What killed Haskell, could kill Rust, too

What_killed_Haskell_could_kill_Rust.md

At the beginning of 2030, I found this essay in my archives. From what I know today, I think it was very insightful at the moment of writing. And I feel it should be published because it can teach us, Rust developers, how to prevent that sad story from happening again.

What killed Haskell, could kill Rust, too

What killed Haskell, could kill Rust, too. Why would I even mention Haskell in this context? Well, Haskell and Rust are deeply related. Not because Rust is Haskell without HKTs. (Some of you know what that means, and the rest of you will wonder for a very long time). Much of the style of Rust is similar in many ways to the style of Haskell. In some sense Rust is a reincarnation of Haskell, with a little bit of C-ish like syntax, a very small amount.

Is Haskell dead?

There was a time when Haskell was the language to watch. During the late 2000s through the 2010s, Haskell was the language everybody wished they could program in, but nobody did, except for maybe a few people. There were some pretty impressive projects that were done in Haskell. There were massive financial projects done. There were payroll projects that were written in. But perhaps the most impressive from the point of view of a purely functional language like Haskell was Pandoc which had a Haskell core. It slapped in the face the whole notion “Haskell is too slow”, or “Haskell just can’t do real things”. Of course it can, and it did.

What happened? Why did Haskell suddenly, pretty suddenly just stop? It’s certainly not alive now. No one is contemplating major projects in Haskell any longer. Who is a GHC Haskell programmer here? I presume there are one or two people with their hands in the air. The GHC Haskell dialect has kinda fallen into an academic language. No one was really considering anything serious.

Haskell in its time was at the forefront of functional programming. It was the language that kind of characterized or epitomized what FP was really about. There were other languages that were functional, but they were only sort of functional. And of course, I'm talking about languages like Scala, or JavaScript. Those were the two major competitors during the same mid 2000s. Scala, JavaScript, and eventually, Golang, and C++ were followers. Haskell led them. These languages learned a lot from Haskell. Anybody doing Scala now knows that the syntax of the for-comprehension and many of the libraries, come directly from the Haskell world.

Haskell ruled.

It ruled in a way that no other language could have at that time. It ruled technically. There was a productivity which is measured at perhaps a factor of five. A team of developers could get an application written and shipped five times faster than a Scala application, or a C++ application. A factor of five is fairly significant in our world.

Haskell also ruled in ways that were just beginning to grasp. How many of you are using monads? But I'm using monads in JavaScript, and I have a little bit of a monads in Rust. In Go, I can do amazing things with monads. The thing about that is that the Haskell people had that in mid 2000s. There were monads and algebraic data types that were immensely powerful long before most of us even thought about a monad.

Haskell ruled in a whole bunch of interesting ways and yet it died. What killed it?

I'm gonna use a word here, and I don't want you to take the word the wrong way. The wrong way you could take the word is "evil", and the other way you could take the word is "ignorant". But it's not quite even "ignorant". The word I'm gonna use is "arrogance".

There was an arrogance in the Haskell community. Not the evil kind, but the kind that told them that they were somehow better. That the tools they were using were somehow better. That the things they were doing were somehow better. There was the arrogance of those people who believed that victory was inevitable. This was not the slapping your face "you, stupid fool golang programmers" kind of arrogance, although there was plenty of that, too. Instead, it was a kind of arrogance of power. Because the Haskell people were writing a pretty powerful code, they did have a tiger by the tail. It was a powerful compiler, it was a powerful language, and they knew they could work miracles.

And yet, that wasn't enough. Something insidious, something subtle happened. It caused their separation, they set aside the rest of the industry. The people outside the community who were writing everyday programs began to look at the corner of the eye where the Haskell people were doing: "Emm… Haskell people don't seem to like us very much, I don't think we're gonna like them".

Some of you might remember the Reddit discussions in the mid 2000s. A bunch of people were there. And they were talking about cool math things there. In those talks, they often were snickering about other languages like Go. It wasn't anything significant, it wasn't anything evil, they were just snickering: "He-he-he, mainstream people, ha!". But I was a mainstream golang guy at that time! I didn't like that. And I've been dealing with language wars in the next couple of years. And I said to them at that time "Do we really want to have language wars on Reddit?". And the interesting thing about it was not about what they were snickering about, because they probably had a right to do that. What was interesting about is my reaction. My reaction was defensive. My reaction was "Well, you guys, go ahead and do your Haskell thing, but I'm the one who gets real work done."

That's the interesting division that got set up at the time. And it was fairly pervasive. There was an attitude among the Haskell community, and again, it's not an evil attitude, not one that was born out of ill will. But there was an attitude that said "You know, our tools are so good, our language is so good, we don't need to follow the rules. We can do something else. We don't have to talk to other people. We don't have to do the other kinds of programs." Haskell people didn't want to do the regular kinds of programs. They didn't want to have to deal with the corporate database. They didn't want to have to deal with the horrible schema that had evolved twenty years. It was just distasteful. And they found ways instead to do things like using category theory, and dependent types. They've built a wall around themselves, and they've chosen to live in a technological bubble. Isolated from the evils of the outside world.

I'm going to define a word here. It's a word you all know. And this definition is just one of many. You can find other definitions of this word if you like. The word is "professionalism". And I'm going to define it as "The discipline of the wielding of power". We have a certain amount of power in our tools, in our languages. But it requires a discipline to wield that power. And it's not just a discipline in the use of the tool, it's a discipline in a relationship to the community at large. It is a discipline that says: yes, it is a powerful tool, but powerful tools kill very quickly. And they kill in surprising ways, so we're going to be careful. And we're not going to denigrate people who are a little bit less willing to use our smart powerful tools.

Let us redefine "progress" to mean: "Just because we can do a thing it does not necessarily follow that we must do that thing".

So what killed Haskell is the parochialism, the inability to address the needs of the Enterprise.

Haskell was a stellar performer in certain constraint circumstances but it was limited in its ability or rather in a desire of its users to address the general problems of the Enterprise. There was a certain purity among those people. They didn't want to step outside and so lead themselves in the soil of real work. There was an "us versus them" feeling of uncleanliness, and those of us on the other side of that boundary felt it palpably. Parochialism is an "Hey you" attitude. It's a way of putting a big banner on the screen saying "Hey you, I'm gonna do my way and screw the rest of you". It's a way to say "We are great in our little domain, and the rest of the world can go to hell".

What is my save?

I want to save Rust and all other wonderful works that are going on in this community from that same demise. Frankly, I don't think it's anywhere near going down that route. First of all, the community is I think more dynamic and larger, and I believe that there is no longer the antithesis "us versus them". Those of us who are "strong C++ hormonal programmers" have relented. And everybody is looking around and thinking: "You know, there might be something to this Rust stuff." But still, what is it? What can save Rust from going down the same path that Haskell went?

I'd name 3 things:

The first one is discipline. Discipline specifically in documentation. A technical discipline that can keep Lord Cunningham's problem from happening. It is writing documentation. And by god, it is a hard thing to do. You just want to write the code. But you should write that damn documentation. I mean all of you have found how hard it is to sit down and say "I'm going to write a good documentation so that others could use my work easily".

The professionalism of humility is something that may prevent the demise, the same kind of demise that occurred for Haskell. The "us versus them" attitude. No, I know there have been some funny advertisements like "Mac versus PC" thing. "I'm Rails, and I'm Java". And I don't think there is any harm in it unless you take it too seriously. Unless you build the wall. Or unless they build the wall in response.

And the last thing would be acceptance of solving the "dirty" problems. Solving the problems of the horrible schema. And we're gonna have to sit down and say "No, we'll deal with that". If we're going to survive in the end, we have to address problems that everybody has. Otherwise someone else will address those problems.

Remember the fate of probably the most powerful and influential language of the 2000s. The language that was at the start of pure functional paradigm, the language that influenced so much of what we're currently doing. The fate of that language was near oblivion. And the people who used it and loved it had to jump to Scala for a living, and it nearly killed them.

We have great tools in the Rust language. We could kill it by making a mess, we could kill it by being arrogant about it, we could kill it by ignoring the Enterprise. I suggest that we not follow that route.

As one might have guessed, this is not an essay. It's a transcript of the following talk by R. Martin with some substitutions made (SmallTalk -> Haskell, Ruby -> Rust, and others). You are free to make any conclusions from this.

What killed Smalltalk, could kill Ruby, too

Load earlier comments...

dunrix commented on Sep 23, 2020

I've not met too much arrogance in haskell community, resp. nothing standing out of usualness. The problem may lay in dichotomy between academic and enterprise demands and, as already mentioned, lack of leadership with a clear vision and compatability/consistency/constraints policies. The wild-nature of development of referential implementation with generating permutations of Haskell-like languages with fast-growing and often overlaying/duplicating/conflicting/deprecating extensions undermines its adoption for sure. The trials-and-errors at fixing some sort-sighted decisions brought to the language (records, class deriving, existential ranks etc) were also too much.

With increasing favour and enthusiasm of Rust even at haskellers much more experienced and skilful then I've ever been, one may question if ideas which keep him in haskell world are really worth it ...

srid commented on Sep 28, 2020

https://www.reddit.com/r/haskell/comments/j0w6pm/haskell_is_not_dying/

timmyjose commented on Sep 28, 2020

> Good article. I spent a career in Java and ended with Go. I have been very happy using Go in my projects and it works well for me. I don't miss Java at all.

Not even generics? Heh.

lisanhu commented on Oct 1, 2020

I didn't understand what you want to say. You are saying Haskell died because of some reasons and Rust community doesn't have the problem, and you are trying to save Rust community for some problem that doesn't exist yet. Your "save" doesn't seem like to be the solution for the Haskell problem, but for something else.

You could just say what is your conclusion about Haskell and remind Rust community not doing the same.

It seems that you are suggesting the Rust community to take care of the "mainstream" ideas, and be polite, is it your points?

I think there are many things that are disputed like is Haskell dead? How do you define its death? Not being discussed by certain number of people?

I believe people are told that all languages work and it's just some languages are better suited for some tasks. Like it's really hard to use pure Java to do HPC and it's not easy to use C to write a quick prototype. Rust can do things fast and is usually free of memory bugs, but it's still having severe problems like hard to implement certain data structures like linked list, lack of accelerator support so very hard to program on GPU. I don't think the people you are referring to can really represent the community since the people talking about "superior" languages seems to be new to programming.

robinhoodhimself commented on Oct 22, 2020

> I think there are plenty of programmers with brain power far in excess of what someone needs to grok Haskell or any such un-typical programming languages and paradigms. What can spur them towards using Haskell are these three promises -

> ```
> 1. Can I develop large programs in Haskell in super quick time ? Compared to lets say Python ?
>
> 2. Can the program I develop have great runtime speed ? Compared to lets say C++ ?
>
> 3. Can Haskell let me develop multi-threaded programs smoothly and painlessly ? the kind of
> multh-threading that gets really hairy in say C++ ?
> ```
>
> I get a feeling that Haskellers tend to focus disproportionately on higher level patterns, structures and stuff that programmers tend to invent themselves when they are up against a problem that needs it.

Totally agree with you.

A Development language is a tool. It's not art. Caterpillar is not working in the art department.

wongjiahau commented on Oct 31, 2020

I think it's perfectly normal that Haskell went down this route, in fact I remember Wadler , one of the creator of Haskell said it's weird that Haskell actually went from academia into production, because Haskell's goals were mostly about researching type system and programming language.

bapcyk commented on Nov 8, 2020

Development is Haskell is about... OK, a functionality written in Python for ~2-3 days, in Haskell it will be ~2-3 weeks. Or it will never happen. Because no such a library :) Haskell has a ton of libraries masturbation of a type-level, and 0 (Z-E-R-O) enterprise ready libraries for REAL TASKS. The biggest problem in Haskell community is that the most Haskell developers... are not developers at all - they have not good programming background, very often Haskell is their first language. And this is a very big problem.

codygman commented on Dec 12, 2020 • edited

> The biggest problem in Haskell community is that the most Haskell developers... are not developers at all - they have not good programming background, very often Haskell is their first language. And this is a very big problem.

I find this hard to believe. Nearly no one has Haskell as a first language **@bapcyk**.

deng232 commented on Feb 7, 2021

I think Haskell's tutorials also got a lots of problems, they aren't teach you about tools that will be necessary when you go out to work,(cabal for example, very few tutorials about it, yes, it wasn't hard to use, but for a beginner(not even master OO world yet, just driving by the curiosity of Haskell world), is a big headache. And many important things aren't being mention in tutorials, they just trying to make it look easy, so when you go out to ask question, you don't even sure what are you saying; therefore, it just blocked Haskell's own new blood.

deng232 commented on Feb 7, 2021

What I feel of Haskell's tutorial is they assume you already know a lots of things already, you just want to learn how to use "haskell" .

tonymorris commented on Feb 17, 2021

This is very funny, cheers :)

rpeszek commented on Mar 5, 2021 • edited

Concurrently to this discussion GHC introduced linear types... ;)

I think there are compounding problems that have caused some loss in Haskell popularity including:
limited financial resources, growing jobs in rust, some loss of talent to AI.
I imagine Haskell attracted lots of bright young people who also like math. Now such people are pulled by AI / ML and it is hard to blame them.

Lastly, Haskell was never that popular. It was discussed a lot but there were not that many jobs and most were local.
I am hopeful that this may now change with remote opportunities on a rise.
One interesting resource that rates languages according to job openings is https://www.itjobswatch.co.uk/

Here is Haskell ranking (https://www.itjobswatch.co.uk/jobs/uk/haskell.do)
Last 6 months - 769
2020 - 943
2019 - 902

(no longer shown)
2018 - - 967 (was the year I got Haskell job)
2017 was the year Haskell passed COBOL

Haskell being dead is, hopefully, just not true.

crocket commented on Apr 17, 2021 • edited

Haskell isn't dead. It is the main language used by cardano smart contract cryptocurrency.

Cardano is going to be big. Buy Cardano now.

tarsa commented on May 21, 2021

> A technical discipline that can keep Lord Cunningham's problem from happening.

It's not about Lord Cunningham. It's about https://en.wikipedia.org/wiki/Ward_Cunningham

1082-xyz commented on Aug 31, 2021

One of the largest blockchain projects, Cardano, is built on Haskell

crocket commented on Sep 7, 2021 • edited

Most smart contracts on ADA aren't going to be written in Haskell, by the way. Cardano supports multiple smart contract languages.
Haskell is too difficult for non-programmers. But, it is great for large-scale programs.

hkinke commented on Oct 4, 2021

Haskell changes your way of thinking about programming. Thank you Haskell.

shian15810 commented on Nov 1, 2021

Haskell rewired my brain for sure, and taught me how to write better production codes. Thanks for the great language!

m4dc4p commented on Dec 16, 2021

Dunno where “arrogance” comes from. Certainly can make jokes about it (ruby!) but overall people on the mailing lists or in peer groups I’ve met are interested in writing better programs that do cool stuff, solve real problems, or both.

Regarding the “Enteprise,” don’t forget “Real World Haskell”, “Production Haskell” and others. You can look back even further to see Haskell support for interop with COM (Windows)! Recently, the Haskell Foundation was set up explicitly to address the needs of industry and academic users of Haskell.

I guess Haskell makes a good straw man here as it’s quite alien looking if you come from OOP/Java/C background. But that doesn’t make the comparisons valid at all.

Just a note that what killed Smalltalk is image-based development. What kind of insanity is that? No way it could scale to large teams.

cognivore commented on Dec 17, 2021

Don't feed the troll!

educob commented on Apr 3 • edited

I am right now studing Rust. I have coded in many languages over the years.

I have to say that Rust seems to be, by far, the worst language I have set my eyes on.
I feel it's philosophy is: "Take the worst of every programming language and exclude the best"

Developing a function with the trait horror is making extremely difficult what is simple.

I am studying Rust to test a dapp framework called holochain. Holochain seems interesting but I am having second thoughts about it thanks to having to code it with Rust.

crocket commented on Apr 4 • edited

@educob Haskell is an okay natively compiled language. While haskell improves reliability, its type system is complex and cumbersome. Zig seems like a better C. I'm learning Raku. It has a lot of things, but it seems like a better script language than python and ruby and perl.

Raku is still somewhat slow, but it has become a lot faster and can become a lot faster in a decade.

If you want a practical language, raku is a good choice.

HallofFamer commented on Apr 20

Interesting essay, it does resemble the 'What killed Smalltalk could kill Ruby too' talk from Uncle Bob. Theres one major difference though. Smalltalk was once a mainstream language widely used in the industry, it declined in 1990s and was effectively 'killed' after Java became a hit. Haskell was never killed, it was never a mainstream language, it just 'avoided success at all costs'.

revskill10 commented last month • edited

Not quite. Most of python dev i know has no passion in computer science and software engineering. They just want to deliver quick, dirty programs which contain hidden bugs that they don't even care to fix before giving it to latter programmers.

That's the issue with "outside world", not Haskell issue. What you call "enterprise" is often just a bunch of dirty works with hidden bugs (to keep the deadlines of course). The issue, is , no one cares much the details to clean the tech debts.

Haskell people just know and reveal the truth.

TLDR is: Do things the right way, not do the right things the wrong ways.

hkinke commented last month • edited

Since Google decided to use Rust in low level Android programming, Rust is now in the top 20 of Tiobe ranking.

bapcyk commented last month

> Since Google decided to use Rust in low level Android programming, Rust is now in the top 20 of Tiobe ranking.

yes, 20th position :)

bapcyk commented last month

> Interesting essay, it does resemble the 'What killed Smalltalk could kill Ruby too' talk from Uncle Bob. Theres one major difference though. Smalltalk was once a mainstream language widely used in the industry, it declined in 1990s and was effectively 'killed' after Java became a hit. Haskell was never killed, it was never a mainstream language, it just 'avoided success at all costs'.

good point.

bapcyk commented last month

> **@educob** Haskell is an okay natively compiled language.

not very okay to be honest - super slowly. Maybe the most slow compiler in the world.

educob commented last month

I have actually changed my mind about Rust.
The frist book I read about it was really bad (at least for me) and I got a very negative view about Rust.
But then I studied a second book and now I understood the why of all "weird" stuff and it makes more sense.
Anyway a difficult language.

zerosign commented 1 hour ago • edited

hmm, Could you clarify a bit about ignoring `Enterprise` in here ? I though some of the " `Enterprise` " already using Rust in solving `system programming focused` problems and I think `Haskell` are also aren't being killed in here, it's just much more focused ? cmiiw