# Actuary Versus Programmer

Zero Contradictions

October 18, 2023

## Contents

## 1 Why I Think I'm Better Off Being An Actuary

I never had a good understanding of what actuaries do until I was 20 years old, and I didn't start to seriously consider becoming one until I finally realized how unsuited I am for software development. The main reason why I am writing this file is so that I can remember for future reference why I eventually decided that I'm better off being an actuary instead of a software engineer.

- Surveys show that actuaries tend to have better work-life balance, higher job satisfaction, and better job security during recessions than software developers do. This is important to me since I'm interested in having a family life, and time to think and write about any interesting philosophy/knowledge that my mind craves.

- An ideal job would have me doing analytical work where I get to do deep-thinking and make big decisions that affect the direction that the company takes.

  - For this reason, becoming a CAS actuary might be the better choice. I've anecdotally heard that their higher than average salaries (compared to SOA actuaries) could be arguably attributed to how they arguably tend to play more decisive role in their company's decision making and profits.

- Knowing that my job plays an important role in insurance (and thus also the world's economy and the general functioning of society) is more appealing to me.

- On the other hand, being a drone who does what he's told for a subject that he has little interest in is less appealing to me. It seems to me that this is more likely to be the case for software engineering and code monkey jobs.

- It would be fun for me to get paid to do calculus, linear algebra, differential equations, statistics, probability distributions, etc for my job.

- Some jobs will pay actuaries to study for their actuarial exams. It's attractive for me to choose a job that pays me to learn more mathematics (and financial/economics related stuff). Learning new things is my favorite hobby to do.
  - It would it make it easier for me to study the Black-Scholes stochastic partial differential equations for making better investments. I would definitely be closer to understanding them on my own if I approach them after familiarizing myself with the content from exams P, FM, and IFM.

- The financial knowledge that I attain from studying for the exams and doing actuarial tasks may enable me to make better, wiser investments on the stock market, thus increasing my income, retirement funds, and ability to be a breadwinner for my (future) children and family. It may also enable me to write more original blog posts and content for my website, since I would become more knowledgeable about insurance, economics, mathematics, and statistics.

- Another (bonus) benefit to me choosing to be an actuary instead of a programmer is that the Frogstar B discord server already has a lot of people that work as software engineers, so it will probably benefit our intellectual movement if we have more people who work in different fields, including someone who will be knowledgeable about insurance, statistics, and financial related stuff to help increase the variety of conversation, topics, and knowledge flowing within our online community.

- I could still do programming as a side hobby, but this time, it would be specifically focused on doing projects that I am passionate about creating (e.g. a program for modeling phonotactics, a python bot for processing spreadsheet data and automating the creation and (improved) editing of wiktionary/wikipedia entries, debate software, Emacs Lisp?, etc). I have so many ideas for programming projects, but I don't like the idea of charging money for people to use them, and I also don't think they'd generate a lot of money commercially or via donations.

- Since being an actuary will require working with data, and since the primary advantage of Lisp is its ability to blend code and data together, being an actuary may give me a great excuse for making time to learn Lisp in further detail, whether that be to write programs that I've been dying to write in Emacs, or to write better programs related to my job, instead of Visual Basic (VBA), which is among the world's most disliked programming languages.

- Even if I somehow don't enjoy the work that I do as an actuary, it would still make more sense to do it anyway, compared to working as a programmer / software engineer, if I'm simply better at mathematics and finances than computer science.

## 2 Reasons Why I Don't Want To Be A Software Engineer Anymore

### 2.1 I'm not passionate about it.

- I wouldn't necessarily be programming stuff that I want to program. Instead, I would be programming whatever my boss assigns me to do. In other cases, I would just be debugging and fixing software issues from bug reports or ticket issues.
- I doubt that I'll ever find programming to be as enjoyable as mathematics or linguistics.
- I doubt that I would be able to enjoy what I'm doing if I don't believe that it would be making a positive difference in the world that aligns with my worldview.
- Debugging is too stressful for me. I don't like reading through lots of stuff that I don't understand.
  - It's said that most software developers spend ~80% of their time debugging, so if I was mostly debugging, that suggests that it would actually be a rather undesirable job for me to do.
  - Other people probably don't have this problem, but I'm often afraid to ask for help online, since it often feels like that I could just solve with problems with a bit more effort. Other times, I'm afraid to ask my questions because Q&A websites have policies against people asking duplicate questions, and I'm afraid that I might violate them, even if my problems are unique and legitimate.
- I wouldn't enjoy writing specific, sequential algorithms to do tasks, especially if it involves re-inventing the wheel, just like many LeetCode problems do.
  - I don't get a kick out of programming procedural things that could be easily replicated by someone else doing a set of instructions and logical deductions. This is the same reason why I don't like solving puzzles. I used to enjoy that as a kid, but now that activity feels too simple, non-challenging, and time-wasting for me to enjoy it.
  - If it's necessary to "re-invent the wheel" but differently, I feel some value to doing that, but it still feels like there's more meaningful activities to do in this world.
  - Perhaps this all indicates that I just don't find programming to be exciting enough, in most cases.
- It's possible that I'm too easily distracted to focus on coding, when I have to do it. I've never had this problem with math for some reason. Whenever I had to do math classes and CS classes in the same semester in college, I always felt more motivated to do the math courses first.
- It seems that I only like the idea of programming and what it could potentially do, but I don't actually like doing it myself.
- I typically want things to "just work" most or all the time, which may be a strong sign that software development is not right for me.
- I'm not sure why, but I just can't feel perfectly comfortable in any development environment that I use.

  - If I use Emacs, it feels more comfortable with my autizmo tendencies, except that I can't get the keybindings, window displays, automated processes, etc to work exactly how I want them. I've heard anecdotally that Emacs can do everything that Visual Studio can do. However, I can't figure out how to do a lot of features that developers are able to do in more widely used environments like Visual Studio, and it makes it harder to follow coding tutorials.
  - If I use something like Visual Studio, I'm bothered by how I can't do the features that I wish I could do from Emacs. I've heard that there are ways to get Vim keybindings working in Visual Studio, but it just doesn't feel the same.
  - IDEs like Visual Studio also feel too complex for me. Maybe I could learn them if I put in more effort, but I still feel that I'll never be completely comfortable using it since everything doesn't feel "just right".

- Most software engineers are able to get excited about the latest tools and breakthroughs in the field. I just can't feel the same level of excitement.

Part of the problem is that it's difficult for me to be motivated to program things that I'm not particularly passionate about. For various reasons, I feel more motivated to do math about things that I'm not passionate about, compared to programming that I'm not passionate about. Math just feels easier to think about, it feels more practical than programming, it feels more natural, etc.

Part of the problem is that I burned my mind out by trying to learn beyond my natural scope of understanding, I browsed too many Wikipedia articles instead of using tutorials instead, and didn't enter college with enough prior mathematical skill and mastery to blaze through the courses, to the point of being ready to pursue a master's degree in mathematics. Another major problem is that my K-12 education and even my university didn't teach mathematics as effectively as they could have, nor did my university give me enough time to practice and fully grasp the concepts taught, which left me with some holes in my knowledge and abilities. As a result, I don't have as much interest from learning complicated details anymore, compared to my interest from when I was 19-21 years old. I still have the capacity to learn complicated details, but it's just discouraging when I wasn't taught optimally, when my skills are no longer comparable to the brightest math graduates (given my current age), and especially since I don't really have the time to learn things in such great detail since I now plan to dedicate my life to working a job and supporting my (future) family.

### 2.2 I don't want to have to work with poorly written code.

- As a software engineer, I would have to work with other people's code, even if it's poorly written. I'm quite autizmo, so it's difficult for me to tolerate and withstand inefficiency. As an example, it quite irritates that I have to live in a country with car-centric infrastructure. I know that I have to deal with it since that's the way the world around me was designed, but it's still difficult for me to acknowledge and be reminded almost every day/week that I live in a world of great inefficiency, and that everybody suffers the consequences of that.

- I enjoy learning stuff, but for some reason, it feels pointless for me to learn data structures, algorithms, and trivial programming APIs, libraries, etc.

  - At least for me, a lot of programming feels too far removed from doing mathematical thinking. It varies, but sometimes it just feels mundane, either way. I imagine that AI technologies will help remove a lot of the more mundane aspects of programming in the future, but I still can't even feel comfortable using any IDE, so I would have to resolve that problem first.

  - Algorithms usually bore me instead of interesting me. I like problem solving, but it's unappealing for me to look at code and figure out how it works, unless it's something that I'm interested in.
  - A lot of it involves learning and using APIs that could become outdated or otherwise broken in the future. I don't want to be doing things that won't last because that feels pointless.

- Functions and APIs often hide what the code is actually doing behind the scenes, which makes it harder to debug things.
  - To me, this makes a lot of programming and debugging feel analogous to working with problems for which there isn't enough information to solve the actual issue at hand. Problems like those are among the most frustrating, at least in my opinion.
  - If there are bugs inside the functions that I'm using, I don't want to have navigate through layers and layers of code just to find the problem. I'm more okay with reading code that is written continuously from left to right, top to bottom, even if it involves really long functions. That kind of code feels more natural because it's like reading a book.
  - It can even be difficult for someone to work with their own code several years, months, or even weeks after they wrote it themselves. Quality comments and good naming conventions can help with this, but it can still be difficult regardless. I doubt I would enjoy going through the process of thinking something through very deeply, and having to go work my mind through that same process again multiple times. Due to my cerebral wiring, I prefer to do complicated tasks all at once, one step at a time.

- Object-oriented programming is the most dominant programming paradigm in the entire software industry, even though it's usually inefficient, slow, and prone to errors. I would rather not work with that infrastructure if I could avoid it.
  - In hindsight, most of the bugs from my college programming projects/assignments revolved around problems created by OOP.
  - Sometimes OOP can have its advantages, but functional programming and procedural programming are usually superior, easier to read, easier to use, and easier to debug.

- It's not unheard of for software engineers to care less about the quality of their code before they plan to quit and move to another company. Some software engineers move frequently from company to company in order to boost their salary and benefits and whatnot. As far as I know, that's not really a thing for actuaries since they work with data instead of code.

- Although C code tends to avoid OOP, C APIs tend to have cryptic abbreviated names (which reduces my interest in learning them), and C programmers are more likely to not write enough comments in their code.

- Some jobs might require me to use an IDE that's more similar to what everybody else uses? I don't think I could handle doing that since I prefer Emacs.

- Given my personality, I often prefer to work alone rather than working with other people. This would undoubtedly make it difficult for me to work on a software development team. I also tend to do things differently than most people, even among other high IQ people. For instance, I always tend to be more meticulous and conscientious, I'm less likely to abbreviate things, I'm particular about how things are formatted, etc.

- This is probably more of a personal defect, but since I have ADHD, I often have a tendency to just fix one or two things, recompile the code, and hope that it works (even if it's unlikely work).
    - Again, it frustrates me when code doesn't compile correctly. Every programmer will say this, but in my case, it may be true moreso to the point where it makes me not as well suited to be among the better programmers.
    - Sometimes, I also have difficulty reading instructions, particularly ones that aren't written very well. This happens more often when I'm too hyper to focus.
- In all the years that I've spent tinkering with things like Linux, Emacs, and programs that only tech-savvy people would use, I seem to have more difficulty in setting those programs up and configuring settings, compared to people who enjoy programming.
    - I suspect that this may be partially because I never had a person to guide me one-on-one to get through all the hard stuff and barriers (compared to people who had parents/relatives who program for a living, and helped their children follow suit in their professions).
- I take great pride in my work, and I would be quite dissatisfied if my work wasn't as good as it could be since it had to use or incorporate poorly designed things.

### 2.3 Regarding the career process and rewards

- I don't like the idea of coding proprietary software that I wouldn't have any ownership over, especially if it's just to compete and get an edge over other companies.[1] Most jobs only pay programmers to write proprietary software, so the odds are that I would have to make a living by writing proprietary software. There are some jobs that pay to write open-source software, but they are the minority.
- While I technically wouldn't have ownership over what I create as a mathematician or actuary, doing mathematics to solve problems seems more appealing to me because I view one of the great rewards of doing math as being able to solve the problems at hand. I wouldn't get similar satisfaction solving problems as a programmer because I really hate debugging and I wouldn't be able to own the company-owned source code, even if it works perfectly.
- It depends on the problem, but in my experience, math problems have usually seemed to be easier for me to solve.
- Actuaries and software engineers both have high, comparable salaries.
    - Perhaps the highest paid software engineers receive more than the highest paid actuaries, but realistically, I'd probably never get paid that high since I would only be average (probably lower than average) in the rest of the software development field.
    - The salaries vary a lot depending on the fields and locations. Either way, there are strategic decisions that I could do to boost my salary.
    - Even if most actuaries get paid less than many software engineers, it's still better to do an enjoyable job, rather than a job that one despises.

[1] Proprietary software has its place in a world of selfish people and companies. If human nature determines that proprietary software has to the primary business model for most companies, then so be it. But in a more cooperative world, I would hope that it's possible to create efficient economic models that enable the production of open-source software more often. I might write a section about this sometime on my thoughts on economics page.

- Many of the best-paying CS jobs require master's degrees or PhDs. The opportunity cost of attending graduate school would be to borrow more debt and increase the number of years where I'm not working to make more money.
  - By contrast, actuaries only need to take exams and then they're good to go. Most actuaries don't need to go to graduate school to get high salaries.

### 2.4 Advancing technology isn't always a good thing

Although I like most technologies and how they do many useful things for this world, it's difficult for me to get behind a career that propels the world into new technologies that it doesn't have yet. I believe that technology has just as many cons as pros (sometimes even more), so it's difficult for me to commit to being a computer scientist since I'd often feel uncertain about whether what I'm doing will have a positive effect on the world or not. If the net effect of what I'm doing is neutral (or possibly even negative) to society, then I'd feel nihilistic as to why I'm doing it. I would be doing any job that I work for the money of course, but I still wouldn't be able to help or stand that it feels meaningless to me, to the point where I'd rather not do it at all. In general, it's a growing trend for people to want to do work that feels "meaningful" to them, but not everybody will be able to find such a job.

My outlook towards technology is two-fold for AI, since it's still so undiscovered. I don't believe that AI will lead to a malevolent singularity, but I'm still pessimistic about how it will affect society. On the other hand, if I'm doing things with known and predictable results, it could feel pointless, as if I'm reinventing the wheel.

In any case, I think I'd feel more comfortable calculating insurance rates, since that occupation has more predictable effects on society and has existed for decades to centuries before civilization started declining (i.e. it's not a contributing cause to evolutionary mismatch). We can also expect the entire world to be facing major catastrophes in the coming decades due to economic, political, environmental, and resource mismanagement. If I know my occupation could play a role in minimizing the damage caused by such catastrophes (via managing insurance rates), then I would feel better about my chosen field of work.

### 2.5 Regarding the hiring process

I am black-pilled about the job market and interview process for programming and software development related jobs. This wouldn't be much to worry about for someone who is a good programmer, but I'm not good at programming because I don't even enjoy programming.

- Many job interviews require the candidates to dredge through repeated LeetCode problems. These types of questions and job interviews are pretty brutal, especially if you don't enjoy writing algorithms in the first place.
  - Personally, I'd never be able to tolerate doing one LeetCode problem after another, full-time *for weeks* on end. If that's what it took to get a great job, then I'd have to look for a different job.
  - Again, I can't stand feeling inefficiency or meaninglessness. When I do, it makes me hate life. I don't mind preparing for tests, but I have to feel like that I'm personally getting something out of it too, like learning (interesting) stuff that I didn't know before.

- Although there's a much larger job market for software developers, it's also much more competitive and cut-throat. It's not unheard of for people to have to do dozens or even hundreds of job applications and interviews, just to get their foot in the door.
- Since software engineering has a much larger job market, that probably makes it more vulnerable to Goodhart's Law, right?
- Programming jobs are more vulnerable to layoffs during recessions, whereas actuarial jobs are not because the need for insurance never goes away from society.
  - Recessions and the ensuing unemployment are often unexpected. The world will have to deal with this for as long as we have an economic system that enables economic recessions through speculation bubbles and such.
- Any good job candidate needs to do their homework and be prepared to compete against other candidates, but the hiring process for actuaries seems more straight-forward. Actuaries just need to make sure that they've passed the exams, and they're often good to go.
- At my current age, my credentials for getting hired as a software engineer are already behind someone of the same age who has a degree in computer science and has spent years doing actual, independent coding.
  - I'd be at a competitive disadvantage for getting hired as a software engineer or programmer.
  - It's also discouraging whenever I see job ads that require master's degree or PhDs, given that I've never attended graduate school (and probably wouldn't study CS if/when I do attend).

### 2.6 Genetics factors?

My father likes computers, and he did program for a while during his teen years and early 20s. He also failed to finish a computer science minor in college (just like I did, except that I was closer to finishing). After that, he never continued programming simply because it is not one of his main interests. I feel like that I went down a similar route. While programming seemed interesting when I was first introduced to it, I doubt that I would be interested in working with it as my primary occupation, although I wouldn't mind using it to a limited extent, especially if it was for something useful that I'd rather have be automated. If occupations tend to run in families, that suggests that if my father wasn't able to become a successful programmer, perhaps I won't either.

Then again, I am much better at mathematics than my father is. He dropped out of calculus in college, and he never took any higher-level mathematics beyond that, whereas I managed to graduate from college with a bachelor's of science in mathematics. I'm sure that he could learn more advanced mathematics if he would just put in the effort for learning it, but he doesn't have any interest in doing that (or most academic subjects for that matter).

My maternal grandfather was a smart man and he worked as a chemist. However, he was so disinterested in computers that he never used the Internet even once within his lifetime. A lot of this was because he hated change, and he preferred to keep everything the way things used to be. I hate changing things too when I find a tailored comfortable way of doings things that I enjoy, but I'm always willing to transition to something if I perceive that it improves the efficiency of doing something.

## 3 Reasons Why I Don't Want To Work In Academia

When I was 19-20, I thought I could learn everything, and I thought I was going to get a PhD and doing research in Academia. I still believe that I would've had a better shot at attending graduate school at an earlier age if I had more guidance during my childhood and adolescent years, but regardless, I no longer believe that I want to do research for a living. I need to write to this section to remind myself why.

- Most research is fake, and a lot of it is just a status game.
- Most of my time would be spent teaching and grading papers, not researching.
- Even if I did have a lot of time to do a lot of research in a field like mathematics, my IQ may not be high enough to be as competitive against other academics. I may not know unless I try though.
- The odds of gaining a strong reputation as one of society's greatest intellectuals are low, especially if I decide to speak the truth, instead of saying things that fit the status quo and popular opinion.
- While I am proud of some of the works that I've made in college (e.g. my senior thesis), the amount of time and effort that it took to write that is kind of turn off for me. I already spend a lot of time as it is writing stuff for my philosophy website.
- It would pay a lower salary than being an actuary or something else.
- It doesn't have good job security, and it's very difficult to get tenure.
- It would be much more stressful.

## 4 Reasons Why I Thought I Wanted To Be A Software Engineer

- Computer science was described to me as "problem solving", and I love solving problems (and still do), so I thought that I'd love it and would be a natural at it.
- It seemed really exciting to know that I could write my own computer programs and watch the computer execute them exactly as told.

- I enjoyed writing many of the first programs that I wrote in high school and college, but now I'm not so sure that I'd enjoy working with much larger code bases with many other people, using APIs that'd have to be learned and potentially low-quality code.
- I knew that it would be a nice, high-paying career, especially for someone with a personality and high IQ similar to mine.
- It seemed really appealing to think that someone could theoretically have nothing more than a computer (and a smart brain) and start making lots of money if they made one awesome app that everybody really liked enough to buy.
- Although I was intimidated by computers before I started using them everyday during high school, I really started to enjoy them when I was introduced to computer games, MS Office products, programs, etc, and even different operating systems, like Ubuntu and all the other Linux-based operating systems.
- There were many obstacles to me learning how to program and since I always like to have everything be "just right" before doing stuff, I always assumed that if only I could just overcome all the barriers, I could start learning programming really quickly, and start working as a software developer.
- I'm the kind of person who prefers to use Emacs, Linux, Colemak, and privacy-focused / alternative / FOSS software. I even run my own custom-designed website that I wrote and designed almost entirely by myself.
  - I couldn't fathom how someone could enjoy being so geeky and autistic with his computer, and still not want to be a programmer... until I realized all the reasons I described in this paper.
- Since my other major in college was linguistics, I thought that I would be really well-suited towards being a computational linguist, because a person with a Mathematics BS, a Linguistics BA, and a CS minor would probably have near-perfect credentials for being a computational linguist, right?
  - It's kind of a shame too. If I become an actuary instead of a computational linguist, then I will be getting a job that doesn't utilize my linguistics degree like I had planned. I don't regret studying linguistics though.
  - However, linguistics isn't particularly useful to know in computational linguistics these days, unless one plans to specialize in making computer work well with human languages other than English.
    * In which case, this would require that I be fluent in a second language other than English, but the only languages that I have a good shot at for doing that are probably Spanish, French, Mandarin, or Japanese. And I would have to invest a lot of time to become very knowledgeable in those languages in order to obtain reliable job security.
- Around early 2023, I was motivated by the idea that adding people like myself to the software development field could counter the biological denialism that's manifesting the world's algorithms and Artificial Intelligence.
- Many smart people are able to program well, so I felt / feel left out by not being able to do it or able to specialize in it myself.

- There are many unique and original programs that I want to write and/or wish existed. I thought that it would be perfect if I was the one to create them myself.
- With some of my software project ideas, I envisioned that they could take off and become the basis for creating a new tech startup where I take the lead as the CEO. In reality however, that never happened.
  - First, I'd have to plan them out. As of *<2023-10-18 Wed>*, I haven't even done that yet. But once I have a plan to guide me, it'll be likely and realistic that I make them happen.
- Many software engineering jobs have systems where programmers respond to ticket request for new features to be added to the already existing software. This would feel similar to my current website's development where I have a list of dozens of improvements that I want to add to the website, but I don't have the time to implement everything as a single person. The idea was that if I could handle and enjoy a workflow that's similar to the one for my website, then I'd enjoy a similar workflow as a software engineer, right?

## 5 General Potential Disadvantages Of Being An Actuary

- You'll have to pass all the exams. They all have passing rates that tend to average at ~40%, and many of the people who do pass are the ones who are taking the exam for their second or third time.
  - I'm good at studying and taking tests, and I won't mind studying things if I enjoy the topics. One should not take a test if they won't be ready for it, but many people often don't have a choice since they could be fired if they don't pass the next exam.
- You'll have to like working with insurance.
- Actuaries have to deal with office politics and corporate environment, although you'll be working alone most of the time.
- Some computer programming will be required, although it'll all be focused around data.
- If you are too obsessed with studying nitty gritty details, then you may not like getting promoted since you'll be looking at data from a higher level from the company's perspective.
  - This might be a downside for me if I decide that I like deep details, but I don't think I'd mind moving away from it if I'd be getting paid more.
  - While I like higher status jobs, I also wouldn't be very satisfied a high status job that doesn't do much. This suggests that doing meaningful work is more important to me than getting jobs with higher status, if I could only have one or the other.
- You'll be sitting/standing at a desk most of the day.
- My linguistics degree probably won't be applicable to the work that I do (but I still won't regret all the courses that I took for it).
- What if AI automates so much of the actuarial field that it reduces the need for actuaries? I don't know for sure whether this will be a problem or not, but I do worry about this.

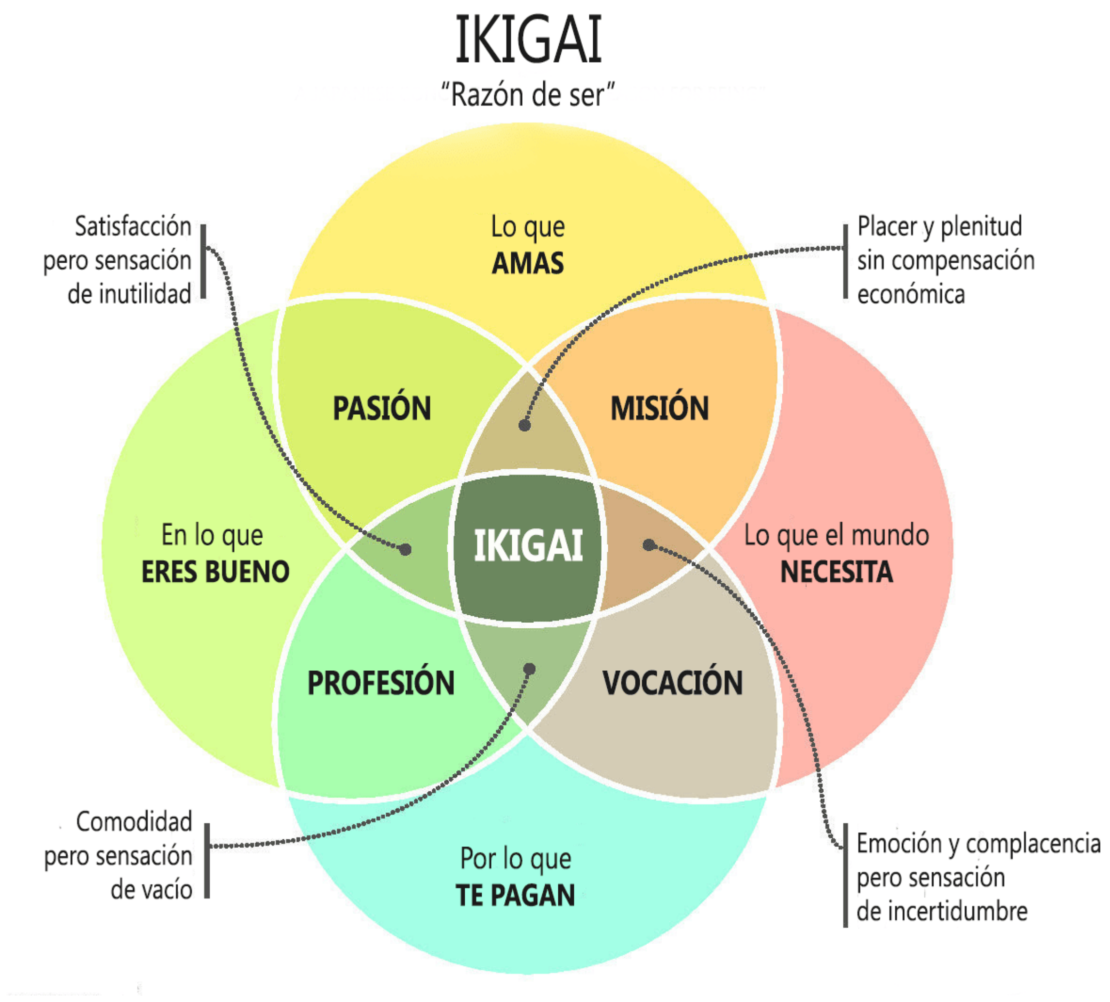
    - If it were possible for that to happen, then essentially everybody in society would be facing a midlife crisis in society regarding their job security and secured future (except for software engineers), in which case, the political upheaval would probably be strong enough to motivate everybody to make the government create a UBI program or something.
    - This may be unlikely to happen. The argument is that General AI is unlikely to become a substitute for human knowledge, even with quantum computing power and increased electricity generation. So it would still be necessary for insurance companies to hire actuaries and human brains.
- Maybe I'll find out more disadvantages to being an actuary once I start working as one?

I hope I don't get stuck with MS Excel, but I'll use it if I have to. I've heard there's better programs out there for doing actuarial work. Some of them might even be open-source and able to run on Linux.

## 6 What Motivates Me?

### 6.1 Ikigai

Being an actuary would probably satisfy nearly all four of these Ikigai categories, for me anyway. I'm likely to be good at it, likely to enjoy it (especially compared to other jobs), it would pay well, and the world will need more actuaries, especially since disasters are only more likely to increase ever more in the future.


IKIGAI
"Razón de ser"
Lo que AMAS
En lo que ERES BUENO
Lo que el mundo NECESITA
Por lo que TE PAGAN
PASIÓN
MISIÓN
PROFESIÓN
VOCACIÓN
IKIGAI
Satisfacción pero sensación de inutilidad
Placer y plenitud sin compensación económica
Comodidad pero sensación de vacío
Emoción y complacencia pero sensación de incertidumbre

## 6.2 My Motivations

I can gain motivation to do something by imagining it as my only choice / option for accomplishing a desire/objective. Alternatively, I could recognize that there's ultimately only one choice for me to take when doing something, when I view myself as an object in a deterministic, cause-and-effect world. For example, I could think of an actuarial job as "this is what I have to do to get paid money, and this is my only option" as my motivator? I tend to feel more motivated to do tasks when I think of them in this way. This mindset gives me a marginal boost in motivation for completing actions that I have low motivation for completing.

Oddly however, thinking that I *have to* do something doesn't motivate as much as thinking that a task is my only option for getting what I want or satisfying my need. This is probably because thinking that I *have to* do something can make it feel like a chore and/or make the task feel more stressful.

Watching numbers and down are a source of motivation for me when I recognize that my actions have the ability to affect their rise and/or descent.

I also gain motivation when I imagine the end results of my work and how badly I want to do the work necessary for achieving the end result. This suggests that having a strong, "must save the world" vision for building my belief network and debate software could motivate me to continue learning how to program, even when it's difficult for me. Realistically, I don't think a "must save the world" mindset would help me get stuff done for almost any career that I do, since most careers simply won't have that impact on the world. That mindset is best reserved for my side passions, including my philosophy website, any passion-driven programming projects (centered around my linguistic and epistemological interests), and all other academic/mathematical pursuits, research, and Wikipedia editing that I hope to do eventually someday.

As I work everyday and pass one actuarial exam after another until I'm fully qualified, I know that this career will become my daily schedule for the foreseeable future, perhaps decades as long as modern civilization doesn't collapse. I've managed a daily schedule before when I went to school and college, and I can certainly do it again.

## 7 My Previously Envisioned Dream Careers Over The Years

- 2012 - This was when I first learned what an actuary is, and my father first suggested it. I didn't realize that I would be seriously considering it as a real possibility 11 years later.
- 2013 - I think I would be comfortable doing many different things, so I'm not sure what I want to be, especially since I know so little about what various different occupations would be like and what they would each require.
- 2014 May - I want to be a computer scientist.
- 2016 April - I want to be a linguist.
- 2017 September - I want to be a computational linguist.
- 2018 October - I want to be a mathematician.
- 2019 February - I want to be a mathematician or an AI researcher that works in academia, and I wouldn't mind temporarily working in CS for a few years to pay off my loans.
- 2019 October - I want to be a data scientist / software developer / actuary / mathematician / AI researcher (whatever will pay my bills the best), but ultimately I want to contribute to knowledge. I first considered data science when I saw a flyer on a bulletin board in the stairwell of the mathematics building showing that 8% of data scientists have a bachelor's degree in mathematics, then I gradually looked more and more into it.
- 2021 December - I'll try to become a software engineer, because I'm black-pilled about academia, I'm not aware of my other possible options, and I haven't yet realized why I don't want to be a software engineer.
- 2023 February - I briefly wondered if I should become a philosophy professor, since I couldn't stop thinking about anything else when I was trying to stop working on my philosophy website.
- 2023 March - I thought I wanted to be a professional software engineer, although there were several problems with that pursuing that line of work, which I thought I could eventually resolve at the time.
- 2023 April 5/6 - I may be better off trying to become an actuary or a statistician instead of a software engineer. On this day, I watched "But what is the Central Limit Theorem?" on 3Blue1Brown, which made me wonder if I could get a job working with statistics and exploring further mathematics instead.
- 2023 May - I want to be an actuary.