# Rafael Castro Gonçalves Silva

*Cover Letter*

I'm a working computer scientist and a developer.

## Prelude

This is text about myself, so you can know a little bit about me. I did this text in an informal style of writing, so you can imagine me saying what is written here.

## How I'm like

Usually I describe myself as nerd, progressist and cachorrista (a brazilian neologism for dog person). Let me be more detailed about those things:
I enjoy nerd stuff like video games, fantasy movies and technology in general. I know, very typical. Second, by progressist I mean someone that agrees with progressism: the idea we should embrace changes that can cause improvements in our life quality (e.g., weed legalization). And last: I like dogs a lot.

Of course, I'm not just three things.

Another thing that I'm is playful and I'm far from the stereotype of a serious adult. Sure I can have serious conversations, but I just find life better to live if I can joke every now and then. Though I enjoy making jokes, I don't like it when it offends someone that did nothing to deserve it I hate when I see someone being humiliated if the person don't really deserve that. I can't watch *The Office* because I fell like throwing up when I see Michael humiliating his employees that have done nothing to deserve it. Search for *The Office — Try My Cookie Cookie* on YouTube and you will know what I'm talking about.

More on me:

I usually avoid things that cause me stress. If I have the option to spend money so I won't be bothered, I do it. Sometimes I just quick the thing that is causing me pain or anguish. I don't mean that I'm a quitter, I just question myself: do I need this? And depending on the answer I do benevolent thing to myself. For example, I disliked my first job in the first few weeks, so I quitted after just two months working there. It was there right call, so I could move on to the next thing.

I'm eager to learn and learning is quite important to me. This characteristic of mine made me want to avoid to work in the software industry. My vision was partially correct I would say: to a company the best tools are the ones that you can easily find employees to hire for. But, usually, those tools are not really the best ones for the job and are the ones that I don't want to learn. I learned Ruby for my job, but don't think I was an interesting thing to learn. However, I have learned very interesting things in my current job. So I don't see working in the industry with the same

pessimist eyes as before.

But my opinion, in general, stays the same: developers should be eager to learn the best technology for this or that specific problem. Think the problem is lives on the fact that programmers most of the time are stuck with the same languages and tools, they should more often look out there and see that languages aren't all the same, and they didn't stop innovating in the 90s. In conclusion about that, I rather work in a place where I'm continuously learning interesting things. There are some very interesting new languages like Rust that I wish to see in the industry more often. However, I understand that you don't see a Rust programmer under every rock like JavaScript programmers.

I'm usually very communicative and I don't fill ashamed of asking people to teach me about something. I don't pretend to know something that I don't. I enjoy working with teams that do *real team work* with members support each other and support each other.

## My life in a nutshell

I was borne in 1993 in the city of São Paulo (Brazil), but when I was three years old I moved to the State of Santa Catarina (SC), in the south of Brazil and I have been living here since then. I lived in some cities of SC, but at the moment I live in Joinville. In 2012, I started my undergrad in Computer Science, here in Joinville, at the Santa Catarina State University. For the first time in my life I noticed that I was doing something that I enjoyed. I loved the courses of Automata Theory, Compilers, Formal Methods and Computability Theory. I guess I discovered myself in theoretical computer science. After my undergrad, I started my masters at this same university.

## Academic me

At the end of my undergrad I discovered that programs can be certified in respect to a specification using proofs. I already knew Curry-Howard correspondence, but when I found proof assistants like Coq and Isabelle, and languages with dependend types like Agda and Idris, I was very surprised to see what people were doing with it. For example, I found out CompCert, a C compiler with a fully verified backend. Come on, this is the coolest piece of software ever made! I started the masters and my advisor had experience with type systems for programming languages, specifically type inference algorithms. So I decided that my masters would be using a proof assistant and would have something related to type inference.

Long story short: I did a Coq certification of algorithm W using Hoare logic using a monad. Google *Monadic W in Coq* and you will find the paper.

After the finishing my masters, I decided to look for a PhD, but things didn't go well so far in this matter. In addition, a pandemic started so this plan is delayed.

## Professional me

I had just finished my masters, I couldn't start the PhD and I had no professional experience. I decided that the least I could do is to solve the latter. Looking for a job was quite sad because I don't really see many jobs opportunities that I find interesting. I managed to find a job that looked quite convenient for logistic reasons and I decided to take it. As I said, I didn't liked it in the first few weeks. The problem was so many, just to get you a glance: it was a startup following this stupid trend of startups that think to be a family. More on that latter.

So I quitted that job and started looking for another one. I found a place that some colleagues from the university were working at and they said this was a good place to work. The company is a

outsourcing one, they provide they services as teams of programmers. So, I applied and I was hired. This company is called Magrathea. Yes, the planet that makes planets.

I was allocated to team working in project that is quite interesting. Being brief: it's a distributed system with concurrency scenarios that must process events in whatever order, so it can keep a materialized view correctly updated. At the moment I'm writing this text I'm still working in this project and the experience so far is great. I find very satisfactory to with a team of people that have a higher level of abstract thinking and can write non-trivial algorithms. Working with them made me discovery how it's like to work with team that help each other in order to kill a dragon. Now I can say that I can be a good professional.

Me and my teammates joke that after this project we don't know what to do because everything else looks quite boring. So, if you are a employer reading this text looking for to hire me, it's better you have something fun to offer me.

I know I don't have much experience as a developer, but now I at least see the possibility of an interesting professional life for me. Though I still wish to do a PhD, I don't see this is an issue for my professional life. If you do, you missed the chance of stop reading this text long ago.

## Things that I like

I like challenges in programming! But not *fake challenges* like understanding frameworks, though this is perhaps one of the most valued things in the industry nowadays. Real challenges in programming usually require you to be creative and *educate yourself* about the problem. There is no silver bullet, you need to learn about the thing that you are solving, you need to read papers about it, you need to read the documentation of the tool that you are using. It doesn't matter which framework you are using if the problem is NP-Hard or if the data size is so huge that SQL databases can't handle it. Legal

## The last cool thing that I did as a job

TODO

## Things that I like and care, but also things that I don't

People should notice that what makes someone a good programmer is to understand Computing Science foundations. Yes, tools are important to be more efficient and so on. And engineering is about using those tools and the resources that you have to solve some problem. But a tool just *stays* important for some time, eventually some other new tool appears to take its places. So in this section, I will only talk about ideas and values that I like and car about, but not the tools that I use. See my CV to know about the tools that I use. I will also mention some general concepts presented in some tools here and give some examples, but I don't mean rant agaist any specific framework or language.

Tests are not the ultimate tool in software quality, but it is the thing that the industry has widely adopted. Before ranting about tests like Dijkstra, anyone should at least acknowledge the different kinds of tests that exists and understand the limitations of each. In my experience, integration tests can be extremely helpful in discovering connor cases of the behavior of the system. Though, unit tests is just crap made necessary by dynamic typed languages. Useful test are about general and connor cases behavior, not if the code is glued nicely. Lets make tests about behavior, not interface connections.

I think we should not forget other styles of quality insurance - I mean, there is a huge and diverse

academic research field about that. If you are dealing with a distributed system why not model it in a model checker? And, please, chose static and strongly typed languages, and go beyond unit tests. Nowdays, the academy is putting its verified code in airplanes and in generators for nuclear power plants. This kind of knowledge shouldn't be restrict to a few nerds at INRIA. All programmers should at least know what is Formal Methods.
I think that software development should follow some basic "practical software engineering", like CI/CD, code review, git flow and so on. Communication in software development is also very important, so we should always make clear "why some piece of code exists", other questions usually are easy to answer by just reading the thing. In my opinion, understanding the goal of some function is more important than understanding how it works. A function should have a single and clear goal. A single function shouldn't solve everything.
It looks like that the Agile Manifesto is misunderstood or people are forgetting its real meaning. Somehow people are selling "agile" as a coach service. That is bullshit! The Agile Manifesto is about values: "we value individuals and interactions over processes and tools". How can you buy professional values for yourself? Or you understand and have those values or you don't. Okay, you could pay someone to teach you about those values, but is your decision to follow it. The decision of having some values is an individual one. You can't expect a team to behave "agile" because the team leader now says that they will start to use "agile", and they usually do that by enforcing some ritual (just like a cult). The decision of being "agile" is individual, not a team decision. My decision is to be "agile", but don't expect me to like your rituals.
organizational silver bullet of the moment. "The employees will be more motivated and they will produce more". Really? How motivated? At least with me this **obsession** is annoying and it unmotivates me. So I may work in a company with values, but not in a place where this is systematically repeated to the employees.

## Rant section

This is the part of my cover latter that I stop caring if I'm being unpleasant.

My thoughts on Software Engineering in practice

Range agaist Object Orientation

Range agaist companies bullshits