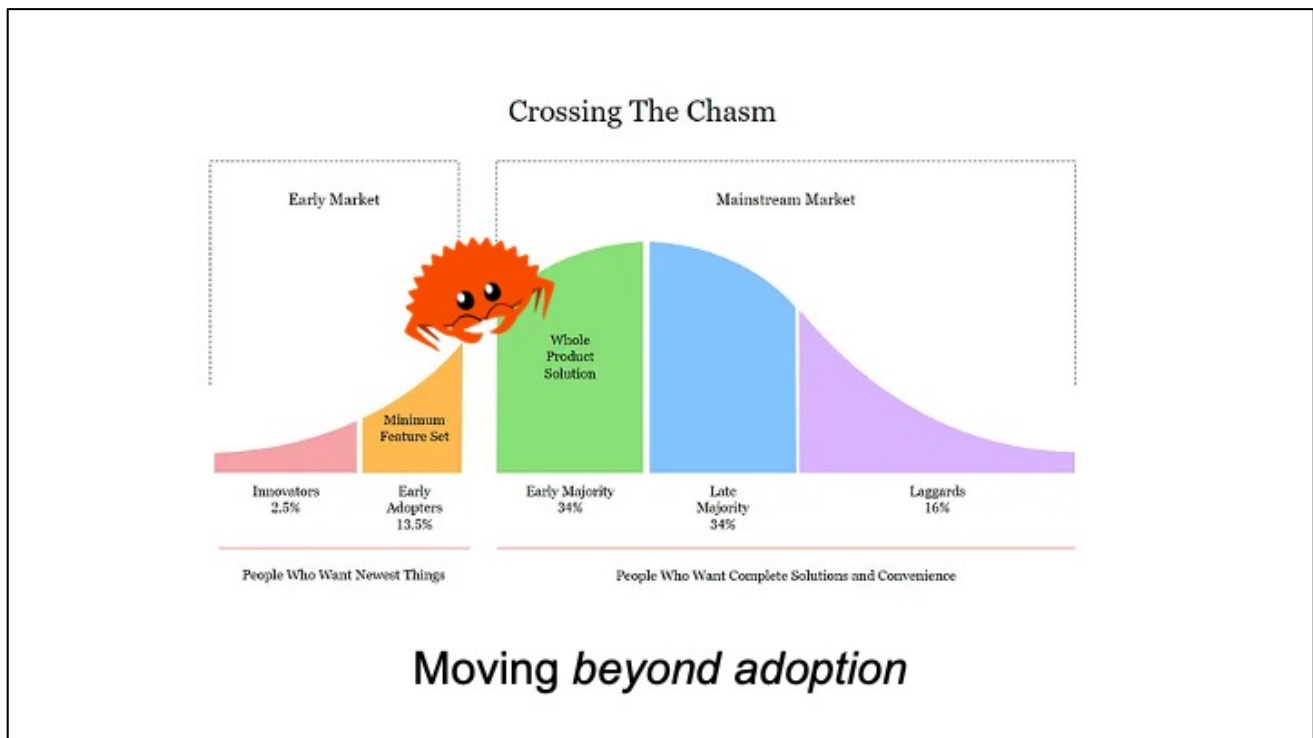




Vision Doc
The Next 10 Years
of Rust

Jack Huey
Niko Matsakis



Niko: So, it's been over 10 years since Rust 1.0 – and we've seen a lot of success. We've moved on from the Early Adopters, folks who picked up Rust because they wanted to push on new things, and we're starting to see people picking up Rust because it's the right technology for them to use, even if it still has rough edges. That's awesome and exciting – and it leaves us with an interesting question... What now? It used to be that when we tried to decide what to do, we could look at "adoption" as the goal – we needed Rust to grow. But I think we are approaching the point where that's no longer true, where we want to start asking ourselves – so, what are we doing this for? I mean, we don't expect everyone to write all their code in Rust, right?

(Jack: gives Niko a look)

(Niko: RIGHT???)

Rust

A language **empowering everyone**
to build reliable and efficient software.

Jack: Look— let's go back to first principles. IF you go to the web page, you'll see this slogan – and there's two key words there. Empowerment and Everyone. The point of Rust has always been to take things that were impossible, or at least impractical, and bring them within reach. We wanted to make building reliable, efficient software something anyone could do – not just traditional systems wizards, but people working in other domains as well.

What do we want to know?

- Why do people love Rust?
- What do people find challenging?
- How does Rust impact people across the globe?
- What are the domains where Rust can empower people?

Niko: Yeah, and so, the point of the vision doc is to go out and talk to people to understand how much we've succeeded. The idea is that we need to build up a shared understanding of just what it is we have done here – we built something that people seem to love. But what do they love about it? What is still challenging? Does that vary country by country? And, perhaps most importantly – is Rust doing its job? Are we empowering people to do new things? And what are other areas where we could knock down barriers?

“But don’t we already know everything?”



Jack:: But Niko– I mean, don’t we already know the answers to those questions? I mean, I’m a Rust user. I know what I love about Rust.

Niko: Yeah but – that is just YOU. Are you sure that’s the same as other people?

“But don’t we already know everything?”



Niko: Well, I mean, I talk to other people too, there are RFC threads...Zulip discussions...

Jack: Yes, that’s true. But then again, let me ask the audience a question – how many of YOU have participated on an RFC thread? Asked a question on Zulip.

“But don’t we already know everything?”



Jack: If we really want a holistic picture, we want to make sure that we’re not getting bias by who we are talking to. There are a lot of people using Rust now who don’t really participate in the old mechanisms we had for getting feedback, so we decided to try a new one.

Qualitative Research!



Don't introduce biases



Talk about *experiences*, not *opinions*

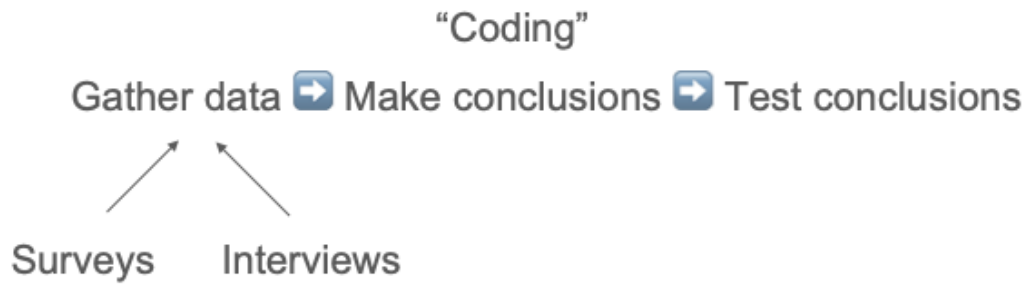
Qualitative Research!

Gather data → Make conclusions → Test conclusions

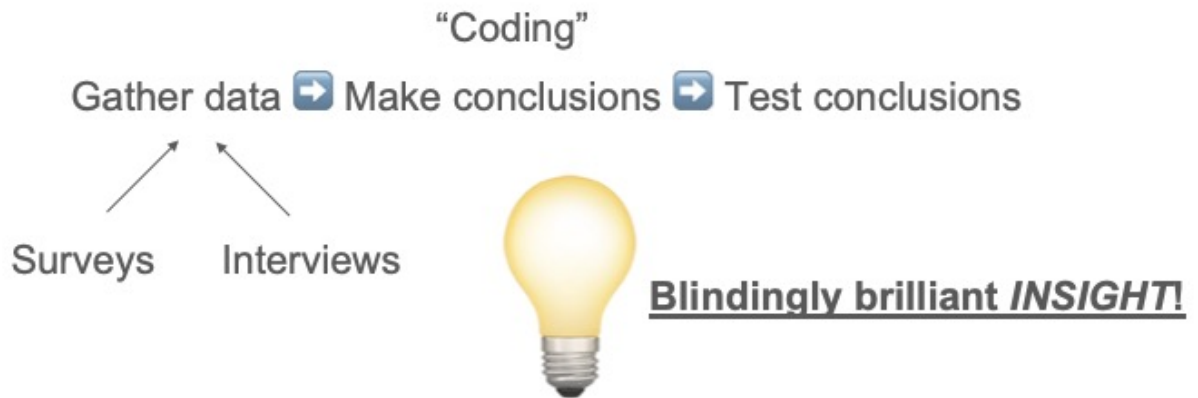


We're here (sort of)

Qualitative Research!



Qualitative Research!



What did we get? 5,499 responses.



Global coverage?

Mostly USA and EU, but also Asia, Africa, LATAM.



Language?

Top languages were Python and JavaScript, but lots of C++, Java, bash...

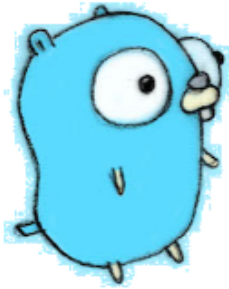


Domain?

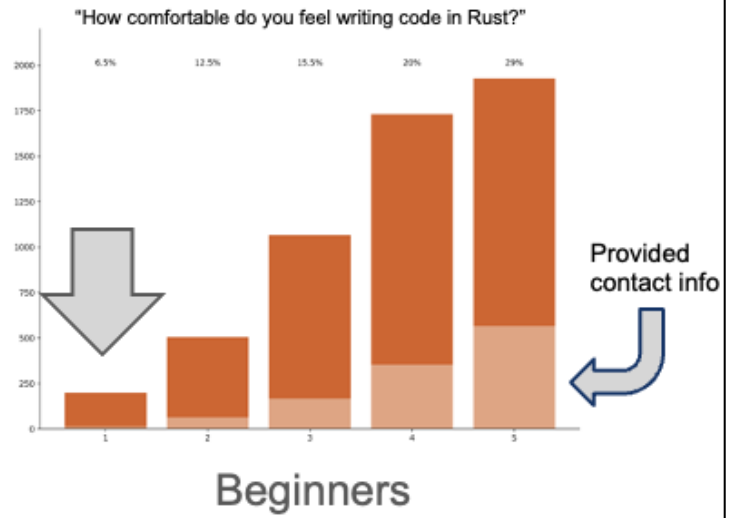
Everything from async to embedded to CLIs (lots of CLIs...)

More details available in our RustNL talk 2025...

Two notable gaps



Gophers



Phase 2: Live interviews

- Conducted ~65 interviews with a wide variety of people
- Each 45 minutes long
- Kept transcripts and recordings (private)

Status: Starting to wind down now, trying to fill remaining gaps (e.g., Rust haters).



What is an interview like?

- "How did you first start using Rust?"
- "What was the last time Rust felt really great?"
- "What's the last time you felt confused using Rust?"
- "Have you ever recommended Rust to someone else? Why or why not?"

You might be wondering how an interview goes. The key thing is that we are looking for facts, not opinions. It's very easy to accidentally ask questions that prompt people to tell you what they think **SHOULD** be true, but not necessarily what is **ACTUALLY** true. For example, to try and learn what people love about Rust, we don't ask them "So, what do you love about Rust" directly. If we did, we'd likely just get what they think they **should** love about Rust, not necessarily what they **actually** love about Rust. Instead, we'd prefer to ask a question like, "When was the last time you felt like *Rust is awesome!* and what made you feel that way?"

#FALLONTONIGHT

Journey... to Rust

♪ DON'T STOP ♪



Empowerment

I studied civil engineering and I studied from the front-end my own, self taught. I was doing high-level languages like TypeScript and Python.

But there was a huge black box, like system-level, lower-level, and I got curious. *I was exploring different languages at first but Rust had a really strong culture, it seemed cool.*

– Full-stack developer



You, too, could be a Rock Star!

*There were quite a few people on Twitter talking about 'I'm a **JavaScript developer** and I'm going to try **Rust**'. That kind of attracted me to it.*

— New dev

*I chatted with about new programming languages with my colleague and he said, '**Have you ever seen Rust? It's so cool.**'*

— C++ dev



Rust at work

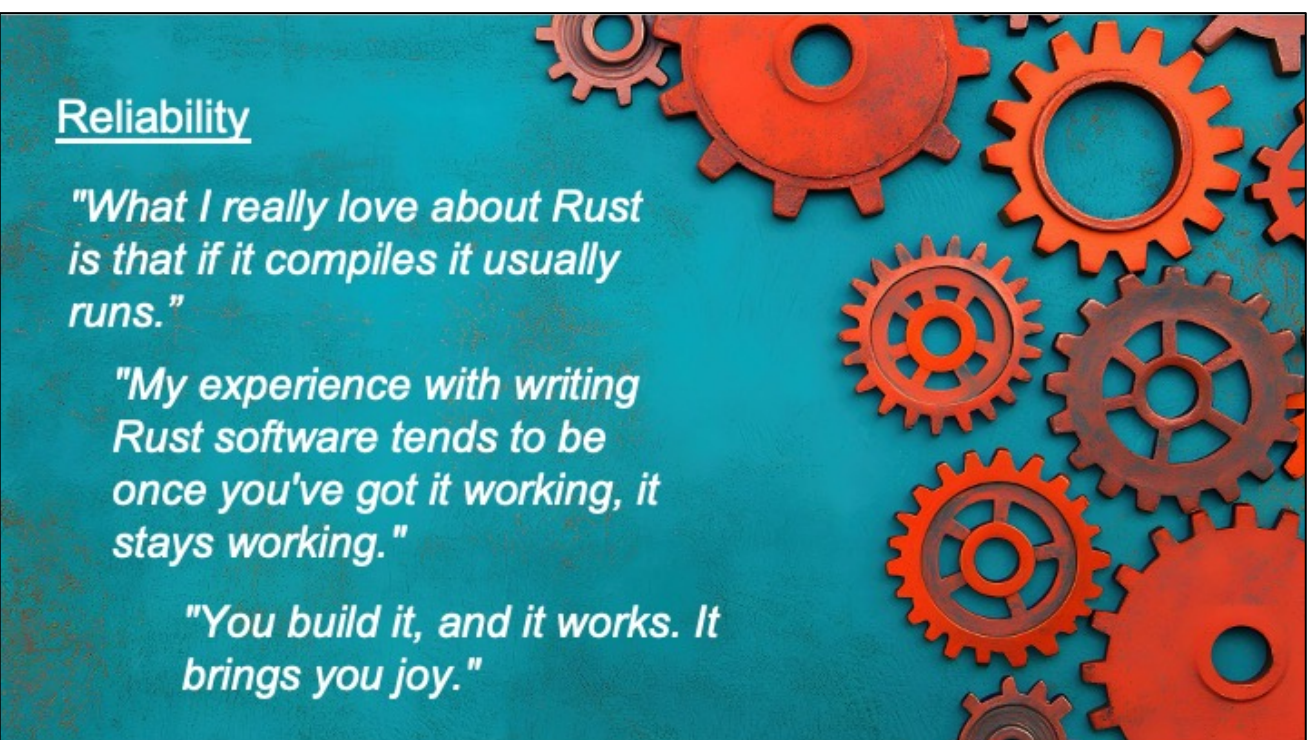
"I joined the team in February this year and until then I had never used Rust. I had heard Rust was good for system-level work. "

— New Rust dev using Rust at work



What people love



The background of the slide features a teal, textured surface. On the right side, there is a cluster of interlocking red gears of various sizes, some with a metallic sheen and others appearing more like solid red plastic or wood. The gears are arranged in a way that suggests a complex mechanical system.

Reliability

"What I really love about Rust is that if it compiles it usually runs."

"My experience with writing Rust software tends to be once you've got it working, it stays working."

"You build it, and it works. It brings you joy."



Polished tools

"The whole cargo thing makes it so much more attractive to use."

"[Building and deploying binaries] just freaking worked."

"The [lints] are really good at telling you what's potentially wrong."

Versatility

*"You can do C and Python
with one language. Rust can
go very low and go very high."*



Strengths

Makes one a lot more conscious of pitfalls and ways to think about **memory management** that I was previously blind to

If it compiles, it works, much faster dev-compile iteration, better together with coding agents

Increased performance



GitHub Copilot flattens the learning curve

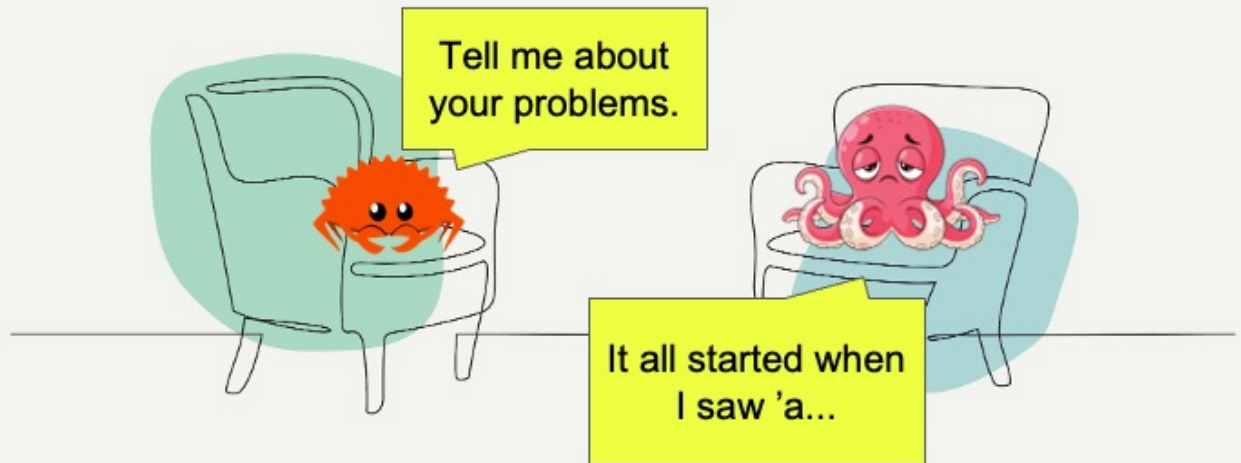
Data-race-related concurrency bugs reduced

Reduced friction == more **motivation** for devs to write tests

Rich ecosystem and streamlined dependency management

Memory-safety related vulnerabilities reduced

What people love...less



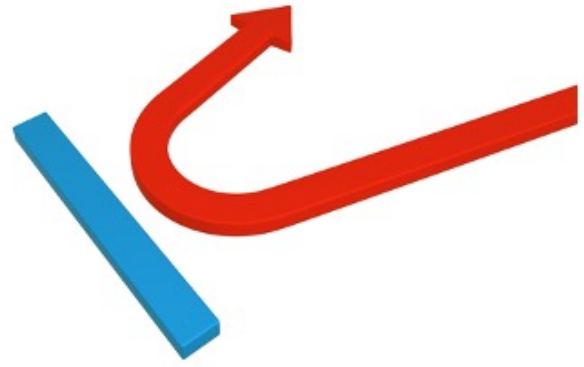
Learning curve



Giving up and coming back later

*As much as I like Rust, when I first started with it, I **actually quit after about a week** or so trying it just because, I didn't really understand the borrow checker and stuff.*

– New programmer



Giving up and giving up

I tried it and it was difficult. And it has stayed difficult, no matter how much I've used it. There are always concepts I can't understand (even after reading about it multiple times).

– Survey respondent



Core struggles


"Lifetime annotation soup"

"Pin<Box<dyn Future>>"

*"Fighting with
Re<RefCell<T>>"*

"Adding `&` and `*`
until it compiles"

The dreaded
'expected &str, found
String'

A scenic landscape photograph showing a mountain range under a vibrant sunset or sunrise sky. The sun is a bright, glowing orb on the right side of the horizon, casting long, radiating rays of light across the sky and the mountain peaks. The sky transitions from a deep blue at the top to a warm orange and yellow near the horizon. The mountains are silhouetted against the bright sky, with some peaks catching the low light. The overall mood is peaceful and hopeful.

Progress is possible!

*"I started somewhere around Rust 1.0. Without NLL it was a painful experience. **Now it is much better.**"*

– Survey respondent

Low hanging fruit?

I started off using `Result` but all the information I found has example code using `anyhow`. It was a bit of a mismatch and I didn't know what I should do.

— *Intermediate Rust dev*



Low hanging fruit? — Global perspective

Structural resources can supercharge grassroots communities that already exist

e.g., meetup
equipment for remote
speaker experience

Translation efforts remove barriers to entry.

e.g., Rust Español
translation effort



Empowering means removing blockers, even when they aren't code.

Journey to Rust — Global perspective

80% of the people learning Rust in Africa are writing in Solana or one of those. Nobody is actually learning Rust to use it for the normal regular things.

— African developer

The very first community that helped me is the blockchain community. [...] They have a willingness to give support to beginners to grow the ecosystem. From there I could more easily tap into the libraries and stuff.

— South Korean developer





Universities drive Rust adoption
(particularly outside of US/EU)





Blindingly brilliant *INSIGHT!*




Rust and Safety Critical

These blindingly brilliant
insights brought to you
by the hard work of...



What is safety critical?



ASIL	Classical Examples
ASIL QM	Infotainment (audio volume, ...)
ASIL A	Cruise Control
ASIL B	Instrument Cluster (speed display, ...)
ASIL C	Passenger Airbag
ASIL D	Actuation (electronic steering, ...)

ASIL => Automotive Safety Integrity Level

"What if [currently-used languages] are not considered "state of the art" anymore? We [the Automaker] should have a back-up plan [by starting to use Rust as we have]."

> Before the left-side:

"Safety-critical software is embedded all around us such as cars, airplanes, medical devices and so on

Historical languages used are from a different era; some challenges exist at higher levels of safety-criticality which are solved in Rust already, such as memory-safety, type-safety, and thread-safety. Rust gives you those things as intrinsic characteristics of the language.

Safety-critical systems fit the mold pretty well for foundational software; no-one cares how the brake pedal makes the car stop, it just *should*."

> Regarding the left-side:

"Increasingly, even for liability reasons it's become important to certain industries to incorporate Rust as a hedge against risk as you can see from the paraphrased quote on the left from a software engineer working at an Automaker."

"As you can see on the right there's different levels of safety-criticality in the Automotive industry.

As the level of safety-criticality increases within a safety-critical industry it's typical for

the complexity in meeting the standard to increase dramatically, somewhere around a multiple of 1.5x conservatively per level. As shown by the document produced by MISRA earlier this year, roughly half of the MISRA C coding standard would not apply to Rust due to the nature of the language and compiler.”

Who did we talk to?

Domain	Integrators	Suppliers
Automotive	3	11*
Aerospace	2	5*
Industrial	1	5*
Medical	1	0

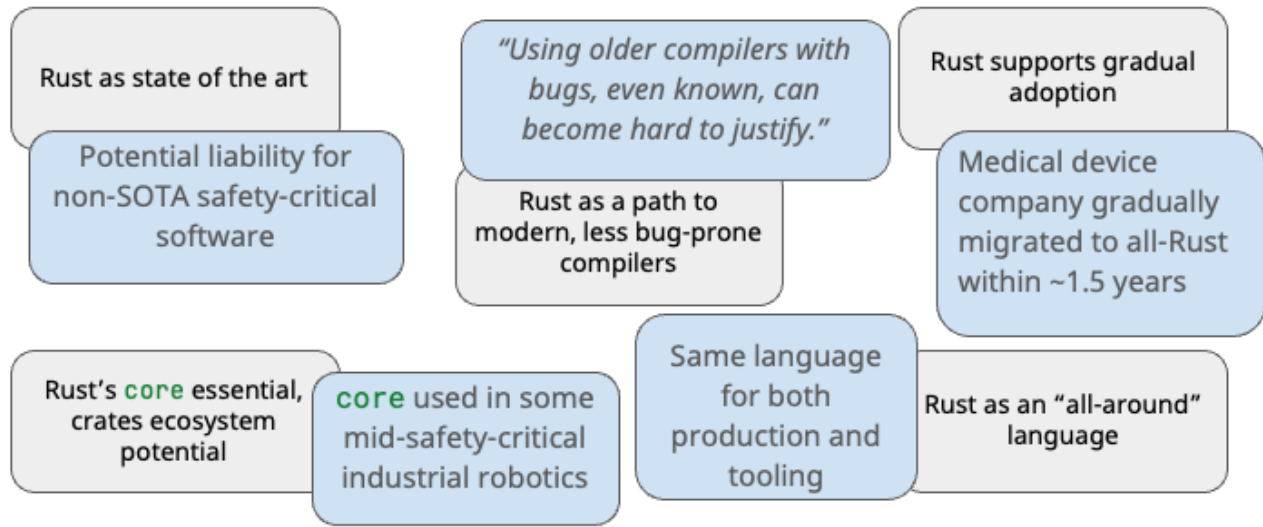
* some overlap

"[As an Aerospace integrator] Rust lets us meet aggressive timelines with confidence. [...] Rust lets us prototype and ship to production [in the same language]."

"You can see that we spoke with a number of integrators and suppliers of safety-critical software. There are some industries we didn't have bandwidth to tackle in this go-round, but would like to incorporate feedback from in the future, such as rail and nuclear."

"You can see a paraphrased quote from an Aerospace integrator that speaks to a different between how things had been done historically and how they can see a dramatic increase in their ability to ship. Previously they'd seen time allocated in their schedules for first prototyping in one language, say Python, and then eventually translating this all to say C++ for production purposes. This step has been removed with the shift to using Rust."

Themes in Safety Critical



"When we reviewed across the interviews performed of those in safety-critical domains, we came up with a number of observations that we bucketed into *themes*. Let's go over them a bit one-by-one."

> Rust as state of the art

"Rust is coming viewed as *state of the art* and quite usable at certain levels of safety-criticality. A proponent for Rust at an Aerospace integrator noted that Rust is seen with some skepticism at first. However, he shared that if you show up and ship these arguments tend to dissipate.

Because safety-critical industries are regulated, there are standards per industry which must be followed. In some safety-critical industries such as Automotive, the standards discuss a technology being *state of the art* and this can have legal liability implications. More on this in a bit."

> Rust as a path to modern, less bug-prone compilers

"Currently with languages used historically in safety-critical industries such as C and C++, compiler versions are pinned for the length of a project or even maintenance of a project. Project maintenance could be 10-20 years as shared by an architect at an Automotive supplier. An engineer at an Automotive supplier shared with us the common view that it's seen as safer to use older compilers with known bugs, rather

than an newer compiler which may have unknown bugs and incompatibilities. Rust has a chance with its Edition system to supply longer term stability, with non-breaking changes possible to use.”

> Rust’s core essential, crates ecosystem potential

“Rust’s core library was deemed essential and put to good work by robotics integrators at a lower level of safety-critical in the Industrial space. The crates ecosystem is rich and offers potential to lower levels of safety-criticality or for adaptation as shared by a team lead at an Automotive supplier. In both cases, at higher levels of safety-criticality it’s essential in certain industries to have libraries used pass safety-certification.”

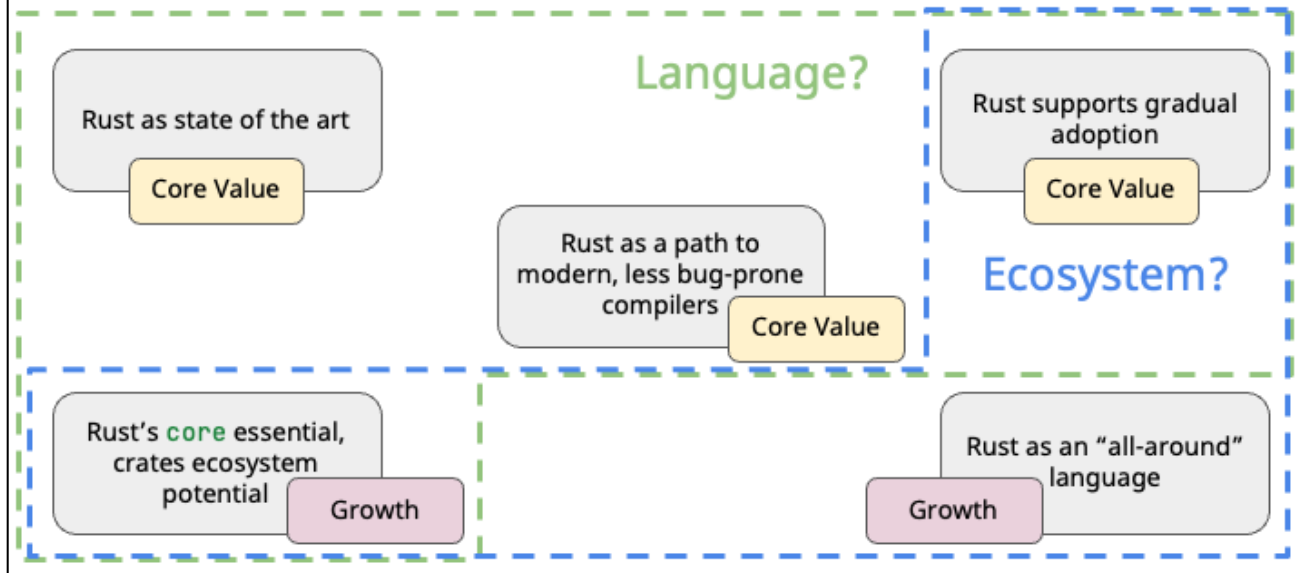
> Rust supports gradual adoption

“We’ve heard stories from different industries about success in gradually adopting Rust. For example, a software engineer at a medical devices company shared that they were able to start at the ‘leaf nodes’ of their existing Python project and work their way inwards, completely replacing with Rust in the course of about one and a half years. However as shared by one architect at an Automotive supplier there are sometimes complications around interoperability and shipping 2 compilers that have given them pause for starting new projects in Rust.”

> Rust as an “all-around” language

“A lead at an Automotive chip vendor shared about their experimentation with the ability to use Rust in domains they would have used different languages in the past. Where in the past the embedded engineering using C would need to request tooling support from another team writing tools in C++ and Python, he envisioned that it would be possible in the future to have one engineer see the process all the way through.”

Themes in Safety Critical



"Beneath these *themes* there are a number of *issues* or *opportunities* to improve Rust the language and Rust the ecosystem. We attempted to segment where possible which themes primarily fell which grouping."

"Note that there are annotations of: Core Value and Growth on each theme. These correspond roughly to the areas where Rust provides strong core value to safety-critical industries and where growth in these themes could be seen as useful to expand where and how Rust can be used in safety-critical industries."

"As an example of a theme that can fall under both the Rust (the language) and Rust (the ecosystem) we can see Rust supports gradual adoption. This falls under ecosystem, since Rust has a strong and evolving ecosystem of ways to interoperate between Rust to C, C++, Python, and so on. This falls under language, since it's likely to see changes be requested to support more seamless interop between C++ and Rust under the umbrella of the C++/Rust Interop Initiative."

Next steps



More interviews



Analyze data



Write and validate insights

Larger organizational story

Project goals

Flagship goals

Vision RFC



See <https://rust-lang.github.io/rust-project-goals/>



That's us!

*"All the things...[if done]...,
would make me a happy Rust
programmer. If not, then I would
still be a Rust programmer."*

*“[Rust] creates a community of practitioners who stay **because Rust enables work they couldn't do effectively before**, not just because they like the language.” — Interviewee*

...let's not just make a language people *will* use;
let's make a language that people *love* to use.