I1 - I see. From your point of view, what is sustainability in terms of software?

P - I guess software sustainability is when you get to a point where other people can use it and either they take on the role of keeping it up or someone's got to keep it up because it's a lot of time with the software. Somebody writes it and then when they're funding runs out they just kind of abandon it and if someone else finds it, fine. And then it's a take it as you want it kind of thing. You see it in a lot of research, I mean the project05 stuff I did completely just gone. The minute I left it, still sitting on gitHub but none even looked at it. So that's it.

I1 - Yeah

P - Project04 is working and it's you know I mean I only helped. Person01 did most of that, and it's one of these things that will probably stay alive for as long as the server that it's on stays alive and if that server crashes they will probably not bother rebuilding it into another machine. He's got, that has got census data in it, for the last few years but you know. Probably no one is gonna put the next year census data in and the year after that. So, how much longer that's gonna really be usable is really a problem. And we see it with Project02, which is something that has been running for ten years and it's, you know, so long as Person02 can keep putting money into it we keep it going we will provide support for people and everything like that but we are not gonna get much more money to keep it going. Someone else doesn't have to take it on and then if someone wants to use Project02 they might have to provide their own bug fixes and things, so that's you know. It's one of these things were they say, your software is free. Person2 would say it's like a puppy being free.

I1 - Yeah, you have to sustain it.

P - You actually paid to get it but you have got to put money in to get it going so.

I1 - Which attributes or features the software has that makes you believe that it is sustainable?

P - Makes Software sustainable… I think for small projects, it's sustainable if somebody else can pick it up, so it's got to be useful and useful not just for yourself but useful for somebody else as well. Which can be just pure luck, is it useful for someone else, does that person actually find it.

I1 - I see.

P - My PhD work kept going for three years because someone else did a followup PhD and found it extremely useful for his stuff. I mean now that he stopped, no one's picked up his work and probably no one will ever look at my PhD stuff from now on.

I1 - So it had to be findable.

P - It has to be findable, it has to be, you know. Someone has to find it, has to be useful for them. It has to be, you know, easy enough to use, I mean, if he find my code, and found that the effort of learning to use my code is going to be more difficult than the actual benefit it gave him, he'd probably throw away and write his own stuff.

I1- So usability of the code.

P - Absolutely Usability. It is . Usability I think that's when you get the kind of double hurdle cause you've got the usable by the novice so someone's gotta be able to pick it up and use it quickly. I mean, if you can't get something running within the first day most people will give up and stop trying. But it also gonna be usable long term cause if it's just the simple cases, people go yeah that's a really nice idea and then as soon as they start using it in anger, a lot blows up because it doesn't scale or anything like that. They ok, I will take the idea but I will write my own stuff. So, the

idea may be sustainable but the software isn't. So it's a very tricky balance to get right between the easy, you have it, the easy first hurdle to get it using and the actual power behind it.

I1- Easy to learn.

P - It's got to be easy to learn and to get something going yet powerful enough to actually do the job in the end.

I1 - I see.

P - A lot makes a difference whether it's a community behind it. A lot of the software that has been picked up, if people, if you get stuck you tend to ask, go internet. You google it. So the kind of stuff that's been picked up a lot you can just google it. Look at source forge and other place, where you can just ask, someones had the same question, oh there's the answer and then you keep going. But if you can't find an answer then it depends on the original team that wrote it supplying the answers, so as long as they are the software maybe sustainable, but as soon as the stop providing answers it get's very difficult to keep it going.

I1 - I see. Regarding the software you've developed, was sustainability a consideration?

P - In the PhD work, obviously no, once you get your PhD, work cares if anybody takes it on or not. The On project03 work I think some people wanted to keep it going but I came in quite late so. In some ways it was and is some ways it was not to be used.There was one bit of work we had to do because in the original grant it said we would do x. So it's like, we tried to argue well it doesn't even make sense now two and a half years later it's a project that doesn't make sense to do it anymore because it's not going to be used by the project, and so we just said, just put something up easily enough you know who cares. It was only actually because one of the other projects was interested in the same kind of things so, well if you gotta do it why don't you do it this way and then you can hand it over to somebody else so. In some ways that was not for sustainability. The project01 project there is much more of an interest in sustainability but, you know, a lot of sustainability work is also in getting continued funding to keep it going. Because just writing the software and putting it in a shelf or putting it up on gitHub or something like that doesn't make software sustainable. So it's writing the software, making it open source and everything like that. But I think that's the easy part, that's the part I've done.

I1 - So some of them were, some of them no.

P - Some of them no. I mean the project05 stuff, it was there for someone to use but if the partners who were going to use it weren't that interested in this, and it's like, that's it, it's just off the shelf so. But so it's different, I think really for software to sustainability is more about community building than just writing good software. Cause just writing good software with a good idea and putting is somewhere is not going to make it sustainable. As I said if nobody picks it up to keep it going or no one provides additional money to keep it going then it's not going to be sustainable just because it's sitting there.

I1- I see. In the software that you've developed was there, the ones that because sustainable, was there a specific point in time that then you decided it would be sustainable or was it sustainable from the beginning?

P - I think it's more.. You are looking at sustainability from the beginning.

I1 - From the beginning.

P - The project01 stuff when we decided to build that on top of another project to keep both that project going and the project01 project going so you look at sustainability from the beginning kind of thing. Obviously actual the drive to get funding and then things like that happens a bit latter on

cause you've got build a reputation before you can do that. You can't go one month into a project and start asking people for money for the next three years they kinda go well, show us you can do something first. But I think it's you sort of know at the beginning you which of the projects that really are sustainable and will keep going and which are the ones that, yes, the project05 was on a weak footing from the beginning, cause it was almost the project to keep something else going and then the underline project had not really delivered. Which makes it very difficult to build something sustainable on dodgy grounds.

I1 - I see. Have you worked on any projects that were not sustainable and were there any consequences of it not being sustainable?

P - I think unless you really write bad software, most software as software is reasonably sustainable. I think software becomes very unsustainable if you make it ultra-specific. If you have one very specific goal and you hard code it and everything you do for that one specific thing that it can become unsustainable. Because someone almost has to re-write the software for his own purpose. You get that a lot more with scripts and things like that. So I don't think my software as software was really unsustainable its more just that there was not an interest to keep it going.

I1 - Oh I see.

P - It's aways one of these balances how generic do you make your software. In the project05 I had tools for putting information into spreadsheets. I could've almost just handwritten spreadsheets to make them and well here's the spreadsheet and I just handwritten it. Then it's completely non sustainable because as soon as the data changes then If i've tried to write so it would it from configuration files without having to change the software. So I think the actual software could be sustainable, but there was not an interest from the partners to really use it there was not funding to keep it going. So as code it may be sustainable, it's freely available on gitHub but we didn't write the papers to advertise it properly. We didn't have the follow on. It's one of these things the code itself is sustainable. The project isn't. If you understand the difference between the two.

I1 - Yes. I've come across some of that already.
P - So in some ways, I hate writing papers. But if you don't write the papers. Papers are probably around longer than the software is. Fiscally the software lies there. Because people do that, you find software to do something, ok that looks good, and then you look last updated three years ago. Most people wont touch it.

I1 - I see.

P - Despite the fact that may be the right solution to their problem but if no one fixed it. Oh it's using old libraries and then is it really worth the effort of changing.