting. And pulling those kinds of modular building blocks into a design, we'll certainly do that as we work with silicon vendors on analog to digital, and digital to analog converter technologies. Obviously tight alignment with their roadmaps and what they're building. So, what we can deploy and what new challenges it'll solve.

So, it's a big efficiency exercise to be able to do and the goal for us would be to get a lot of reuse, but also, to present it in a way to our customers that want to follow the technology trends as well, that they're not having to learn something brand new, every time there's new capability from NI. They understand how it works. They understand, maybe they've written their application in a certain way, and will attempt to drop in new capability that preserves the code that they've written before, by leveraging the same API's. That just ends up being this huge, huge deal for not just our aerospace and defense customers, although for them literally, changing code is sometimes an act of Congress. But also, as engineering teams that our customer locations are getting tougher to recruit for. And the way design is happening across distributed supply chains is making collaboration, all the more important. We feel like it's really important that we're taking the extra steps in the design and development of all these tools that makes our customers as efficient as they can be.

Ultimately, we want to take the test and measurement process out of the like painstaking critical path that it sometimes becomes in the design of a new electric vehicle, for instance. And instead make it this opportunity to really extract more insight and more data, more value out of the the vehicles that are being developed and ultimately make them even better products as they continue to evolve. But it's honest to not make that harder than it has to be. And so, we try to do that with elegant, elegant design techniques.

[00:14:34] JM: Since there's such a wide variety of use cases for LabVIEW and the hardware products, I'd like to go into a specific domain that you work in, which is aerospace, and just get a sense for a customer prototype and or prototypical customer and maybe just go in a little bit deeper into what a day in life looks like and what they're using LabVIEW for as well as whatever other hardware products they'd be using.

[00:15:08] LS: Yeah, so a customer that's been in the news lately, based on their success, which is always a good reason to be in the news that we've worked with a lot is Virgin Orbit. So, they're building different type of satellite launch vehicle that actually flies from under the wing of a 747. So, it's a differentiated solution for them, they can launch from anywhere in the world, wherever a 747 can fly, they can put this rocket under the wing, and then be able to very affordably put small satellites into orbit.

So, when they go about that engineering challenge of how we're going to develop this rocket, you have a bunch of different domains and subsystems you got to tackle there. One that we worked really closely with them on is the avionics side. How is this rocket going to guide itself? How is it going to fly? And the way that's being done lately is through different variations of concepts of kind of digital twin architectures where you're going to want to build as much as you can, or certainly you're going to simulate maybe the engine control concepts, or the avionics control concepts in software. You're going to get a good sense for what those things look like. They may be using our tools to do that. They may be using other tools.

But at some point, you want to start applying real world data to it, way before you're launching it off the underlying of a 747. So, you'll build the flight electronics, and you'll maybe even lay out all of the cabling to all of the sensors that are going to be on this rocket. You're going to put that all out on a table with real actuators, and again, with real links of how the cable will look. And you're going to start exposing that equipment, that full avionics subsystem to different stimuli. It would see when it was flying. And the goal there is to be able to simulate not just a launch, but maybe hundreds or thousands, or hundreds of thousands of scenarios that that rocket could encounter when it flies. And so that's done, they'll write individual code modules for what the sensor simulation needs to look like or what the what that engine control unit supposed to do to move a flap or something to control the rocket. And then they'll run through a sequence of those simulations, and maybe they'll expose all the sensor inputs with electrical stimuli that would mimic what the different aerodynamic conditions would present to it.

They'll run this over and over again. And then when they actually build the rockets, they'll continue to have those systems up and running. Because sometimes when the rocket is flying, they might encounter a different circumstance, want to understand, “Hey, what if we need to run some scenarios in flight, they have that test bench to be able to do that.” That's just one subsystem. They would do similar kind of design, prototyping, validation and deployment, for again, the engine system, the telemetry, other comms, the structure of the of the rocket, and they'll use LabVIEW, a lot of times to write the code to do that. They'll use maybe another software protocol of ours called VeriStand, walk through the different scenarios, the different conditions that all those sensor inputs are going to receive. They may use another software package called TestStand to be the test management environment that's going to sequence how a bunch of different parametric measurements are occurring. And they'll be using our hardware kind of scattered either to simulate the sensor inputs, or to provide other kinds of stimuli that ultimately the aerodynamic conditions might replace when it's actually deployed.

Because generally, our stuff doesn't get deployed. It's more of the test equipment, but we wanted to have as much fidelity to the real world as we can, and that's usually just higher dynamic range measurements, more accurate sensor simulations. And so, the better we make our products, the more they can be programmed to be exactly like the conditions that Virgin Orbit is going to face when they're launching that rocket.

[00:18:52] JM: Can you tell me more about the process of going from doing digital twin testing to real world testing?

[00:19:03] LS: So, if you think about the kind of continuum of going from just a pure computer simulation. And you think about, okay, what can I learn from a computer simulation? You can learn a lot. But sometimes physics, you're modeling physics. There's a certain amount of physics that you can model. There are limitations, because the world is tough to model, as it turns out. So, then you kind of move a step further, you say, okay, well, what, what parts of the real world can I start to co-model with the simulation? That might be where we start talking about putting software in the loop with other pieces of software, or putting hardware in the loop with the software, so that you start replacing pieces of the physical world that you modeled with real world elements.

And then you kind of keep taking that further and further until you're actually building prototypes that you expect to fly and just record data on. I think that's been one of the fascinating probably, points of view to watch with what SpaceX has been doing with their starship launched development down in South Texas is they've said, there's a certain amount of testing we can do in a lab, we maybe just want to launch a rocket that we know is going to fail. But we're going to learn so much more about the environment. And it's actually more timely or cost effective for them to accelerate the path through that testing and simulation curve, to incorporate more real-world data, knowing that they're sacrificing a prototype along the way. They take that data, they go back a step to their hardware in the loop models, re inform them, maybe re engineer some pieces, and kind of go through an iterative process to get there.

And then, in reality, it doesn't stop once you get to your final finished product, because you can keep instrumenting up those devices. There's lots of – I'm not trying to pick on Elon Musk here, but certainly Teslas are always recording data about their performance, and informing the design teams at Tesla about what's working and what isn't, to where the the testing process actually continues after it's left the factory, and that can be iterated back into future designs, or improving the simulation that you started from in the first place. So, the next time you go to build something, you're doing it with even more knowledge of the real world. And that kind of continuum, sometimes happens inside a small team in an organization. Sometimes it happens across multiple organizations. So, there's a lot of discussion about how to share data across that world. How do you make sure you're using all the data you've captured? That's probably one of the more fascinating ways I think the industry is going to evolve in the coming years is just trying to make sure that the massive amount of data that's being recorded at all steps of that process and in use is being put to better use, because it's really an untapped resource right now.

[00:21:49] JM: Let you just return to the case of aerospace. So, once a plane actually gets or a spacecraft or whatever gets to production and it's flying around and proven to be safe, I imagine there's ongoing tests and ongoing procedures to ensure the continued viability of of of different crafts. Can you give me a description of kind of the continuous testing process that a craft undergoes and what devices they're using?

[00:22:27] LS: Yeah, I mean, this is the classic kind of trade off analysis, like what measurements do you want to make, when and where? And what are the tradeoffs with doing that? Because in our world, yes, you could put the type of instrumentation that we make, you know, suitable for a lab or that bench I described earlier. You could put that on every commercial airliner, and be making measurements all the time. You don't, generally today, because it's more weight. It adds affects cost and the dynamics of how airlines make money. But in some cases, there are very small wireless sensors, not that we make, but that are available in the industry, that are going to be embedded into all the different engines, or going to be embedded in, or like strain measurement devices embedded in a lot of the fuselage of these aircraft. And those are recording data all the time. That's actually an increased area of emphasis right now, certainly on the engine side, to be able to constantly be measuring fuel efficiency, recognizing that they never want to put more jet fuel in an aircraft than they need because it's more weight, which makes it less fuel efficient. But also, just to be able to understand their procedures on – and a lot of this is settled science, of course. So, it's not like they're coming up with amazing new insights.

But when we're coming in at this angle, or at this speed, is that as efficient as this one? Or is it a specific pilot known to be better at doing this than another pilot? What about different weather conditions around the world? How much more insight do we need about being able to think about travel time or engine efficiency? And again, be able to have that just kind of be a constant stream of data. There are some examples that escaped me about overseas flights and the certainly terabytes of data that they're recording every time something's going across the Atlantic or the Pacific, just solely for the purpose of feeding that data back to the manufacturers. And then again, if that’s being done right, you're not only just feed it for how's that plane doing, but you're also feeding it into the design of our next engine, the design of our next air airframe, or the avionics systems, or all the different subsystems you can imagine on an aircraft. That's a huge area of emphasis right now, because so much of it is becoming micro tweaking, and the small changes you can make that could have a big impact on the economics or the safety of aircraft like that. So, you're going to see some persistent measurements that are going to happen all the time. Everyone would love to do more, but it just becomes the size, weight and power tradeoffs associated with the instrumentation.

[00:24:54] JM: Are there other cases where you would have multiple devices work together to do some kind of recording or reading? I guess, I want to get a sense for what it's like to test the integration points between some kind of multiple device workflow.

[00:25:13] LS: That's interesting. So, I'll start by saying that I think most of the systems, we deploy our multi device test systems. It’s usually not ever one parameter that you're measuring, it's the combination of maybe dozens of parameters that make up insight. And those parameters are going to be very mixed in terms of their measurement types or channel accounts, or you name it. A good example of that might be, I want to know how this engine is working at what speed the airplane is going. So, whatever vibration measurement or fuel efficiency measurement I want to do on the engine, I'm going to intersect with a measurement from the pitot tube explaining how fast the aircraft is going, and those will be different signals.

So, when we're developing our suite of instruments and our hardware platforms, we do a lot of integration testing. We have a really kind of elaborate setup, that for every new device we build, we'll put it up against the suite of all most all of the other representative devices we've ever built, and how well they both kind of electrically and mechanically, maybe radiated emissions, how much they might interfere with each other and make each other not be good citizens next to each other. But then also, how well they're used at the same time affects the ability to stream data, or how well they can be synchronized in time or phase across a backplane.

So, we're going to run through a whole bunch of those different suites. And sometimes that influences whether we want to use something like Ethernet as our common back or common communication bus for everything, or whether we like other architectures, high speed serial interfaces, or PCI Express, because we look at what in aggregate can we accomplish, while these things are working together across that single pipe, or multiple pipes? And how well does that actually meet the needs of the user. And it can vary dramatically, because there are a lot of – the examples I gave you before in that aircraft are probably not extremely bandwidth intensive. If you're making physical measurements in the world of vibration, or those sorts of things, you've been able to do that for a long time. And with as much data as you want, it gets more complicated when you start doing multi-channel RF, or microwave communication measurements, and trying to record a spectrum at the same time as you're moving, and those sorts of things are just bandwidth hogs, frankly. And so, the way we architect our instruments on our platform to accommodate those use cases, could be very different than how we architect different platforms for lower cost or lower bandwidth applications.

[00:27:53] JM: How do you determine the right sample frequency for recording device like a temperature sensor, or some kind of electrical sensor?

[00:28:06] LS: Yeah, so this used to be a bigger issue for some of those entities today. We used to have to really sweat kind of the Nyquist sampling theorems, if you're heard about those, where you have to sample at least two times the amount of the frequency of the signal that you're trying to measure, or to be able to represent different dynamics of it. You should be five times or 10 times. There was a lot of focus on that early when analog to digital converter technology was more of the bottleneck.

Now, I think ADCs have gotten so much higher resolution and higher frequency, that especially for something like temperature, there's only so many times you need to measure temperature in a second. It's not going to change that fast and it's true of a lot of the physical measurement phenomena, where you start to need a lot more of the kind of sampling theory starts to be in the RF or in the high speed serial interfaces, right now, that the computing platforms have evolved to be 400 gigabit or these different, very fast rising and falling edge devices.

So, if you want to be able to really characterize that or draw an I diagram of how that communications link is working, you're going to be using oscilloscopes that can sample it, extremely high sample rates, many, many giga samples, and that's even in hundreds of giga samples per second. And bandwidth that can be in the 10, 20, 30 gigahertz or more, terahertz kind of ranges with different techniques. So, that sort of matters more at different ends of the spectrum. And I think, it kind of takes a different dimension when you talk about communications, where you think about, “Okay, I've got a carrier frequency at this six gigahertz and I want a bandwidth that I want to be measuring at the same time onsite writing on that carrier frequency. So, what's the instantaneous bandwidth that I need going through? How much do I need to sample to get in the right Nyquist zones that?” It starts to get very RF black magic he in order to do it right. But that's again a lot of the tradeoffs we go through when we're designing these RF instruments. Is this an RF instrument that's going to be primarily to broadcast or measure like a 5G signal? Or is it going to do radar pulses? And how does it need to be architected to do one versus the other.

Ultimately, the more we can make that all about software, the better because every customer would like to have this universal measurement device. And with unlimited budget, you can approach that on some level, as long as you've architected the software, right and can do as much as possible and digital signal processing.

[00:30:38] JM: When you need to convert something in the real world, like temperature, or voltage measurement, to a – this would be an analog signal to a digital signal, there's measurement that has to occur, and then there's calculation that has to occur. Can you walk me through an instance of a measurement device and how you take that analog signal and reduce it to a digital signal?

[00:31:11] LS: Yeah, so typically, we try to make that as lean and simple as possible. But over time, it's required some level of filtering or amplification on the signal as close to the signal source as possible, so that you get as much fidelity as you can, and you're trying to maximize the content of that signal, before it goes into the analog to digital converter. You know what ADC you have, it's got a range for this, let's say plus or minus 10 volts or something, that's the range. So, how do I make this signal as close to the full extent of plus or minus 10 volts, so I can use all 16 bits of the resolution of that ADC, and get as many different steps digitally that I can represent the analog signal with.

So, there'll be some type of filtering or amplification in order to present the signal in that way. And then we hope that, the ADC is going to do a lot of the heavy lifting there. It's going to convert it into some stream of ones and zeros, and then we're going to be able to – we will reconstruct that the way our driver works, and then present it hopefully as engineering units, not just as ones and zeros. But as sort of the way you made the query and make this measurement. If you say, if the device is set up to make measurement and temperature, you're going to say, “Hey, give me this back in degrees Celsius, a driver can do the work to simplify that, so you're not constantly doing all that processing math on your own.” That's a low value-added task for most engineers.

But it depends. I mean, there's also a benefit to just give me the raw data, because I don’t want you to apply to any math to it. I want to be able to provide all the signal processing on my own, and we can do that too.

[00:32:53] JM: Do you have any examples of when a product gets shipped with a bug? And then having the bug discovered in the wild? And having to triage that bug and and figure out an appropriate fix?

[00:33:13] LS: Yeah, I mean, I'd like to say, we never really suffer bugs, that would be naive, of course, and just not true. So, we think about that in terms of what's the customer expectation on that? And what are the consequences to bug fixes, because there's usually some consequence. Now, you think, well, it could all be upside. But if you're not experiencing the bug, it's in a part of the driver are part of the product that you never touch, if we're constantly asking defense customers to upgrade their software, they get pretty annoyed by that, because it's somewhere that it's hard to do that with. Whereas in other places, like with our apps on our phones, right now, they get updated all the time, and just kind of always are presenting us with the latest, hopefully, bug less version of the software as we would want.

So, we think about that, in terms of the different customer use cases. We also frankly, have evolved a lot over time and how we're able to do that as we've shifted more from the waterfall methodologies of massive releases to more agile code drops on some of our software. We're getting more nimble about doing that. But again, it has to match with those customer expectations there. We're trending towards more versions of our software that are like, "Hey, if you want to standardize on something and not touch it, buy this version. If you always want the latest, feel free to buy whatever the latest release we've come out with is and we'll batch up the bug fixes on some of those longer lifecycle versions, whereas on the other ones, we might be able to be a lot more responsive to whatever bugs are found in the field."

[00:34:49] JM: Can you walk me through the supply chain and production side of things? So, once you get a device designed, maybe a prototypical device you can talk about in the lab and you want to get that device ready for mass production. What are the steps to doing that?

[00:35:11] LS: We're probably not unlike most hardware manufacturers in that sense that, we will chase a prototype of a technology block, we'll integrate it to where we feel like, "Hey, this is a candidate for what the release product would look like." We'll go through validation suites, we'll do design for manufacturability design for test analyses to make sure that when we do sort of throw this over the wall to the production facility, they know exactly how they're going to check that it's built to spec or that it's performing the way it's supposed to. They'll make cost analyses of when to do that, whether to do it, whether to test the bare board by itself, or whether to populate half of it and start running a different test. Or whether to test nothing until you get it completely built up and you just test for functionality, hook up the signals to it. And if it works, we didn't need to do any of that other testing beforehand.

That, I think is a microcosm for what happens in the supply chain across all levels right now, like on the semiconductor side, as things are going to systems in a package, where you might have a bunch of different dye that are all being put together, you can decide. Well, I want to test each of those dye in some way first or – it's too much to have all these different tests insertion points. I'm just going to wait and put it all together and do one test. Or sometimes, if I'm so confident in the design, I might wait until it's assembled into some kind of subsystem and forego testing there. Usually, it's not binary, usually, there's a real cost benefit analysis to different tests at different places, so you avoid scrap as much as you can.

But for us, and for most of our customers, that ends up being this big people process and technology kind of conversation about how they want to architect that, what makes the most sense for the economics of their product. And in some some cases, you look at and say, “Well, this is a $1 part that's going to go in some arcade toy game, that if it doesn't work, cost of failure is not that big of a deal. We'll live with 90% yield on those things, because who cares?”

On the other side of that, of course, is these mission or safety critical systems, like the automotive world deals with, where you're going to test that thing in almost every way possible, to ensure that there's no escapes from that manufacturing process. The same is true, again, for our parts. There are some that are very low cost that we're going to minimally test because the expectations are different. Some are very high cost that are going to mission critical applications, and we're going to test them as many ways as possible to ensure they meet the specs.

[00:37:48] JM: What about hardware updates? So, you always want to be refreshing hardware and improving it? What's the process for, I guess, first of all, a hardware device, meeting the expectations that it needs to? And then furthermore, once it reaches production, issuing hardware updates to it, and I guess, getting the customer feedback that leads to those updates?

[00:38:17] LS: Yeah, so probably broken record on this. It depends by customer segment. There's a lot of customers that would really prefer not to think about hardware updates, that would just think, “Okay, I want this thing to be this thing. And I want it to work as this thing forever, so I don't have to revalidate what this thing does. I have some confidence.” What they will do is they will send it either back to us or somewhere to calibrate it, so that it goes back into the specifications that it had when they bought it. But they generally don't think about field upgrading their hardware. There are other situations where – obviously recalls, if there's a problem, if there's something that yes, is safety critical, then that's a different scenario where we'll have to either go to them or have them send the product back to us to be repaired.

But in general, our customers like to buy the product for what it is, deploy it for what it is or was, and then if they need different capability, they look to the next model. And so, we do have to do versioning in that way, and a lot of times we maintain products that we know are not the latest and greatest, but just to give our customers the ability to keep buying the capability that they've spec into their systems. There's a lot of probably evolution on that mainly because of the roll of software. Again, as we continue to put more and more of the functionality in software, that gives us more of a common hardware platform. If it's a big old blank FPGA, that's the core of that instrument, and if there's some new capability that we want to deploy, either they can do it themselves or we can push a new firmware update, hardware is the same. That's becoming more ideal for us. That is the holy grail of a lot of my customers in the Department of Defense want, because they get buried in sustainment all the time of all of these really niche or esoteric black boxes or PCBs that were last manufactured in the ‘80s. And so, they want to shift all of that over to open systems, architectures, software defined radios, things that they can deploy, that really the brains and the thing that needs to be maintained is the software, the hardware is just kind of the software deployment vehicle at that point. That's an exciting trend both for them and for us, because we think we can do a lot more value delivery in that way. And it just lowers everybody's cost to sustainment.

[00:40:38] JM: Give me a sense of the the future of NI and where you're going beyond the current product lines, and kind of what what the vision for the future of the company is?

[00:40:51] LS: Yeah, I probably alluded to it in a couple ways. But we have this foundation, where we make great hardware, we make great application software, we help engineers develop test systems. But a big part of the future of NI and the future of the test measurement industry is about getting more out of data. It's about understanding more, and we made an acquisition two years ago, a company called Optimal Plus, that's a very well-established company in semiconductor production test, where they're installed at all of these production test facilities around the world, just acquiring data on all the devices, billions of devices a year. And then they're giving that data, of course, to the to the device manufacturers to inform their yield, to inform their process quality, whatever information they want to have. We feel like that's the tip of the iceberg.

We've been going down that path for a while. That acquisition accelerates that. But we want to make the connections between data that's acquired at all phases of the product development cycle with other phases. And that could also be with other companies. So, to be able to say, we run into this problem a lot where, I've got this chip, and I've got this chip, and I've got this chip. I put them together on a board. They're all within spec, they're all exactly as they're supposed to be. But for some reason, when I put them together, they don't work. And it's this kind of tolerance stack up phenomena, well, this one's kind of at this edge, and this one's at this edge, and this one's at this edge. Turns out, when you put them all together, they maybe don't work, right, even though they're all good.

We think there's more sharing of data and insight that can happen across supply chains, between vendors and customers or end users that can greatly improve the product quality that's out there, can improve time to market, ultimately lower cost for all parties involved. And we think that that can be established through a better set of common tools, a better set of standards that people are using to record data and store data. And then frankly, some cross-industry collaboration. There's a big initiative right now with the US Department of Defense, on zero trust for all the semiconductor supply chain, all the semiconductor suppliers, how can you guarantee that you've got that chip from the right place, and it's not a counterfeit? You can do that with data. You can compare the data that comes from the normal manufacturer of that chip to the data of this chip. It's not that complicated, especially when I'm sure your listeners hear about a problem like that, it’s like, “Oh, there's an easy algorithm you can write to do that.”

It isn't terribly complicated to do the analysis, what's complicated is providing the framework to set all that up. And we think we, as data producers, in a lot of cases from our test systems have a lot of value that we can add there. So, that's going to be a big area of emphasis. And not everybody is ready for that. Not everybody wants to embark on that kind of broad data quest. But there's a lot of steps in between there and where people are now. And there's a lot more we can do with software, on operational intelligence of how all these test systems are working, make sure that they're all operating as they should be, that all the devices are calibrated, all the performance indicators coming from that machine makes sense compared to other machines. It's a goldmine for machine learning opportunities to just bake in more intelligence into the way all of our customers are doing test, and ultimately then, how they're validating the performance of all the products that they're making.

So, we'll be continuing to pursue additional ways to add value and collaborate across the ecosystem to do that, as one of our big focus areas going forward. Of course, we'll continue to make more and better great instruments, more and better great pieces of software that engineers can use to, but the value can be up leveled quite a bit.

[00:44:35] JM: We've touched on a pretty wide variety of subjects. I guess, just to close off because this was an episode that wasn't as focused on software, tell me a little bit more about the integration points between the software and the hardware. I know we touched on this before, but maybe you could talk about another domain in which we could go into the the integration points between the software and the hardware.

[00:45:01] LS: Yeah, I guess I'll also echo maybe thanks to your listeners for suffering through more maybe more hardware centric conversation than they're typically used to. But I would say, we do think the hardware software interface is a critical thing to pay attention to, as we're designing systems for interoperability, or is we're trying to maximize the way that these systems are going to work, not just today, but over the long term. A lot of attention being paid to those types of interfaces is critical. There's a lot of industry initiatives right now on this. If I think about deployed mission hardware that goes into like that fighter jet I talked about earlier, there's an initiative called the sensor open systems architecture, that's going on with the US Air Force right now, to make all the hardware designs sort of fit into a standard pattern. So, that if someone's designing a computer in one of those card modular boards, it's designed in a similar way as another vendor might be doing it.

So, the Air Force decides, "Hey, this is a great mission computer for the F-35 or whatever aircraft", but that vendor goes obsolete, they can drop in another one that works in the same way. That's certainly like a connector pinout, power consumption, form factor discussion, but it's also a software interface discussion. Because if you don't have a sense of the common software architecture that's going to talk to that device, the hardware interoperability only gets you so far. And there's been a lot of emphasis, again, with the US Air Force on designing the whole stack of their mission equipment in such a way that it's containerized and it can be upgraded in the field.

There was a big article a year ago about how they upgraded the software on one of the spy planes while it was flying. And the spy plane was built in the '60s, the U2. And that's sort of what you start to enable when you've been thoughtful about the hardware and software interface of these types of systems. It's also clear that we spend too much on designs of those types of defense systems. The reason we do it is because we're over engineering things and making them very proprietary or confidentially architectured by one integrator, and that's just not the speed at which, probably most of the listeners on this call are functioning right now. So, watching that evolution, I think is pretty fascinating, because it's cultural. It's how much of the stack do we own, versus we leverage? How do we think about deploying the software? How do we think about that hardware software interface? It's pretty fascinating to be a part of it, and we don't do much deployed hardware like that exact example. When we try to use those same patterns of thinking in all the test systems that are being designed because that's obviously a far better way to do this than making them very stove piped and hard to maintain and hard to upgrade over time.

[00:47:56] JM: Well, Luke, thank you so much for coming on the show. It’s been a real pleasure talking to you.

[00:47:58] LS: Yeah, absolutely. Loved it and hopefully, was useful for your listeners as well.

[END]