The Open Source Way

Episode 13: Rust – A Language on the Rise

Transcript

Karsten: Welcome to the Open Source Way. This is our podcast, SAP's podcast about the difference that Open Source can be. In each episode we'll talk with experts about Open Source and why they do it the Open Source Way. I'm your host Karsten Hohage and today I will talk to Andre Borrmann and Philippp Degler about Rust. Andre celebrated his 10th anniversary at SAP last year. He started as an SAP solution manager consultant and ABAP developer and is now part of the Strategic Development Projects Team as full stack development expert. Last year, he started the SAP internal knowledge transfer series "Rust: What the Crab?" to spread the knowledge about Rust within the SAP. Philipp is a professional C++ developer by day and a passionate Rust developer by night. He joined SAP in 1999 – just like me, by the way. Since then, he has worked in different areas, from high-level application ABAP development to low-level systems development. Currently, he works as a development expert in the ABAP language group, and together with Andre, he runs the Rust at SAP community. Now, hello Andre, and hello Phillip. In the beginning, when we talked about Rust, there were five of you; now, there are two. Is it right you kind of gambled about who would get to be in the podcast?

Andre: Well, yes, we recently did that because all of us were keen to be part of this podcast series, and so we played a famous variation of the game "Rock Paper Scissors," which some of you might know from the "Big Bang Theory" movie series, which is "Rock, Paper, Scissors, Lizard, Spock." And happily, Phillip and me just won all the rounds we played.

Karsten: Okay, the Lizard Spock version is like the nerd version, is that right?

Andre: Yeah, that's true.

Karsten: Okay, well, it's an IT podcast, what do you expect? How does lizard look?

Andre: I guess it's like a half opened hand, like if you were trying to bite someone.

Karsten: Okay. So it's like the silent fox from school just with the open mouth, right?

Andre: Yeah, exactly.

Karsten: Okay, and Spock, I guess, is the two fingers, two fingers live long and prosper. Right.

Andre: Yep.

Karsten: Okay, do I qualify as a nerd or...?

Andre: Yeah, you do.

Karsten: Okay, great. Then then I'm basically authorized to talk about Rust with you, it seems. Now, Phillip, apart from the first thing that comes to mind – a mixture of iron, hydroxides, and oxides – what's Rust and why is it at least recently getting important for SAP?

Phil: Well, let me rephrase the question a little. It's not a question of what Rust is, what Rust is not, and in the first place it is not a game. It's a programming language. Sure, it's also a game, but it's a really good programming language. And it's not just another programming language. It's really an evolution in programming language, at least for systems programming languages. And there is a big hype currently in the community in the programming world. Rust is now available, or let's say it has been in the open for quite some years, but it gains more and more traction. And I think part of this reason is that the mother of the language, I would say, the founder, is Mozilla. Mozilla was one of the companies that made Rust happen, and they really put a lot of love and a lot of work into this language, together with an Open Source community that they founded from the get-go, so Rust was Open Source from the start. And I think this is one of the reasons the language is so successful. There is proof that it was a good idea to do it like they did it - open source and together with all the experts from around the world: When you look at Stack Overflow, they did a survey, or at least there is a category in the survey for the most loved programming language, and Rust won this category five times in a row. And the interesting part is, you get a questionnaire that you get as a developer or as a participant in Stack Overflow, and you should rate certain aspects. And for the aspect

most loved programming language - when you look at how the question is phrased: "When you have tried the language, would you use it? Would you keep using it?" So, in my opinion, a lot of talks about Rust talk about this aspect that Rust is the most loved programming language. But if you look really clearly at the question that they ask in Stack Overflow, it's really more like: "You used it, do you still like it?" And the answer is a clear "Yes!". And this is very important, for a programming language, because when you try it and you keep using it, then that means it must be good, right?

Karsten: Then now we have the survey result that it's a liked, a loved language; is it also being used correspondingly often?

Phil: Yes, definitely, yes. everybody who is part of the programming world and who also participates in the Rust community can see it on Reddit on all those other social channels that you can follow Rust on, there you can see that a lot of famous companies lately joined the Rust community and announced projects that they are doing with it.

Karsten: Okay, just making sure, sometimes things are loved very much, but are still not very successful.

Phil: Yeah, absolutely.

Karsten: André, so it's the most beloved language or the one with the greatest survey results in Stack Overflow for five years at least. How does that impact development at SAP overall?

Andre: let's do a short time travel before we answer that question. Because in 1972, when SAP was founded, we started with our own development language. It's called ABAP; you have recently had talk about that as well. ABAP could be also seen as a first flagship of Open Source, even though only customers who bought SAP software could really look into the source code, if you like. And with that language, SAP was capable of being a market leader for enterprise software right from the beginning. And that's because we promised to provide stable, safe, reliable, and maintainable software for decades to come. But if you fast forward a little, then you can pretty easily see that the market has changed tremendously. New technology has been invented and flooded the market, if you like, like technology platforms, cloud products and the like. So, SAP was

kind of under pressure to keep up the momentum of being a market leader in those sections and also grow into those new spaces like technology platform and cloud. And so, from my point of view, Rust is kind of a natural and perfect fit there, because Rust's promises are exactly the same. So being able to provide or implement software, which is reliable, safe, stable, and secure, so to speak.

Karsten: You mean, the Rust promises are exactly the same, the same promises as the original ABAP promises?

Andre: Well, I would say they are quite likely the same, I'm not sure whether they're really exactly the same, but they're going into the same direction, so we as SAP company have made the promise to our customers to provide them with safe and secure and stable and reliable software. And that's kind of also the thing, which is written on the banner of Rust, if you like, really, so that you have a programming language that allows you to develop software that also adheres to those promises.

Karsten: Just trying to make sure that we're getting the differences and similarities right here, because you also said something about ABAP being kind of one of the first Open Source languages; of course, as you said, not by license, but only because if you had access to the files, they would be readable.

Andre: Exactly.

Karsten: But is that also the case with Rust? I thought that's a compiled language. And so, if you get a delivery of something executable, you can't read it anymore.

Andre: Well, that's true if you look at the compiled binaries, but the delivery or distribution of all the components you would need to compile your final binary is delivered by source code. So, the different pieces and parts are called crates at Rust. And there is a huge, hosted web page, which is called Crates.io, and there you can find a whole bunch of libraries and crates, all readable source code, and will always be downloaded as source code packages to be compiled into your final binary. And the same is true for the whole Rust compiler, So the Rust language itself is also completely available as Open Source.

Karsten: Cool, just trying to be sure that I understand the comparison correctly. So let's get back to what it means for SAP. You were saying that the Rust promises fit perfectly.

Andre: Exactly, that's one part of the answer, I would say, and the second thing is that historically core components of SAP software are written in huge chunks in C and C++, and Rust comes out of the box with a great interoperability into existing C or C++ code bases. So that would be also a great candidate to, step by step, migrate or refactor security-relevant parts of our software, for example, with Rust, to reap the benefits of that programming language. There's also the situation currently that if you look at job postings, for example, there are more and more job postings that contain Rust as a programming language for new talents as "nice to have" or even as a prerequisite. And we at SAP always try to hire the "best-in-class" talents. So, we can't really ignore that part. And that's a movement as well.

Phil: Maybe I can add to this: In our area, we recognized we had an internal project, we wanted to create a tool and we decided to create this tool in Rust. And then we announced it and said, hey, if there are some students or some colleagues who want to help us work with this, this project would be fine. And the interesting thing was we had a lot of those specific projects that we offered to students and stuff, but this very project got the most attention. We even had colleagues in areas who were not students or were not really looking for something to program as such. They just saw that there was something to do with Rust and they asked: "Are you doing things with Rust? Can we do this project together with you?" So, there was some internal interest which really overwhelmed us and clearly showed us that the interest in Rust, not only outside of SAP but also inside of SAP, was quite big, surprisingly.

Karsten: And so that was within a number of all Open Source projects where Rust created that main attraction or was it not necessarily all Open Source projects?

Phil: No, in this case it was just an internal project, more or less also a toy thing, I would say, because we were offering those positions for students that should not do productive code. They just had the chance to meet SAP and to do some projects with us.

Karsten: I'm just trying to find out if it's just the overall attractivity of Open Source with them young folks or with developers, no matter how old they are, maybe, or if it's something particular about Rust. Because from what you've said so far, okay, it's a programming language, it has a compiler. It's cool. It's on the rise. But what's the real difference compared to other languages that makes it necessary or so cool?

Andre: It's not only the fact that it is Open Source and it's not only the fact that it's Rust, so it's both of them, but also the thing that Rust really does different right from the beginning. So, if we see programming languages that have been there for decades, that are complex and current software products, they all suffer from the same issues. Recent analysis of other software companies has also shown that the main root cause of, for instance, security breaches or other bugs are off memory corruptions. So, issues with memory access, heap or stack overflows, double free of instances or uses of memory instances that have been freed already or kind of buffer overflows like accessing arrays out of bounds and the like. Right. And while those languages really evolved over the years and each language has tried to tackle those everlasting issues differently already, coming up with new ideas or concepts like using garbage collection or introducing after-development tools like scan tools, vulnerability scan tools, or securities scan tools to find out whether the developer has introduced some bug while accessing memory or the like, Rust really grabbed those issues at the root and introduces completely new concepts to that to ensure that this kind of runtime surprises can't really happen. So that all checks are - as much as possible - executed during compile time and give the developer a good feeling while saying: "If I am the compiler shouting at you, don't worry. I may act like a sergeant sometimes. But if I'm fine with your code and it compiles, then it will run. It will run safely without any issues with memory corruptions or the like." So, there will be no runtime surprises, so to speak. And the real second pillar, which is also, I would say, as important as the language and the concept itself, is the community, the Open Source community. They are totally friendly, and they are always there to help you to become a better developer with Rust, so providing this kind of stressed term idiomatic. So, they really help you to write better Rust code that not only compiles, but it also is concise in a way that you have good quality code that you can provide and have in your software product, so to speak.

Karsten: Okay, so you have this combination, like when you say the community, you're still allowed to play with your friends, but when you get home to your compiler,

someone's very strict with you and tells you to, I don't know, "wash your hands before you eat dinner!"

Andre: Exactly. "And wash them twice!"

Karsten: Okay. And other compilers don't do that?

Phil: Sure, they do, but differently.

Karsten: Okay. How differently?

Phil: Well, I think this is one of the really, really important aspects of a tool that you use as a developer: you have to have good error messages. So, when you run into a problem, the better you understand the problem, the faster you are fixing the problem, the faster you are evolving your code. And the Rust compiler really has extraordinarily good error messages. So, this is a completely different level compared to C or C++ compiler output.

Karsten: I think even us non developers know that there are very different error messages, and I can imagine that that's the case in compilers just as well, but also from compiled software that's actually there for me to use. We all know that some error messages just tell you something went wrong, and other error messages tell you what went wrong. And even better, some error messages even tell you how to fix what went wrong.

Andre: And that's exactly what the Rust compiler does, right? So, it does not only tell you what is wrong, it's really verbose. So, it really tells you what the current issue is and what might be the possible and helpful fix for that issue. And that's kind of also – I mean, I wouldn't go that far and say that's an artificial intelligence thingy, but it's kind of a trainer, right. So even though you don't have a trainer in person who explains what Rust is and what the syntax and language construct is, and even though it is a steep learning curve, you can start right from the beginning and the compiler is also there to help you to become a better developer.

Phil: But I said that other compilers might not have ideal messages or error messages; I don't want to bash any compiler here, I mean, Rust really has the advantage that it's a new language. And they started from scratch and they could learn from others' errors. And clearly, Rust is also a mixture of good concepts and of nice features from other programming languages that they have adopted. So they took the good things from other languages and put it into Rust.

Karsten: for you as a developer, what does all that we have learned about Rust or talked about Rust so far mean to you?

Phil: Yeah, I think I want to tell a personal story here. I just looked it up before this talk. It was 2014. We have a platform that is called openHPI. It's a platform sponsored by the Hasso Plattner Institute. There you can take free courses and learn and improve your skills. I did so, too, with a parallel programming course. And at that point in time, the students of this course should program a parallel program. At that point in time, I was not too much into parallel programming. So therefore, I also did this course, right. And then we were free to choose the language to implement this task. And I just read from some news blog, I read about Rust and I said, why not try Rust? And this is interesting. I never had this effect before with another language I really fell in love with the language from the get-go. So, it was love at first sight. And then, also, I had some issues: As we just said, the compiler is really harsh. The learning curve of Rust is a little bit higher than other systems programming languages, maybe. So, it takes you a while until you get your program to compile. But for me, the most impressive thing was, I think it was a heat thing, you should distribute heat on some surfaces and stuff like that. So, this was a parallel task.

Phil: And I finished the program. I ran the program - and it just worked. I was really blown away, like I really had to fight hard against a compiler. The compiler was yelling at me all the time. I was frustrated like hell. And it finally compiled, so that it finally compiled was a big relief for me at that point. But then it just worked. Yeah. And I think this was the moment in time at which our love at first sight grew, because after that point I really decided, like in this Stack Overflow survey, I decided I want to continue using Rust. And I have been doing this now for – I have to calculate – seven years I also decided I also want to use Rust in my daily business. And I think this is the story, this is why we are here today, because there are developers at SAP, and the numbers

are growing, who really fell in love with the language, who see the potential of the language and who see: "This is not only for me, this is also for the company, for everybody Without sounding pathetic, but I, I really had the impression that I needed to do something, and that's why I founded, together with André and other colleagues that I should mention, we first were in the coffee corners chatting about Rust and saying how nice it would be to use Rust in our daily jobs and stuff like that, right. So, in the meantime, we have a community.

Phil: But to get back to your question, Karsten, you asked me as a developer. And so, I have been asking myself for more than four years now: why is Rust so different, why do I feel different, why can I say that I "love Rust". And I think the answer is that - maybe you remember yourself - in your life when there was something hard, when you had to fight for something, when you had to work for something to get it done, and it's really hard work. But you get it done at the end and you feel that you have made an increment, a big increment. So, you grew from what you had to go through. And for me, it's similar with Rust. Whenever I have to fight the compiler again because I really conceptually misunderstood something, the compiler won't let you. So, I could have written the same code in C and C++, maybe, and just compiled it and maybe it would have worked. Maybe I shipped it to the customer, and seven years later there is a bug in exactly this point in my code where the compiler did not yell at me. Rust would yell at you, would say, "No, I won't allow you to shoot yourself in the foot." And if you are an experienced programmer like me in C and C++, you might look at this message of this compiler and think: "Just wait a second. What is wrong here? Did I misunderstand something?" And then at the end, when you finally understand, okay, this is the reason why I really misunderstood this the whole time; then you grew.

Karsten: Okay, so what you're saying is what I always say about school, like from a certain point you only learn from the strict teachers. Right.

Phil: Yes, exactly.

Karsten: Okay, so Rust is the strict teacher. I just had this thought: do you have to have a certain personality to fall in love with Rust? You, as an experienced developer?

Phil: Yeah.

Karsten: Because the other compilers do not teach you anything anymore, or maybe you have to be a little masochistic to like being yelled at by the compiler for the first couple of months. Or is it that, I don't know, you're especially keen on quality, because the compiler already does 75 percent of what QM usually does?

Andre: Well, I would say, it's like becoming a parent, right. Not only that you feel right for it, it is just a decision. And when you are working with Rust and the compiler, it sometimes feels like raising a child: You'll be shouted at and whatever. But when your child is smiling back at you, then everything is fine again. And that's kind of how it works with the Rust compiler. So, it is shouting at you, it is crying, it is not doing what you want, just like a child. But once you compile and it runs it is like this child is smiling back at you, and then everything is fine. You feel happy and confident that your work has been honored to some extent, so you don't need to be born to use Rust, but you just need to make this decision and say, okay, I'm now ready to start something new. I'm curious, I want to try something out, and I don't fear the compiler shouting at me.

Karsten: Okay, now we have Rust – it must really be quite the miracle language there, because we have it in both roles now. We have it as the sergeant major who's yelling at you and the child who's being yelled at. No, anyway, André, what was your first contact with it? We had Philipp's story. Now, what was yours?

Andre: Well, that's also kind of a nice story, I would say, because it was also three or four years back when I started to do some bare metal Raspberry Pi operating system development in my spare time. And as those usually start maybe in Assembler or some higher level programming language like C, I started with C, to be honest, I rumbled with the several issues there. So, I started, very soon, to ask why my operating system was getting complex with different memory issues, so to speak. And it is really hard on a bare metal system to really dig into those bugs because there's no operating system already there. There's no debugging tool. There's no possibility to step your code line by line and see where the issue to tackle that down is. What also adds to the complexity is that the Raspberry Pi is a four-core small PC, so to speak. So, it's quite attractive to do parallel and concurrent programming there, which was my aim, so to speak. And then during an internal SAP event a colleague passed by and saw me complaining and kind of crying about my issues. I wasn't able to really get to the root cause and he said:

"Have you tried Rust?" And I said: "Hm, Rust? What's that?" And he told me it was a new language that promises you wouldn't have any memory issues so easily in your code. And then I tried it and then that started the same story as Philipp's. So I really fell in love at first sight and thought it is hard to get the code running to build your operating system with Rust, but once it has compiled, all the kind of issues I had previously without knowing where the core was running into any issues because my memory is corrupted – they were completely gone. And then it really started to be fun and it really kicked me to further and further provide crates and tools for the Raspberry Pi bare metal environments in Rust. And that's really fun.

Karsten: So, you both fell – not completely immediately, but when you first got past the compiler for your first time – you fell in love with Rust?

Phil: Exactly.

Karsten: With you both being in love with Rust, now, what are the initiatives within SAP that are already happening? Or is there anything, is it just kind of getting people onto Rust or is anything already being developed in Rust and so on?

Phil: Yeah, maybe I'll take this question. As I said already, it is a grassroots thing at SAP. We as developers discovered this is a treasure. and that we should make it available to more colleagues. And therefore, starting out of the coffee corners, we really kicked off an internal meet-up which has now been held for some for some years. I would say it's been two years already that we have been meeting internally and more and more colleagues are joining in. And out of that initiative, internal tools have been built that we use for our compiler architecture, the CI/CD landscape, and stuff like that. So there are already things that we do at SAP with Rust. But it has not reached, I would say, a business level. We do not have, I would say, a visibility in management areas with Rust, And I think is okay and this is good because otherwise it would maybe lose its Open Source character and its steam and its soul that it currently has. We have a community now at SAP that is tailored to Rust and that would really like to bring Rust forward. And maybe I should mention that also André and other colleagues who are joining me on this on this endeavor have done a great job so far of really educating colleagues and spreading the word about Rust.

Karsten: Okay, thanks.

Andre: Thanks for that. The only thing I would like to add here is really that using a new programming language in an enterprise context also comes with its own challenges. So, we are also currently evaluating the enterprise readiness, so to speak, of the language and of the tools and how we can meet the high quality standards and product standards we are obliged to adhere to within SAP to really ensure that the software we are delivering to our customers is in the best possible quality,

Karsten: the lucky thing is, from my experience, that sometimes even with the size of SAP, these grassroot things still work. I mean, look at, say, SAP HANA; that was not in the first place a management idea. That was some indexing engine, people meeting with some database engine people and having wild fantasies. Seriously. Now, anyway, you talked about enterprise readiness. Is it there? André, would you say it is or is there a still a ways to go?

Andre: Well, I would say there are some ways to go, right? I mean, Rust starts, as Philipp also mentioned, quite at the beginning, it's kind of a system programming language, which is kind of low level still. But as there is a huge Open Source community around Rust and the crates are evolving tremendously, there's a number like more than fifty thousand crates that are already published at Crates.io, and they're all available as Open Source and they are building on top of each other. So, if someone comes with the kind of low-level crate, then the next one might have an idea to build on top of that. So, we are moving further and further, climbing the hill, so to speak, to get more and more high-level crates to be used in enterprise contexts, for example. And there are initiatives like "Are we game yet" or are we web yet, which are driving those things to get more and more high-level attention or attraction to Rust, like building games, 3D games, 2D games, whatever, purely in Rust or developing GUIs which are transferable or let's say interchangeable, depending on the platform or platform agnostic, so to speak, and all these things. And that's also seen at other companies like Amazon, which provides an Amazon Web Service SDK crate, currently in an alpha or beta release, if I'm not mistaken, completely written in Rust, which addresses the Rust community. And recently there was the Rust Foundation, founded by big players like Microsoft, Google, Amazon, Facebook and Mozilla, which are platinum members. Looking into the glass ball, I'm pretty sure they will also drive the language further, which is kind of the logical

evolution of the language to receive more high-level crates and to ensure enterprise readiness.

Karsten: Okay. Sounds like the big ones are already in there, and that is, of course, actually in the end the guarantee, maybe even more of a guarantee than just being cool, when the big ones get interested. I guess I'll have to, I think, talk about Rust with a guest I'll be having sometime in the next month. He was part of the Watcom compiler developers. That was a C++ compiler.

Karsten: The cool thing is that we have an SAP colleague now who was like back in the 70s or 80s or something already developing on that compiler. And that compiler compiled all the cool games of the 90s; it just reminded me of it, as you said, "Are we game yet?" And that compiler compiled like Doom, Quake, Descent, Tomb Raider, you name it. So, I'll talk to this guy in another podcast. it just reminded me of that when you said: "Are we game yet?" Now we're basically, through with the most important things we wanted to share about Rust. So, why don't you try and summarize it for our listeners. What are your three key takeaways that you would like everyone to remember from this?

Phil: I would make it one. Use Rust, it's awesome! Now, really, when you think of developers, we do our work and we need tools, and who would build a house without professional tools? And to me, Rust is a very professional tool. It's a tool that can make your lives easier. It has batteries included. It is really designed well. And a lot of big players are using it. I think we should not ignore when there's something valuable for other developers and we should try it. So just try Rust! That's my message to you.

Karsten: Thank you very much, André and Philipp, for being our guests today. It was nice to have you here. This is our last broadcast before our summer break. Our next episode after this one will be on September 29th, as usual the last Wednesday of the month. Thank you all for listening to the Open Source Way. If you enjoyed this episode, please share and don't miss our next one. As I said, on September 29th then. You'll find us on openSAP and all regular other podcast distributions like Apple Podcasts, Spotify, and the like. Thanks again, Phil. Thanks again, André. Let's all say goodbye. Bye bye.

Andre: Thanks for having us.

Phil: Thanks, bye bye.