P: I see. Um, from your point of view, what is sustainability in terms of software?

P: Right. Um ... the thing about, the thing software, sustainable software is that it is software that, um, someone can use now and can use going forward in the future, and if it's truly sustainable, they can rely on it to be, um, in particular, I tend to take the view that software is truly sustainable if it is possible to have a small business that is, um, using, that is building, building their thing on top of the software because then, as they tend to be very, very committed to having, using that software, 'cause, because if the software goes away, then that business goes, and, and the, they become very, very keen on having software, um, kept going.

I1: Mm. (Affirmative)

P: Whereas with a large, with a large organisation, such as this university, um, there is, you can get sometimes, uh, spend, spend lots of effort on something and then just decide, well, yeah, we don't actually like that, and then go off and do something else. And that's happened to me a few times, so, uh, I tend to take the view that, um, what you really want us to have is to get some people involved in who, who are really committed to keeping in control the software. It keeps on going because then it's, then you've got a really good chance of being able to say the software will be, will continue to work, going forward, at your soft, at your software that you've written to use this library, or this tool, and the business process is that you build up around the tool and continue to be meaningful, going forward, even if it happens to be it slowly evolves, but that's okay, it's, it's one thing to have the things be evolving. It's another thing if it's just left to with a ...

I1: I see, it's kind of a comfy type of community.

P: It, community is, I think, uh, a very important part, yes. Um, but I'm a bit more specific than saying just there is a community.

I1: Yeah, yeah.

P: You've got a community who relies on it. That's the, it's that simple, I think.

I1: Um, which ...

P: I tend to prefer for it to be a is, isn't, is, a, uh, business community, because they are more likely to actually define funding for stuff, whereas an academic community can say, oh yes, we rely on this, but we're assuming it's all going to be free. I've seen that happen a few times as well, and we end up with some horrible fights over who will actually pay for anything to be done. Anyway ...

I1: I see. Which attributes, or features the software has that makes you believe that it is sustainable?

P: Well, um, well, I already said that. It's particularly the community, but that's not an attribute of the software itself.

I1: Right.

P: Um ... I think that's software probably has to be document, probably has to be documented so that, or, or I've had, or, or discoverable so that, so if someone wants to use it, they can find out how to use it, either by they just pick the software up and it just explains itself, or because there's some good documentation that explains how to use it. Um ... it probably needs to be reasonably, um, routinely written, and it needs to be possible to, for it to be, uh, for people to, other than the original developer to be able to, to, to compile it and reduce pa - package with it, and things like this.

It's not going to, it, for something to be sustainable, it's going to not be hold into one person. Um, so this tends to mean things like e-support, or automated tools of various kinds. That's it, that's the build process is very well described. They got doc - they got documentation, you've got, you've got tests, that, that helps and awful lot and, um, there must be sort of an example code, and things to make use of it. That's what the - that's the sort of thing you tend to want to have with a lot of sustainable stuff.

Aside from that, um, you tend not - the sustainability, it tends not to rely on software features, it's as such, apart from it, um, it tends to be more, it's, it's coupled to the software doing something useful, which, um, the, either there isn't an alternative for, or what it is much better in its niche than the alternatives, um. But, but specific features is something very simple for example. It's all very clearly features of the software itself, as opposed to features of the - it, it's much more to do with features of the community, features of with the things around the software, as opposed to the software.

I1: Okay. Um, regarding the software you developed, was sustainability a consideration?

P: Sustainability most certainly was a consideration. In the case of the software I've developed, on of the key things. Um, it's a piece, it's a piece of sof - uh, of service software, so one of the key things, that was a very early decision was to that it would not constrain what people would write to client software and everything. We would instead, um, go to more effort to make sure that the interface as it publishes our very clean and as much as possible to clearly described. So there's rec - there's recently extensive documentation, including not just what it is they can call, but also what is the pattern of methods you, how do you use these methods to achieve things? So, um, the idea is that someone can come along, read the, read the, um, documentation and then just go off and use it and, and get us going very quickly, um, despite the fact that this is very complicated software, actually under, underneath covers, but, I've done my best to hide the complexity. So I think, so I think that's, so that's very much concern, um, for how I've been planning to form making the things ascend, but also making sure these things, that, that, it's well, tried to make it well-structured, well-commented, and things like that.

I1: Uh, was it sustainable from the beginning, or was it a consideration at the developments that started?

P: Um ... I'd say it was certainly a consideration in the first couple of months of, of starting work on it.

I1: So -

P: So it, it was, right from the very - because about the first thing we did was we did a, um, a, if you will, a user market study to actually tell me what people actually wanted and things like that. And we should have filtered that through our ideas of past experience and of developing several software in the past to say nobody asked for this one, but our experience tells us that it is really important if we did this and things like that. The other things is it's certainly in to say, oh, having got this, basically, user survey down first, then there was just the art to, to prioritise, but yeah. With the, the original development of that software, there wasn't much certainty, um, a clear idea that you would live beyond the project. It was funding it at the time. So, yes, it wasn't even the current project that funded it. It's already sustainable across multiple projects, so, so yeah, it's, we, we do understand quite clearly what it was intended to be to migrate from being a - the output of a project to being a east of sustained software and product, if you want.

I1: Alright, um, have you worked on any projects that were not sustainable?

P: (laughs) Yes.

I1: And if so, um, were there any consequences of not being sustainable?

P: Uh, this, it, I, that's a very good question. I have worked on projects where the software, the architecture of the project's software was, um, ended up as being over 400 pages, so fairly dense, just, just for the architecture, not including the API descriptions, or anything like that. Just the architectural things, and that was completely unsustainable because it turned out that the software was impossible for any, anyone to actually deploy in full, and it would only work if all the pieces were deployed. Funny, that didn't work.

I1: (laughs)

P: Uh, we learned, uh, a lot from that, um, that one of the important things is that, uh, uh, you don't try and do a boil the ocean type project. (laughs) And then build a system that solves everything. It's that you build your systems to solve one specific task and to do it. They do their job, they do their job, even if the rest of the world isn't doing their job, and then you run into a clean interface, things like that, which I'll be talking about. And then, like that, it would be, like that, it can be useful all on its own. It has its own purpose, it has what it does, it has what it doesn't do, um, all the sort of plastic, um, things that your sort of do with the management type study.

Your software, you, you have the, you do your own thing and analysis of the things that you dos, don'ts, might dos, probably shouldn'ts, and you know, you go through all this stuff and do your analysis and then define very clearly, what is your market niche, and then you sit in that niche and you don't go off and do random other things outside. Um, that helps an awful lot because otherwise you get, uh, commence problems with scope and creep the scopes, but you have to be very careful to keep it. Well, at least it wasn't the sprit of what it is, of what it does. Um, that is, that helps a lot, but, uh, but, yeah, that other, that other project was just a complete disaster. Um, let's see, the key problems that it was too complicated. Too complicated, by far. It was, it was just a, that was, that wasn't a system for doing automated contract negotiation.

So with, uh, so with semantic negotiation and things like this and it would go off and, and if you had all the pieces there, then you could do automatic tracking of contracts, automatic, uh, SLA enforcements so it would go, and it could go off and allocate more machines, and it could go and it would and it could manage the expectation of cost, and all that sort of thing, it seems. And when it all worked, it was absolutely wonderful, because you, you just say I want this thing to happen, and even if you knocked out an entire, um, even if you knocked out, say, Amazon web services, which actually happened during our, um, by pure chance it happened during our final review. Halfway through, we lost Amazon web services, because they had one of their very rare outages and that's the, and the software just kept on going.

It, it had transparently flipped over to use a different system, which we had configured as a back up and it was really nice. We were really, really impressed. We were the - but, um, but it was too complicated. The trouble was it required, um, basically, it required the entire project to do another, to do another, it, entire project team to be able to do another deployment. You, and you'll never be able to keep that sort of, uh, large project assembled in that way. You know, it was it had to have partners all over Europe. It was just, just one of those things.

I1: I see.

P: But so, that's the big thing we learned, to keep it simple.

I1: Yeah. That's all.

P: Okay.

I1: That’s all I have for you today. Yeah, it's supposed to be really quick.

P: Mm-hm.

I1: But, well, thank you very much.

P: Okay.

I1: Um ...

P: Yes.

I1: Hopefully, Person01 will bring cake.

P: (laughs)

I1: She promised, but she didn't show up.

P: Yeah. Oh dear, oh dear.

I1: You say that.