1 Introduction

Hello, and thank you for considering me for this position. I just wanted to tell you I found this a very interesting interview to work on. I have tried to complete it quickly and honestly yet still to a certain quality. However I may have left a spelling error or two in, and have not precisely tracked all my references. I believe this is in the spirit of what you were asking of me.

Sincerely, me

2 Engineering experience

2.1 Describe a skill or knowledge you acquired recently that has been impactful for you. Why did you make this investment? What has the outcome been?

I have been studying legacy code and techniques for managing it for a number of years. Recently I had a breakthrough when I read the book "Growing Object Oriented Software, Guided By Tests". It took Test Driven Development which I have been using for some time and expanded it. It always felt like there was a "And Then Magic Happens" step that took the overall big picture requirements and allowed you to break it up into a lot of little steps. Most practitioners seemed to pretend that the design happened magically with no effort. This was an epiphany for me and has led to me being a lot more productive when I am programming.

I made this investment because a project I had created had become legacy code. It was our flagship CAD software we gave away, and had been modified and grown by many different engineers over many years. The entropy of the code was reduced and safety was increased enough that it became possible to make changes again. This was a process that took a little while and was not a sudden miraculous change but rather a gradual change.

2.2 What new skill would you like to learn? Why do you think this is important or timely or interesting? Why do you think you will be good at it?

Frankly, I want to learn Rust better. I believe it will be a pivotal language in the future. I have worked through an udemy course and plan on taking a couple more as well as searching out some rust open source projects to start contributing to. I will also start playing with my own projects. I have never struggled to learn a new language so I believe it will be relatively easy to master.

2.3 What kinds of software projects have you worked on before? Which operating systems, development environments, languages, databases?

I have worked on signal processing on FPGAs in VHDL and C. I also developed the algorithms for this in Matlab. I have programmed 2 full golf card motor controllers in assembly and C on a TI F2808 DSP. I used ubuntu as my dev platform, and a mixture of python and makefiles for my build system, and emacs for my editor. I used python to build an interface to the first controller and C# on windows for the second controller.

I have used Java to create a desktop CAD software package that runs on Windows, Linux, and Mac (www.protocasedesigner.com). It was specialized CAD software for making sheet metal enclosures for

electronics. I used a mixture of windows, ubuntu linux, and mac osx to develop since it needed to run on all of these. It started at Java 1.6 in 2009 and when I left the company was at Java 11. I ran a team that took that software and created automated tools for our engineers to use to create work instructions for our manufacturing team. I also created a SOAP reader for a piece of equipment we used and made a microservice of the reader so other services in our quoting and engineering could use it. I have created software that allowed us to analyze pricing changes on the fly with all the data we already had in Kotlin, and created a piece of vision software that identified parts on a printer bed and identified their location and rotation so that we could print on them without lining them up in Kotlin with openCascade using Intellij on Ubuntu.

I have used Linux from Scratch, Debian, Gentoo, Archlinux and Ubuntu (starting around 10.0 I think...). I use lubuntu for my primary OS these days using the bspwm windows manager. I am familiar with Windows as well. I have developed in Netbeans, Microsoft Visual Studio, Emacs with terminal, Eclipse, Intellij, Microchip's MPlab, and many others.

2.4 Why Rust? What is your history with this language?

I love the memory management. I like that it feels a bit old-school and powerful without having the vulnerabilities of older languages. It takes the careful checking off the human and puts it not on the runtime but on the compiler where it belongs. I believe it can be used for embedded software (which is one of my loves) without the overhead of C++. I also think it will be a good move to develop more expertise in it. Finally, I know it doesn't take long to learn a language once you have learned many. I have experience with C,C++,C#,objective C, Assembly, Pascal, Ada, custom microcode languages, VHDL, Java, Kotlin, Groovy, Scala, Smalltalk, R, Matlab, Python, and Basic amongst others. I love new languages and am looking forward to mastering this one.

2.5 Would you describe yourself as a high quality coder? Why?

Yes I would. I have been coding since 1983 when I bought my first computer at a garage sale. I made many mistakes and learned a lot. I studied math and computer science at university, and then spent the last 24 years of my career as a systems engineer, software developer, and R&D Manager. I had projects that lasted 14 years so I learned what mistakes bite you later, and how to take messy code that has suffered from the entropy of time and multiple hands, and make changes safely without rewriting the whole thing. (My favorite books are "Working Effectively with Legacy Code", "Your Code as a Crime Scene", "Refactoring", "Refactoring to Patterns", "Growing Object Oriented Software, Guided By Tests", "Test Driven Development", and "Tidy First").

I have also had the privelage to work with many incredible mentors in my career. I have mentored a few young developers and engineers as well. However I can look at code I wrote a while ago and think that it wasn't very good. I believe this just means I have learned a lot in the meantime as I never stop trying to get better.

2.6 Outline your thoughts on open source software development. What is important to get right in open source projects? What open source projects have you worked on? Have you been an open source maintainer, on which projects, and what was your role?

I have used open source software but have not had the privelage of working much on it (I did provide a couple of script fixes back around 2001 I think it was for the amd x86_64 platform on Linux From Scratch). I have been employed by private companies that did not contribute, and was not able to dedicate time to open source. I would love the opportunity to give back however!

2.7 Describe your experience building large systems with many services - web front ends, REST APIs, data stores, event processing and other kinds of integration between components. What are the key things to think about in regard to architecture, maintainability, and reliability in these large systems?

At Protocase the CAD software I designed ended up having a web-quote integration via YAML files sent via http requests, and ended up getting used internally integrating with some of our hardware through SOAP interfaces running on microservices that used JSON over sockets, Solidworks through dxf files and C# add-ons we created, and other equipment through directory watched server instances of the equipment provider's software. Finally it had to operate with our internal custom ERP system which was primarily a set of business logic, manufacturing flow, and information control software for our Mass Custom Manufacturing business written on top of a mysql database.

Keeping individual pieces simple and keeping the APIs simple and unchanging as much as possible, as well as having automated deployments seemed to be the most important. Note I am not saying continuous deployment. I found it important to have to think before deploying, but having all the pieces of the deploy happen correctly automatically was very helpful.

Also, when I wrote the software on the motor controllers, I had to have a changeable API that required the same code changes in 2 places in 2 different languages and projects, so I wrote a python script to take a text file and write both sets of code. This was a fantastic success as it kept me from having the possibility of the two codebases getting out of sync.

2.8 How comprehensive would you say your knowledge of a Linux distribution is, from the kernel up? How familiar are you with low-level system architecture, runtimes and Linux distro packaging? How have you gained this knowledge?

I am a little rusty on the kernel and modules. I am familiar with packaging and most of the fundamental tools. I got most of this knowledge by working through the Linux From Scratch distro creation process back in 1998 when I first got fed up with Microsoft Windows and their C/C++ compiler and switched to linux and the gnu compiler.

Most of the embedded systems I have worked on have been low-enough level that I was just using roll-your-own schedulers rather than embedded linux. Again, would love to increase my knowledge and

experience here.

2.9 Outline your thoughts on quality in software development. What practices are most effective to drive improvements in quality?

Using a code complexity metric (my preference is either the Cognitive Complexity metric, or the sum of all the indentation in a file from "Your Code as a Crime Scene") goes a long way. When you combine that with the number of times a file has been changed in recent history from your git repo, this tells you where your bugs are going to come from. This is where you should really spend your time. It is worth actually refactoring this intentionally to knock their danger levels down. Otherwise, if you don't understand something write a test for it. Then refactor it so that you would not have needed to write the test in the first place. Then make your change.

If you are writing new code, and it isn't just a quick spike to see what way you are going to head, use Behavior Driven Development at a high level and Test Driven Development at the low level to create your software ("Growing Object Oriented Software, Guided By Tests"). It usually doesn't go any slower, and you don't get stuck as easy. When you go away and come back you remember where you were. Other people learn what it was that you intended when they run the tests and modify things. It preserves the code quality and prevents entropy (as long as you are decent at writing tests).

Pair programming is also very useful. It allows knowledge sharing and prevents getting stuck, and usually gets much cleaner code that does not suffer as much code entropy. However I have learned to call this practice "Working Together" because management expects engineers to work together a lot but the "Pair Programming" name seems to make them think they are getting less work done. I think this is because they think of programming as similar to building a building. It is more like planning a city. When you start, the city is small, just a few houses. Then as needs change, things get added, removed, etc. Before long you have a big mess on your hands if you don't learn to handle the change well and make things easy to change.

Other important practices are keeping commits atomic (one change per commit), frequently merging in to master, and whenever you are struggling with direction try throwing out the last few commits. You must talk to each other a lot, maybe even do pair programming. Speak with the customer often enough to understand their pain points and use the tools you build, use lint checkers and code quality metrics, and use a circuit breaker (found in Basecamp's Shape-up mini-book).

2.10 Outline your thoughts on documentation in large software projects.

I am going to break this into two sections: documentation on code for developers and documentation of software for users.

2.10.1 Documenting code for software developers

I believe the code and tests should do most of the documentation. If you need documentation for a task, create it new on a whiteboard. The reason is that any document that encompasses the whole of the project is unreadable generally, and always seems to get out of date. And when you whiteboard up a structure you can leave out the bits that don't matter for what you are doing.

There should be almost no comments - only for things out of the ordinary that you couldn't figure out how to make self documenting.

There is a big caveat. If you are working in a high-risk high-cost-of-failure environment, then extra care should be taken (safety is more important than efficiency). This could include architecture documentation and API documentation that is always kept up-to-date by careful procedures that are always followed. However usually good testing and a set of Behavior Driven Development test cases combined with Test Driven Development Unit Tests do a much better job of documenting things in a way that never ends up out of date.

For parts of the system that are fairly stable however I do find some UML diagrams useful. Also, for any complicated interactions a set of use case examples (preferably written into tests) are very helpful in talking to stakeholders. Timing diagrams also are very helpful. Speaking in examples is one of the most powerful way to get details out of the stakeholders.

2.10.2 Documentation of Software

When documenting software for users, there are at least three sorts of documentation: tutorials for general education which walk people through the software or subsection of the software and try to confer a general knowledge. Then there are howtos. These take a specific task and break it down into steps that can be repeated by the customer. Further there are reference guides that cover a lot of the important information around the software.

I believe that a good webpage based written document with lots of screenshots and the occasional video or animated gif is excellent for tutorials and howtos. Of course built in documentation is really helpful too. One example of software that used to have incredible built-in help was google sketchup. I haven't used it in ages, but when I did play with it, there were animated gifs showing the mouse and buttons which very clearly showed how to draw all the shapes. It was brilliant!

That said, you must write documentation that your user wants. So if they like videos, you must put out videos too.

Now when something covers a physical skill such as woodworking, nothing beats videos for learning these skills (besides actual in-person instruction).

For Reference work, Wikis are an amazing tool that can allow a lot of collaboration in the development of the knowledge pool. Care must be taken to keep them organized. Standards should be created and moderated.

2.10.3 What practices should teams follow?

Any important tutorial or how-to should be followed by getting a naive population of customers to follow the tutorial or howto. Their feedback should be taken into account. You cannot effectively judge what a person who doesn't know something can do.

Also, keeping systems as simple as possible (avoiding premature optimization or speculative development) will help reduce the need for extensive documentation (as well as prevent bugs).

2.10.4 What are great examples of open source docs?

I've always been amazed at how easy it is these days to figure out how to do anything. For instance, if I want to install octave on ubuntu, I can search it and go to https://ubuntuhandbook.org/index.php/2023/11/gnu-octave-8-4-0-released-ubuntu-ppa/ and find a good document that tells me everything I need to know. I might think it should be organized a little differently or whatever but it works. I can find everything I need there.

https://ubuntu.com/core is very clean, well organized and helpful. It also happens to be pleasing to the eye which is not necessary but is a really nice addition. It probably helps the choice to adopt ubuntu-core as it looks professional.

2.11 Outline your thoughts on user experience, usability and design in software. How do you lead teams to deliver outstanding user experience?

For starters if you are going to design or work on a tool **you must use the tool!** Otherwise you cannot talk intelligently with the customer about what their needs are. I also believe you should only change things for good reason. Moving a button can wreak havoc on an engineer that has developed muscle reflexes that expect it in a certain place.

You must love the domain you are working in, and love the customers. Respect their work and desire to help them. So you must have empathy. If you don't come by it naturally, then you must come by it from shared experience and really use the product yourself. As a team leader I required that my members spent enough time working with the tools they made, and talking to the customers they served, so that they could understand the problems that needed solving, and what problems might not.

I am a believer in minimizing customer interactions, not because we don't want to talk to them, but because the less back and forth they have to go through, the more efficient they are.

2.12 Outline your thoughts on performance in software engineering. How do you ensure that your product is fast?

For starters, speed is almost never the issue in software. But when it is I have always had the best luck with logging systems for doing a binary search for the slow spots. Then optimize. Never optimize before you have a speed problem as most optimizations make things harder to read. The exception is when things are slowed down by some unintended consequence (like reloading a file from disk every time you access an object). This is also easy to find with logs.

2.13 Outline your thoughts on security in software engineering. How do you lead your engineers to improve their security posture and awareness?

I think you must know what the best practices in your industry are and follow them. Working with a partner on critical code really helps here - common sense is easier to come by when there are two heads working on something. There are also some good tools out there for analyzing your dependencies for any known vulnerabilities, and you should be using a lint checker for your language. Finally, know when you are working with something that could be compromising and take extra care. Security by obscurity is not good enough.

2.14 Outline your thoughts on DevOps and DevSecOps. Which practices are effective, and which are overrated?

I have not had the opportunity to work or even read much on DevSecOps. But DevOps definitely has its place. I believe you start to use the practices as they become necessary. What is overrated and what is useful depends on what you are doing, what your environment is, how big your team is, etc. I have

not experienced the full range of possible work environments so I hesitate to disparage any particular practices. In my experience Continuous Delivery is of lesser value than Shared Ownership, Automation, Continuous Integration, Cross-functional Teams, and Continuous Monitoring.

I do know I have seen first-hand that CI/CD greatly improves throughput and cuts down on the occasional really difficult deploy that takes forever and causes a million bugs. I remind myself of Kent Beck's statement of Constantine's equivalence:

$$Cost \approx Cost(change) \approx Cost(hard\,change) \approx Cost(coupling) \tag{1}$$

Continuous Integration and Automated Testing usually prevents you from going down a rabbit hole of really bad coupling causing you really big costs of change. https://substack.com/@kentbeck/p-142791879

3 Industry leadership experience

3.1 Describe your speaking experience at industry events and conferences

I have not done this on a large scale. I would love the opportunity but the closest I have done besides my work experience as a Mathematics Instructor has been speaking to groups of students about our engineering tools at Protocase. I have also organized and run a hardware hackathon. I have also spoken at community technical events and lead lunch-n-learns for my team for about 10 years.

3.2 Are you a thought leader in any particular area of technology?

I don't believe so - that would be a huge claim to make. I am very familiar with several thought leaders - Uncle Bob, Martin Fowler, Kent Beck, Michael Feathers, Steve Freeman, Nat Pryce etc but what I am good at is seeing through fuzzy requirements and finding problems before they arise. I also am a strong advocate for several techniques that are maybe cutting edge.

Perhaps I don't see myself as a thought leader because I surround myself with people who know more than myself and try to learn as much as possible from them.

3.3 Describe any experience working with startups. What did you draw from that experience that would be relevant for this application?

I spent 4 years working with a startup creating a couple of motor controllers for high powered golf carts. I learned a lot about what good enough is, and what risks are ok and what are not. I worked with an experienced electrical engineer who had started and run a couple of successful startups before. I also learned a lot about good engineering practices from him.

My last job was with a near startup. They had grown to about 40 employees and were not quite making a profit when I joined. I helped steer company direction through the Socratic method - asking questions. I am not afraid to admit I do not know something and just ask. This allows many mistakes of ego are avoided. Egos seem to be involved in many failures in startups. In my first startup the electrical engineer believed in a new technology so much that he would not make a backup plan. When the technology didn't live up to its promise, the startup failed.

3.4 Describe any experience working in a public company. What is important for your colleagues to know about being a public company?

You fight a different kind of battle in a large public company. Middle management is not incentivized to speak truth to power, or to take risks. So you have to be an advocate for what you see as reality. Also, you must be a self advocate if you are a high performer or you will not get compensated as one. This is from my experience at Raytheon. I doubt all public companies are like that, I think it was a result of the vast size of the company more than anything.

4 Education

4.1 How did you rank in your final year of high school in mathematics? Were you a top student? On what basis would you say that?

I did not finish high school, but I went immediately to college and finished with a Masters in Math top of my class with a 4.0/4.0. I was always a top student if not the top student in math and science. As a child we lived on a clam farm and had to dig clams in the middle of the night to eat. My father always told us "If you don't want to do this your whole life, learn your math!". So we all did. But I was also the top math student in my family which was tough competition!

4.2 How did you rank in your final year of high school, in your home language? Were you a top student? On what basis would you say that?

I was a decent student in english but I didn't become good at written (and indeed even spoken) language until working at Raytheon. I had an amazing mentor/ boss there who was one of the best engineers I have known, and also one of the best communicators. He understood people better than anyone I ever knew and taught me a lot. When I took over running the R&D department at my last job I also started studying communication and conflict management. I read intensely on the subject and discussed it constantly with my bosses and coworkers. I took a few classes in this time as well.

4.3 Please state your high school graduation results or university entrance results, and explain the grading system used. For example, in the US, you might give your SAT or ACT scores. In Germany, you might give your scores out of a grading system of 1-5, with 1 being the best.

My SAT scores were 1560/1600. The breakdown was 800/800 in Math and 760/800 in English.

4.4 Can you make a case that you are in the top 5% in your academic year, or top 1%, or even higher? If so please outline that case. Make reference where possible to standardised testing results at regional or national level, or university entrance results. Please explain any specific grading system used.

In my university I certainly was. I scored the only perfect 4.0/4.0 in my Masters Degree. There were two other students who were vying with me for that spot but they lost out in a difficult Differential Geometry class the last term that I got an A in and they did not.

When I took the SATs my scores for university entrance, I scored in the top 1% in the nation.

4.5 What sort of high school student were you? Outside of class, what were your interests and hobbies? What would your high school peers remember you for?

In high school I was an ok student. I found the work not challenging or interesting. I got good grades but did not try hard enough. I did not feel challenged or held to any kind of standard. I loved shop class, welding class, electronics. I also loved the outdoors and hiking. People remember me as nice but quiet and awkward and being that kid with his nose in a book all the time.

4.6 Which university and degree did you choose? What other universities did you consider, and why did you select that one?

I went to Western Washington University for a Masters degree in Mathematics. I was able to take a lot of Computer Science courses as well. I had considered Harvey Mudd but decided I could not afford the tuition so I went to my local school instead. I also applied at MIT and Cal Tech but did not get accepted there despite my high SAT scores.

4.7 Overall, what was your degree result and how did that reflect on your ability? Please help us understand the grading system for your results.

In my last degree (Masters in Mathematics) I scored a perfect 4.0/4.0 and took the most difficult courses I could. I also scored in the top 20% in the global Math Modeling Competition IM2C. It reflected well on my ability to learn and absorb knowledge. Particularly the understanding of why and how. I was not that good at rote memorization so had to lean on understanding concepts and being able to figure things out.

4.8 During all of your education years, from high school to university, can you describe any achievements that were truly exceptional?

All the professors came to my thesis because they didn't understand the topic and wanted to learn it. It was on Frobenius's Theorem on non-conservative vector fields. Also placing in the top 20% in the IM2C was a large achievement. I did all this while paying my own way through most of college and university.

4.9 What leadership roles did you take on during your education? Did you conceive of, and drive to completion, any initiatives outside of your required classwork?

During my education I was usually working so did not take a strong leadership role in a large sense. However I always ended up the default team leader for any group activity if there wasn't someone who came in to take the role. People always came to me for advice, perhaps because I worked in the Math Tutoring labs whenever I could, and perhaps because my unique upbringing gave me a wider set of experiences than most people get. This has always served me well because it gives me the ability to view things in a different light.

I also took part in the IM2C international math modeling competition. We had a 3 day time period to answer the question "Predict the effects of a 1km wide asteroid striking the earth". Very broad. I was our team leader and got this done on time and we placed in the top 20% in the world with our paper titled "Oh Sheep"

5 Context

5.1 Outline your thoughts on the mission of Canonical. What is it about the company's purpose and goals which is most appealing to you?

I like working on things that matter to people. As a service organization Canonical gets to do things that matter to a lot of people. I enjoy personal interactions with customers (even when things have gone poorly) because it really satisfies my need for a sense of purpose. The idea of making open source software and then providing service is a wonderful model. Firstly, it gives that open source software some stability. Ubuntu has been my favorite distribution for a long time because it is professional and dynamic. Easy to use, not overly bloated.

The open source side also means that it gets a lot of eyes on it. This reduces the chance of major security holes or deadly bugs.

I also love that the company has a strong belief in documentation, but gives good guidance on how it is done. Too often everyone is left to document in their own way. Knowing what the expectations are and how they must be met makes it much easier to be consistent!

5.2 What do you see as risky or mistaken in our offering, positioning or strategy?

Well Canonical has been around for quite a while, and is still here. That is the best proof that in general the offering, positioning, and strategy are good. As far as risk, I believe there is a lot of risk in AI as a ton of work is being done there by many many people so a good chance what you do won't end up used. But at the same time the payoff can be high so it would be foolish not to get in on it too.

In the past there was the unity desktop environment which I believed was a mistake at the time, but it was a mistake Microsoft and Mac also made and have since backpedaled on to a certain extent as well.

Finally, the biggest risk is that people will not associate the name Canonical with the commercial services they offer.

5.3 How should Canonical set about winning, commercially?

From looking through Canonical's offerings, I would guess there are two major income streams. One is support for ubuntu and other Canonical products through ubuntu pro, and the other is more like contract work for companies wanting to do something specific with opensource who don't have the internal expertise to do this. Not knowing how successful Canonical is besides not having gone out of business, I would say it is more of a sales and marketing issue. Canonical has a broad range of products and there should be a huge market out there. Canonical's products work well and add value.

If I wanted to understand how to improve my marketing and sales performance, I would be getting engineers to go with sales to talk to the customers we have so they better understand the needs of customers. I would also look at how Fedora and Oracle manage their reputation. We know they are for profit groups who are doing a community service but have something strong to offer rather than community groups that do this work out of the goodness of their hearts. So if we need work done in their area of expertise we would think to approach them.

It would also be interesting to see if there was a support level Canonical could offer for sale for personal use of ubuntu for those that didn't want windows or mac but weren't strong enough technically to feel comfortable jumping into linux. Like a $20 support license.

Finally could Canonical start some certification courses and charge for them?

5.4 How should Canonical amplify its impact in open source?

This is somewhat hard to answer as Canonical is doing such a good job with this already. But maybe if we could get some cool open source hackathon type work going on with high school and universities we could get more people interested at an early age. Also, when I did a google search of how do I contribute to open source projects, canonical did not show up in any of the first two pages of results. So we need more blogging, tutorials, etc on the subject. Lets be a thought leader in what it means to contribute.

The pages on contributing to Ubuntu are a bit confusing and if I want to fix bugs I don't see a direct set of directions on how to get started with this at https://ubuntu.com/community/contribute/ - I see helping triage bugs, helping with documentation, and how if you have a fix you would submit the patch. But a leaderboard that showed the top contributors along with a hit-list would be really great here!

Finally if we could publish a free class on how to get started contributing to open source for software developers it would be a boon and a good marketing ploy.

5.5 Why do you most want to work for Canonical?

I need three things out of a career.

- a good remote job that lets me travel now and then and pays well enough
- a strong technical challenge
- the ability to make a contribution that matters to someone

I believe Canonical meets these criteria very well. On a side note, I prefer that my development environment is a linux terminal with emacs and some text based build tools. If the problems occasionally dip into a math problem that is great too! I really enjoy working with others and do not see remote work as a way to avoid interaction. More than anything it is a way to be flexible in where you live and not spend most of your free time commuting.

5.6 What gives you the most satisfaction at work?

Solving a problem that makes someone life better, mentoring someone, figuring out how something works, and being part of a community/ team.

5.7 What would you most want to change about Canonical?

It looks from the website and my experience that the company's presence and direction are engineer led. This is great, but it could probably use a bit more customer-centric influence in the marketing. At my last job I went on a lot of sales trips with the sales and marketing teams to meet customers and help the sales and marketing teams develop a sense of how to interact with customers who were engineers. It also gave me great insight into what sorts of problems our customers had, both with our products so I could make them better, and with their own jobs so I could create new features and products that could help them.

In general with a technical company, if you want to increase the value of the company you get more return for your money by increasing spending on marketing and sales in my experience.

Now obviously I don't want to come across as knowing better than the company, nor is my intent to criticize the sales or marketing teams. I am not privy to the inner workings of Canonical so this is just my thoughts from what impressions I can get. Still, I love this sort of discussion and would be happy to engage in a deeper discussion about this if I am chosen for the role once I know the lay of the land. In this case there might be room for some education of the public in the services Canonical offers.

5.8 What gets you most excited about this role?

I am very excited by the transition to Rust in the industry in general. I want to be part of that. I would also like an opportunity to give back to the open source community. I see you have some embedded work with ubuntu core that I did not know about. It would be awesome to help this along as well if that was possible.

Finally, it seems that Canonical is big enough to have an impact, but small enough that I could have an impact on it.