Note: Transcript is pending review and revisions by The Maintainers. Please contact us if you have any comments or questions.

August 2020 Ross Spencer

Speaker: Are we recording by the way?

Speaker: Yes. So now that our introductions are done this session will be recorded so Jess are you the one who's going to hit record on this?

Speaker: We are good to go. We are now recording.

Speaker: Thank you.

Speaker: Thank you. Yeah my name is Ross Spencer I'm a software developer in my day job, Artefactual Systems. Outside of my day job I'm also a software developer. So there's a lot of (inaudible) in my life. (Inaudible) Star Trek, Star Wars or the Stars. Well just ask my partner I'm probably happiest on the wing so I'm such a traveler which would make it really torturous for me to sort of go out to look at the stars when I know that I can't quite get out there yet. So I'm very much a Star Trek person and I think I don't know I just love the writing in that show and there's one episode where Picard I don't know if he's breaking script or character or if this was written they send Geordi out on a wing mission to a Klingon ship and they stop the show almost for Picard to explore how he views the world and how he's looking out at this (inaudible) and he's like I can't believe this is how you see and you know it was really wonderful. You know I just I mean it's such a great meditation on so many things. I thought that was going to be a good segue into my introduction but actually what I wanted to touch upon was when Jordan and I first started to talk about this I suggested it might be a bit of an existential meditation but I'm not sure if it really lives up to that you know what I've got down on page here. But what I hope I get right given the group and how much discussions tend to be is that I will ask a lot of questions that I hope will stimulate some good discussion. I think the running time will be under twenty minutes and so that would be great. So no pressure for asking questions at the end but you will need questions. So my GitHub profile shows that I've been working open source since 2012 but actually outside of GitHub I was publishing to a website called Planetsourcecodes.com since about '98, '99 which makes me feel incredibly old right now. Applications like the Yoda lottery number generator and the original pigeon racing game simulator. It was called Pigdge Racer. My publishing didn't really die down during that you know the decade in between when I you know got back onto GitHub but GitHub really did make things "take off" for a number of us. I think at least provided better tools to socialize our work. It was at the National Archives UK that

my work really became open by default. I was open by default because I couldn't get the support I needed at work. I recognized open source might be a way around them. I created two separate projects there around (inaudible) identification in democratizing that work. There was a related in house tool which that work was based upon which was published in the open and to the public and departments. Interestingly both tools I created were open source but more important than there being open source was just that publication as a service that could be used. One tool the Signature Development Utility enables you to (inaudible) identification signatures compatible with the (inaudible) format identification tool which scans file formats for (inaudible) file format matching numbers. Being able to publish that onto a website was key to opening up that process for other colleagues in the fields where the process we used internally were very much based around our colleagues database with restrictive access. It's interesting for me to reflect upon taking the (inaudible) open source from them and think about relationship with publishing in general. We really didn't get any investment incurred from external contributors but folks did get use from the tool. Ideally what I would have loved to have happened by having the (inaudible) plan would be for others to take (inaudible) user interface making it more user friendly, enable to aggregate lots of these file format signatures into one where it was strictly limited to one primarily because of my lack of (inaudible) skills. But the code is all there so folks were able to do that if they wanted. I have a little bit more experience now and so I think that there might be a little bit more news on that work nearly a decade later, later this month. Still without a great deal of participation in that effort (inaudible) I do ask questions both from the time and with a little bit of hindsight. Should I have promoted it more? What are the indications of that? I really only have my personal (inaudible) to talk about it at the time. Was it the right language to write to put it in? PHP with clunky (inaudible) JavaScript and some pretty (inaudible) html. If it was the right language for the need would things like lack of testing or quality of code what people are. I think it might. Another nice open source project that came about in New Zealand again related to content analysis came about in New Zealand and it was again related to content analysis but whether we would have written it or found ourselves contributing to another project was probably about 50/50. Contributing to another project would have been really nice. I started with some due diligence looking at a tool called C3PO. The C3PO came from Clever, Crafty Content Profiling. The tool was developed by a Petar Petrov and Artur Kulmukhametov at the Technical University of Austria in Vienna. I should just know that I think that's already two Star Wars references in this work. It would take analysis of (inaudible) and provide collection level analysis of statistics and charts. Actually you know what I can send you a link as we talk about that. For a developer without the ability to create compelling visuals something Petar knows at the moment it was a nice looking choice for us. With some decent effort over the

weekend and then back at work on the first Monday back after deciding to try to use the tool I wasn't able to get the application up and running. I didn't understand the dependencies well enough and I used Java which I already found quite difficult to work with. And also using something called the play framework for publishing web apps which I had no idea about. Any particular dependencies in general were going to be a tough sell to the department which have pretty rigid laws around that because of the other government departments that were on the same network. So we started to look at creating our own tool. It took the output of format and implication tools (inaudible) data components that help with the (inaudible) preservation system. I'm not very good at naming. For a while (inaudible) engine but a colleague at archives in New Zealand has since suggested (inaudible). So let's go with that. The features of the tool included (inaudible) style at the time and it was called Python but the only things that were available in the standard Python language. The language allows you to import through (inaudible) libraries through a tool called Pip. Pip will save those libraries into the correct locations on the (inaudible) system in virtual environments and then they can be used. But again, writing around the dependencies in the department was going to be tough. By the time Pip couldn't even be used over the network so there would need to be some managing upwards across teams and outside of archives itself if we were to try that. I work a little bit more freely now but I thought it taught me a lot and having (inaudible) organizations I try not to assume what (inaudible) organizations to install and use other software. I want us dealing with something different. If you can get a hold of a programming language and use it I still believe there's a lot that can be achieved but I know for some of you it can be tough. I should note as well I'm not advocating that everyone should code here there is a lot that you can contribute to open source without code. Another feature of the tool was that as we started to demonstrate the value of it we were able to socialize it a little more with the organization and we (inaudible) create entry level digital preservation tool tips in an application (inaudible). Because it was open source that knowledge is encoded now and available just by downloading the (inaudible) GitHub which I think particularly reinforces mine and I hope others decisions create open source projects. Some questions I have about that tools experiences are were the C3PO repository more active? Would it have made a difference to how we first attempted to engage with it? (Inaudible) I think contributing to it would have been perfect for us at the time. I think it can enhance reputation and it's really important not to reinvent the wheel and to support projects as we ourselves hope that we can be supported in the future too. How many others in the (inaudible) have looked at something but found its entry points really difficult? Hands up from Jordan. I have also then chosen to write something for yourselves. Good to see. We'll just find a better alternative maybe even pay for something. Do many others have restrictions on what they can install at work? Not just to

code but any tool they might want to use. Creating this tool initially was easy to do alone but eventually it took a team in (inaudible) New Zealand to contribute that feedback and to then create something more useful. Making it public was an important goal achieved. We were sharing knowledge. Ideally though putting into the common (inaudible) seeking input from others, looking for bugs and fixing them and there was a (inaudible) translation too which would have been cool. I mention in response to Devon's (inaudible) message this week that there is a temptation for me to say there's a great intersection between what we need and (inaudible) open source. But for all of the analysis tools arrangement of packaging tools we create is it in fact just a reaction so the (inaudible) we work with around the outside of the (inaudible) preservation workflows. If you look at some of the other work we were doing we were training things like the output of our analysis tool so other mechanisms that interface with much bigger systems our open source tools (inaudible) does start to look like a reaction. Tools for passing metadata from enterprise record management systems which largely seemed to be closed source and the extracted data always…and those systems where the data extract mechanisms aren't always intuitive. The records are yours of course. Tools for creating submission information (inaudible) preservation systems as well which is work (inaudible) depending on the system. When talking about records we talk about create to maintain. It would be great to see more open source options when we create and store records rather than having to write more code and response when it comes to disposal and archiving. I hadn't really a point there actually but it's just something I like to meditate on. I should search upon my current role because I am a software developer at a company that maintains open source software. How do things like change when working in open source professionally? Well first I think for all of us doing this if you're in a software house or somewhere else we're already doing it professionally but there isn't a difference. I appreciate there might be great gaps in the resources available to you or the amount of buy in you can get from management. But yeah working for a company that's known for software perhaps more specifically services around software doesn't change too much. I don't feel as I wrote this today anyway. One of the things that folks can really benefit from is pure review of courage. I had witnessed that from my first role many moons ago or was it (inaudible). But it has been good to change my current style of reasoning. Being able to work with (inaudible) daily now (inaudible) my time outside of work to sell out to an organization really makes a difference to my well-being and also it does prepare you a little bit more to take on more complex projects and make your work feel more professional and make it something that you feel like you might be better able to maintain into the future. Some of the challenges are familiar. Even a well-known application doesn't necessarily have a massive number of public submissions. In a more well-known application as well there's more understanding too that submissions aren't necessarily a

blessing to the project. As it is relied upon in certain ways by users it is an important skill for our project (inaudible) to help maintain its source consistency. And for developers it's a skill in shaping external contributions into something compatible to the styles and encoded knowledge of the system proper. In Jordan's email before the talk that's the (inaudible) and the turner community building skills that we need to develop. We spoke about resources. Well even having to review – actually looking at the same things seems to be able to make a difference in quality and splitting responsibilities like coding while another writes documentation tasks or deployment scripts for example. Of course where I work we probably have a factor of 10 more that work on the product lifecycle. Not insignificant, but not quite as (inaudible) project either. And if you really need to at the very least (inaudible) there are ways I can see that we can do this through distribution systems like GitHub and (inaudible) that are available to us. I've spoken about (inaudible) projects, but the backlog of products is many. I've selected these two because they're probably the ones that I have invested the most in and also they speak most about my intentions (inaudible) in digital preservation and information records management. There is an emphasis on maintaining long term record data, looking after information, and opening up to others. Making it open is always in the hope that it will be useful to at least one other person or group of people. (inaudible) hypothesis knowing something is out there that has been tried even if it didn't work or showed something other than you anticipated is important too. You can save others time by putting in your most esoteric or drafty work out there. There's a lot of other projects available in my GitHub (inaudible). None of them really got the open source magic sprinkles though. It would be nice to figure out the magic for (inaudible), but I do suspect it will be asking to bite off more than I could chew so I had to (inaudible) have further questions sort of that stem from all of that experience. Participating in a study over Christmas from the University of San Diego one question that popped into my mind was what it would mean for a project to essentially go viral, to be what some might judge, successful. They say to make a website successful you need an Instagram, Facebook, and a Twitter as well as your website, but for (inaudible) projects we would like to share it you would also need to prepare to write the code and to follow up with (inaudible) or be out there promoting your work in person in conferences. How do you do that without backing? How do you do that if you're working in your spare time? Does it differ culturally? Larger countries with greater population may help promote your work or do different locations have greater openness to technology? Would a (inaudible) be enough? They're a good thread for specific programming languages, but I haven't done that, which speaks to another point. It's scary and there are still plenty of issues around open source and being out there on the web. It's almost easier not to promote work at all. I feel like publishing from my experience anyway might be safe. I think less people are looking than you might think and there are

fewer people with the skills to look at it too deeply, but actively promoting something to seek contributions does feel like a much greater leap. I'm going back to the top a little. We had one project that was difficult to resurrect, but I think people might have had better luck since. It does make me think using existing code within a few years of its creation is such a low bar. Others in the room have more experience here, but how do we really preserve it for later to use later on? Unrelated I know there are restrictions at previous organizations of what we could and couldn't do. The resolution was to work in (inaudible) only. I personally think there's a resilience that might bring in and potential to preserve. All you need is the code and its run time. There are no dependences, but there's some distance with that work. There is a trade off to the enjoyment you might get from developing and the ease of creating something. Would it be worth it? Would it really be worth it? After all what is the lifespan of what you're creating? And related further is a question I really like to think about. Is there a perfect language? Is it something that exists? Is it something new that doesn't exits? Would some standardization of languages and practices help specifically code sustainability and socialization and sharing of knowledge? What would it need? Ease of installation? Ease of translation? Ease of writing text? Ease of writing (inaudible)? These have all been considerations when choosing anything at all. In short, is there a framework that the heritage sites are going to adopt to share a vast knowledge and skills and amplify capability? That's not really a (inaudible) today, but it does happen to (inaudible). Really the last thing to put out there is I would love to know what resonates in this very brief reflection here with others on what open source means to them.

Speaker: Thank you so much, Ross. I am halfway through responding to a comment of Kelsey's, but I'm really happy to hear not just some of my – as someone who is a total newb in the open source community world I work with. I lias with developers in an open source shop, but I actually remember being asked in my job interview for this position how do you decide whether or not to learn a tool on your own or let the people with the expertise do that work and in my first year and a bit in the role I'm like okay, I'm going to understand the basics of Git workflows and how to understand how our deployment fits into this broader community of software development which of course is informed by a political economy of its own, but really the most humbling part of this whole process for me was learning how to communicate with developers as someone whose been outside of that for so long.

Speaker: I can't even communicate with developers. I get so wrapped up in my head that it all comes out the wrong way.

Speaker: Yeah. Kelsey, I barely even know what a (inaudible) is. I see Kelsey dropping in with what's a PR in the chat.

Speaker: Jessica asks would you like to get on the mic and speak up, Jessica? Or I can just read…

Speaker: Yeah my question is I've been around for a long time and my first open source software was the X-window system which I worked on a first distributed version of in 1984 and so I've been used to – everything I'd write working for the Smithsonian, everything I write is effectively open. The source is always available and I've been distributing it for a long time, 30 years and 25 on the web. I've done a website for a long time. My problem is I've been doing this for longer than most of the sites. My site has been accessible continuously (inaudible) with all the software on it. I have a different timeframe than most people currently writing stuff I think and so I'm curious – I think about the past and into the future although my future is limited, but I've been working over in astronomy. I literally work on long time scales, sort of the intermediate ones between the length of the universe and in orbit are close giant planet, exoplanet, so I'm just curious what timeframe people expect software to last. My feeling is now because people write in newer and newer languages that first old stuff gets lost and then second new stuff has an indefinite future lifetime. I just wonder what people think about lifetimes. Also for those who work purely with data what do you think about for your lifetime of data and data access, a digital data (inaudible). I'll let people talk about that.

Speaker: It's a really interesting question. I really personally really like writing line tools. I think in some ways there's no real time limits to those if it's the right sort of utility for people. As for other software its interesting. It is a really interesting question. I hadn't really thought about how long I expect things to last other than I anticipate various web based things. I just follow the sort of general rule of thumb there. Most things that use the web in some way seem to have like a three year timeline and then things on the backend of that might last longer, but yeah, given the choice I'd probably be writing command line and then just maintaining those over a longer period.

Speaker: I wouldn't mind jumping in. So I forgot to introduce this in my introduction, but I work in the open source side of Scholarly Publishing Ecosystems and my first experience in any kind of open source tool development came when we tried to estimate just how much money our authors at my university are spending to make their publications open access. So it involved a lot of the collection of non-machine readable data, cleaning it, making it machine readable, and as soon as we managed to run this tool for the first time everything was immediately out of date. I know one of the big things that as a community to work towards is allowing for the open access open source

supported federated search of academic publications without needing to pay for access to such databases. My biggest struggle in my work is the giants of scholarly publishers either purchasing open source projects and bringing it into their privately maintained infrastructure or like I mentioned just making it as hard as humanly possible for folks to work with their data. I wanted to know if you or anyone else out there has any suggestions for struggling against the behemoth machine. I know part of this is the incentivazation of open access and open source work in ways that we don't tend to see recognized as often, but I just wanted to know if anyone has any questions as to how we rage against the machine.

Speaker: I work with people here at the Smithsonian who run something called the astrophysical data system and I've worked a lot with at least one of the main people there. What we do is astronomical data has the advantage of having no monetary value in general. I had some great tapes from NASA that were labeled no monetary value from a space shuttle mission. Luckily in astronomy a lot of the publishers are nonprofits and we have issues with the big companies too, but we maintain a textual, at least image and text mixed database of all astronomical publications back to 1815 here and its (inaudible) around the world at several different places so its an interesting process. Most of the stuff that's published by a nonprofit becomes public after a year or so and they maintain it now. So it works pretty well. Its tricky linking nature and (inaudible) into this whole ecosystem. We have their abstracts always, but not necessarily their texts – unless you live somewhere you can get it the system links into the publishers database so its done pretty raw. The data that's in the papers, we think of data as stuff that makes the tables and graphs up. We're still working on how to maintain that in a reliable way, but that's the kind of thing I work with. So we publish archives and we have something called a virtual observatory which is a group which for 20 years has spent a lot of time trying to make protocols to get to data from different places just within our field. So it's doing it a general way I can't even comprehend. We're lucky because we have some standard data formats that have been around for like 40 years and my software that I distribute is based on that format and so it's really very widely used. I never know how widely used it is. I find out when people complain about bugs and I fix them or they fix them. So I wrote a whole – I did a presentation last fall on how to write shareable software of things I've learned over my career. I'll put a link in the chat to that.

Speaker: That would be great, thank you.

Speaker: Yeah I wouldn't mind piping in on the earlier question. I'm the founder of Artefactual Systems and I started the access to memory (inaudible) over 10 years ago when I saw it working as a consultant in the field of digital preservation and online access to archival collections, just saw the need for open source tools. I was very excited about

the open source movement and saw this small niche market that was really underserved by that. So on kind of a wing and a prayer I just started these projects. I got some small funding contracts to actually get these projects up and running, but then I had no clue on how to actually keep them up and running. At that point I was going to conferences trying to convince institutions to adopt the software and use it, but at the same time I had no clue whether I was going to be able to maintain that software. I quickly realized I couldn't do it on my own – that I would need more skilled developers onboard to do that and just more actual people involved in the whole process. I jealously looked at the west where there was Mellon grants and there seemed to be million dollar grants being handed out left and right for projects to start quite often for open source products and there was no equivalent in Canada, like not being based in a university essentially I had a private consulting company so I was considered a private entity. I didn't qualify for any of that money and at the same time no investor was interested in investing in a company that gave away software for free to archives and libraries, like a small niche market giving away free software. It was not – it's still not considered a good investment strategy. I essentially evolved the support structure around the open source projects as to the bounty program like it was just a matter of going from one contract to the next convincing people that the software that we had which wasn't CLI. Part of the whole goal was to make friendly user interfaces for archivists and librarians to use to give them access to digital preservation tool and bring them together in a single suite. So yeah the bounty program I managed to convince one institution after another that you might want to adopt this software and then maybe you want to fund the next (inaudible) for it. That in and of itself is how the company is established and where it is today is since then transitioned to offering maintenance programs and contracts for the software which is now part of its revenue stream. Quite frankly I burned myself out doing that. I hit the wall. I was doing that for two or three years too long. Like I just couldn't keep up the charade. It was so much work to keep contracts coming in, manage staff, starting at spreadsheets worrying where the next few dollars would come from to keep the whole thing afloat. So I was very fortunate to be able to hire some good people and I just said I can't do this anymore, like I'm burnt out, I need to check out and I passed the company off to some very talented people I had hired and they kept the thing afloat and grew it. Now I'm living the founder's dream where I'm able to come back and work as an employee working as an analyst on those projects and not having to worry about the management headaches and other people are managing those issues. But artifacts itself has established itself as a major player in this field, but now, sorry this is a bit of a long rant, but I feel its all very relevant, everything that's been said this morning so far related to my own experience, but now we're running into the issue of the code base itself being 10 years old and its holding us back from doing new cutting edge things and we run into issues like losing –

like moving from Python-2 to Python-3 is an example and having a really old unsupported relational map in (inaudible). So things that are holding out the code base that we need to evolve, but now we're in a position where nobody is willing to privately fund like maintenance of software. We've got hundreds of institutions now running the software and relying on it. Some of them pay us as service providers. The majority don't. They're just benefiting from the free software which is great, that's why we did it. We knew there wasn't lots of money in archival libraries and that's one of my motivations was to provide free software, but at the end of the day we can't get funding for somebody to say here's some housekeeping money so you guys can bring this up to the next version. We still have to find ways to put money aside from contracts to do that work and somehow now juggle contract work with being able to do the maintenance work and it involves a lot of planning. There's some major architecture analysis and decisions that have to be made that we're now – I don't know if Jessica has run into similar things with her projects, but that's a real challenge for maintaining free open source code is when you start hitting that first generation and having to move to the next. So anyway that's a bit of a rant, but that's my experience in the field.

Speaker: I'll just say the way I've been able to do this is sort of maybe weird, but I sort of think of myself as a tool maker. I do a variety of things, but most of it is low level, but I also work on projects that incorporate the tools. I'm also in a location where there are lots of other people that use it. So this all started when I learned how to use a system that I ended up not using for a big project – a space shuttle telescope – and when the money ran out on that they hired me because I knew all this stuff and I had started writing a project which this was in the late 80's and I'm still maintaining this project and people still use it. We wrote a paper in 98 and that paper is still cited quite a bit in astronomy. In the meantime I write data pipelines for newer and newer instruments and some instruments that have been around for a long time I keep trying to streamline the pipeline because I run it. It's been an interesting job. I'm on overhead and so as long as there are lots of grants coming into this center for astrophysics where I work I'm in pretty good shape. I have no kind of tenure, but I've been in the same job for 30 years now so it's an interesting situation. It used to be more common than it is now and I'm afraid younger people won't find jobs like this because they don't really exist very much and maybe they never have, but I've had several of those. This is my second pretty long term job. The first one was only eight years. It's been an interesting situation. I'm really reluctant to apply my situation to anybody else, but there are lessons and things I've learned along the way. What I write is pretty technical software so one of the things I just wrote about in something is that I worry that because the software is so widely used the technical information that's in it isn't known by people very much anymore. This has always been true with some software I wrote. We were at a conference in France in Strasburg where

you have to get their (inaudible) and there were all of us that knew how to use the world coordinate system which is published but still complicated to use. So the only people that understood it had to fly back to Paris on a small plane and everybody was joking that if we were on the same plane and it went down nobody would be able to maintain the software anymore because nobody would understand what's going on at the lowest level. So this is always a problem with things that are technical. It's spreading the knowledge around. It used to be in science that each new generation – well for the 1950's to the 1990's anyway – each new generation sort of wrote their own software so they'd have to learn stuff and then write the software. Just before the coming of the web it changed at least in my field and software became widely shared. So most of the things that people do in astronomy the software is open sourced whereas in my previous job in the 70's and 80's you didn't really share your software because the software that did your science for you, you didn't really want other people doing the exactly the same thing. Now there's so much data that everybody doing the same thing is much more common and sharing your software doesn't mean you don't get credit. That's the other difference is that if you work hard at it you can get credit for the software in the literature and so I've tried hard to do that and I've been a proponent for publishing in (inaudible) publications (inaudible) because you get credit for it, which helps your career. I can tell the people I work for that we need this, but all these other people are going to use it too and that will help the software do a better job for us. That's my argument. It's worked so far through five bosses because I've stayed in the same place, but the people above me have changed.

So we're just coming up to the end of our call now, but I want to close with one last thought. For those of us who aren't coders what are the not easiest, but lowest bar to entryways that we as individuals or as institutions can support open source work?

My favorite pastime when I'm not writing code is reporting bugs and issues. It's a forum for any of your software that you're using to report these things. Just put that information out there. First of all it lets people you're using the code, but second of all it tells others that you might be struggling with the same thing as them. I think there's a lot of value to just reporting issues and I think a lot of people tend to keep them under their hats.

Yeah, I do two things. I publish about my software so there are references and people cite it. I like that. I can tell people that I work for that these people are using it, they're citing it. And the other thing is my email is on every (inaudible) that I wrote and every main program that I wrote and so if people find bugs they can write me and all you have to do is read the (inaudible). Since its open source it's been around long enough that it gets to run on different architectures so it's usually compiled on other (inaudible) which is tracking (inaudible) no matter what. You don't even have to do a configure. So that's what I like. I like to get feedback. I think feedback is the biggest thing and if you cite it

then people you work for know you are using somebody else's software and if any kind of support comes up then that's a (inaudible) they'll know too.

Thank you so much for sharing everyone who has participated in the conversation today. I've been sitting here trying to chat in the chat and wrap up a call, but also look for a presentation that one of my colleagues is giving on software and code curation and preservation next week. I can't find it in time, but I will share the link with the list. Thank you, Ross, for being our special guest today and leading such a nuance conversation and bringing questions back to the group. I will post the link to the Information Maintainers Google group in the chat right now so for those of you who are on the Maintainers list and also want to be on the Information Maintainers list please sign up right there. I'm really excited to keep the conversation going. Thanks so much to all of you for attending, especially the folks who are showing up for the first time. It's really great to meet all of you. Once again I'm just so happy that all of you and myself – I'm so happy to be a part of this community every time I show up.

Thank you all.

Thanks everybody.

Bye.

End of audio