Pīkau Name: Communicating well when programming (CT P0 5)

Video Name: Writing for the Next Programmer (EMP10-4)

Presenters: Tim Bell and Joanne Roberts

Tim: When people think about programming they are often thinking that you are writing a program for the computer. That's kind of true but you really of course are writing the program for the user of the computer.

Joanne: But the user won't be looking at what you are writing maybe.

Tim: That's true but they are using your program.

Joanne: Ah, yes.

Tim: So whatever interface you give them that's what they are stuck with. You're right, they won't see the program. But someone who does see the program is the next programmer.

Joanne: If they come to fix up something that's wrong in the code.

Tim: Yes. The way that most software projects work is that it may well be given to someone else either because you work somewhere else now or because you're on another project or you're in part of a team and someone else in the team has to work on it. So one of the things to bear in mind is that a computer program is also being written for the next programmer. I've got an example here of a program. I've deliberately done it using quite bad style. In particular you notice the variable names 't1' and 't2' you can't really tell what it does.

Joanne: No.

Tim: In fact it almost looks like a mathematical puzzle. It's quite confusing.

Joanne: It looks like algebra.

Tim: It does doesn't it. Funnily enough it's actually the same program we just worked on before.

Joanne: Thought it looked familiar.

Tim: In actual fact it does exactly the same thing. If we run it it ‘asks’... well bad interface, it just says ‘next’. It actually means the ‘next number’. I’ll go with numbers I can add up. So if I give it a 3, and then a 2 and then a 0, then it says ‘5’ is the total.

Joanne: Yay.

Tim: So it’s exactly the same program but really hard to read. We end up with rules around programming that make things easier to understand. The first one of course is to choose variable names that mean something. So this ‘t1’ variable, we can see in the count that it is the thing that is adding up the total. I’ll just rename it. It’s so easy to go with a short name but the next person can’t make sense of it. That’s going to be the total that we do. The nice thing with Scratch: you change it there it changes it throughout. Now suddenly the program’s starting to get readable.

Joanne: Yes.

Tim: You actually need fairly good literacy skills to think ‘what’s a good short word (or maybe two or three words), that describe that total?’. Even ‘total’ may not be a full description. I was using ‘grand total’ before. It could be ‘total of the cost’ or something like that because there will probably be other totals in the program it could be confused with. Then this ‘t2’ thing of course was just the last number entered. I’ll call it ‘number entered’. The fact that I even put the word ‘number’ in it indicates to you that it’s probably a number, rather than just ‘what’s the product?’.

Joanne: It wasn’t clear when the cat just said ‘next’. I was like ‘well, Joanne?’

Tim: Exactly. It could be a list of people, anything. Now I’ve written the program in a way that communicates to the next programmer. In the classroom situation it’s communicating to the teacher because again you look over the student’s shoulder and you wonder what on earth they are doing.

Joanne: And they make it work then they want you to help fix their bugs it’s like ‘I can’t actually read your code’.

Tim: Exactly. In fact even to themselves because they may have created variables and you lose track of “was ‘t2’ the total or was ‘t1’ the total which one was it?” and so on. By actually putting in clear words or phrases there that explain what each variable is they’ve clarified to themselves what they are trying to do.

Joanne: You could use the ‘comment’ capability of Scratch?

Tim: Yes, exactly. Every programming language allows for comments. So I’m going to add a comment to it. First of all it’s really good whenever you have a chunk of code to say what it does. ‘This adds up a series of numbers that the user types in’. I’m not thinking this through too carefully. It’s actually a lot like writing a sentence or a paragraph or an email. You need to do it well. I could have typed in there ‘add stuff’ or ‘does the adding’.

Joanne: Can you just say how you brought that ‘comment’ up because I only discovered this about a month ago.

Tim: Sure. I just right click on the grey area.

Joanne: Ah, right click.

Tim: And ‘add comment’. Or at least that’s the way I do it. There maybe other ways.

Joanne: Fantastic. I couldn’t find any other ways.

Tim: You can attach comments to the code but I’ll just leave it there. It maybe worth putting a comment here that says ‘Keep asking for numbers until 0 is entered’ or something like that. Again I’ve articulated to myself and to you as a teacher and to my colleague who might be looking at it further on what it is. Make sure the comment is in a place that is relevant there.

Joanne: It’s almost like ‘show your working’ when you are doing a maths test isn’t it? If you just put the answer you might get half a mark but if you put ‘show your working, explain it’, full marks.

Tim: It takes a little extra time but it can save a lot of time down the track. Quite often when people write programs they start by writing the comments. You imagine if those comments were there, that’s almost the design of the program. That helps you to know what code you want to have. The other thing with comments is that you can have too many. If students are told ‘make sure you use comments’ and they go ‘well to be really good…’ and they put a comment on every line. Just for fun I’ll do this. ‘Set the total to 0.’ Well that is true (if I can get it to stay there, it doesn’t want to go over the top of it). You could have comments all over the place. The point is that you can assume that the reader knows what the language does and you don’t need to explain it to them. Too much information can prevent people from getting the key information as well. You can see now that literacy, coming up with words, getting your spelling right, clear communication is really important.

What we’ve been looking at is summed up in something a guy called Alan Blackwell once wrote which is that “Many skills of a professional programmer are related to social context rather than the technical one.” Because programming seems like a very technical thing but as we see here it’s about communicating well, being clear in what you are doing. With HCI we are thinking about the user and how we are communicating with them. With programming we are thinking about how am I going to communicate with the next programmer? I think particularly for students it’s a good habit to get into. They often see programs that they do in class as a ‘throw away’ exercise. Languages like Scratch often you’ll see names like ‘Program 1’ or ‘Program 2’, they just go with those default names, ‘Sprite 1’, ‘Sprite 2’, whatever, which gets very hard to figure out when it starts getting complicated.

Joanne: It's making me think of the Y2000 problems when they thought all the computers where going to stop. I suppose there would have been a whole lot of coders going in there having to read someone else's code from 10-15 years prior hoping to understand and hoping that our toilets would continue to flush in the New Year.

Tim: Exactly. A huge amount of work had to be done then to identify everywhere that a date was being used and how many digits it was using. If someone had put comments in their code that said 'check the current date' or 'work out the age' or something like that then you could search for those and find out where dates were being used. Especially if the variables had names like 'date' or 'year' or that sort of thing rather than 'y' or 'x'.

Joanne: Yes.

Tim: With the example with Y2K you are communicating to someone in the future who might be looking at it. It might be yourself in the future who has to look at it. Also when people are programming they are usually part of a team (if not a huge team), and they have to talk to the other people, communicate which part they are working on. How that's going to work? How it's going to fit in with everyone else's? It's actually a very social exercise.

Joanne: A lot of collaboration going on.

Tim: That's right. So programming: primarily a social exercise, but also a technical one.