Michael Jackson

Interviewed by

Mark Jones

7th November 2018

At the

BCS London Office,
5 Southampton Street, London, WC2E 7HA
Kindly provided by BCS – The Chartered Institute for IT

Welcome to the Archives of Information Technology. It's the 7th of November 2018, and we're in London, at the offices of the British Computer Society. I'm Mark Jones, an interviewer with Archives of IT, and today I'll be talking to Michael Jackson.

His name has been familiar to huge numbers of software designers and developers for many years, from the mid-1970s onwards, and he's a pioneer, some would say a radical pioneer, in the field of software design and development. He published five books between 1975 and 2001, and has written numerous papers over the years. He is the J in JSP and JSD, and as I think we'll find out, the problems JSP and JSD tackled are still very relevant today: how do we make computer systems which are reliable, maintainable, and do precisely what they're intended to do.

Michael, thank you very much for agreeing to be interviewed.

Thank you Mark.

If you would like to make any preliminary remarks you care to.

[00:57]
Yes. I had the impression that a lot of the archive interest is in, is in the development of organisations and relationships, and commercial aspects, and larger social aspects and so on. I have been a very reluctant participant in business; even when I had my own company I was a very, a very reluctant manager. I jokingly said that it was management by neglect, but what it really, what it really reflected was that my interests have always been much more technical in the sense of program design, not technical in the sense of, of electronics or, so on. So I think, I need to, I need to say that at the beginning, so that you won't be, I suppose I'd say, disappointed at what I talk about in each of the questions you ask me.

That's fine. Thank you for that clarification. I agree that to date many of the interviews have been around business and the development of companies, but I think the, one of the intentions of the trustees is to broaden it a little bit, and interview more about the technical pioneers of the period as well as the business and commercial pioneers. So I think, hopefully what we'll talk about is a blend really of…

Good.

…your commercial and business history, together with JSP and JSD, and, problem frames and so on.

Yes.

Thank you for that.

Good. Good.

[02:42]
So if we go back to the beginning, if that's all right. I believe you were born in Birmingham in 1936?

Yes.

Parents, Montagu and Bertha.

Yes. Ah, ah you see, this shows. And I bet you know my bank account number and my password as well. [laughs]

No no, that's as much as I know.

You stopped there, you stopped at that point did you? [laughs]

Yes.

Yes.

Do you have any brothers or sisters, Michael?

I have a sister, Joan, who is two years older than me.

Right.

Yes.

And, I'm guessing because your first school was in Hampstead that your parents probably moved at some point. Is that right?

Well, yes. They moved to London in 1938, when war was declared, and people started being worried about what that might mean. 1939, late 1939, my parents had family in Scotland, and we went up, and spent a lot of time with them. And when the Blitz came, my mother, my, my father was away a lot of the time, my mother didn't want us to be evacuated in the traditional mode of being taken to Paddington station with a label round our necks, so she did a private enterprise evacuation, and took my sister and me travelling around, Scotland, and all sorts of, all sorts of places in England. And then we came back to London, and then when the, you might call the second Blitz came in '44, we went off again, and lived in Cheltenham, and so on. But when we came back to… During this time, apart from a little interlude in Edinburgh when I went to school in Edinburgh for one term I think it must have been, my education had basically been by local tutors that my mother was able to rustle up for me and my sister. And then when we came to, back to London… Well, I, I should say, I think I, I've missed something out of the record here. The first school I went to in London was Henrietta Barnett, which, as I'm sure you know, is a girls' school, but they took boys up to the age of eight. And, so, when I was eight, and I couldn't stay at Henrietta Barnett, and, I went to The Hall in Hampstead, which is quite a well-known school, people in this area of London.

I found some interesting alumni of that school actually.

Yes.

They weren't contemporaries of yours, but it shows the broad range of people.

Yes.

Oliver Letwin.

Yes.

Giles Coren.

Yes.

Clement Freud.

Yes.

And Nick Mason.

Yes. Yes. Yes.

Anybody who you remember from school particularly?

Brian Lapping, the TV journalist, producer, historian I suppose you would say. Nick Faith, who was a well-known journalist in his time. He's died recently, sadly. That's all I can think of at the moment, but, maybe there are others I should have thought of.

[05:58]
It's all right. So after there it was Harrow.

Yes.

Was that your choice, your father's choice, or…?

[pause] It was… Actually I tried at another school first, to get a scholarship. When I was born, in 1938[sic], my parents had no intention or thought of any kind that their newborn son would be going to a public school. And, so, when the time came I didn't have my name down for any school whatsoever. And I tried at one and didn't get in,

and then I tried at Harrow and did. And I think, it was just that they had rather perhaps enlarged their ideas about where they might send their children to school.

Did you board at Harrow?

Is that b-o-a-r-d, or b-o-r-e-d? [laughs]

Well I'll come to the other bored in a minute perhaps really, but, no, I was thinking of boarding as in day versus boarding.

Yes. [both laugh] Yes. Yes, yes, I did. And, from what people are accustomed today, it was quite, quite radical. There were, essentially, three terms of thirteen weeks each. There was no going home, not, not at the weekends or any time at all. You were, you were just there. And in many ways I think, that was very good, because it meant that, I mean I was in a house with 50 boys, which was sort of average at Harrow at that time, what it meant was that you had to come to terms with the other 50 boys in the house. And, I think, I had been rather, I don't know what I'd say, consider it precocious, and, bumptious, and, I don't know what. So it probably rubbed a few sharp corners off, I'm not sure.

Mm.

But I enjoyed it greatly. At the end of every term I wanted to go home for the holidays, and at the end of every holiday I wanted to go back to school for the term. So, very satisfactory. I liked it, yes.

[08:03]
So at what stage there did you have some idea what you might like to do as a career? Well you obviously… You went on to read Classics, so at some point you must have thought that that was where your abilities lay.

Ah well having… I've explained that already. [laughs] I mean this… It was… It was absurd. I probably didn't even know what a barrister did. But, this was a decision forced on me, or, me and my family, my parents I suppose, by the absurdity

of English education for people of my age at that time. School Certificate, as I think it was then called, if I'm right, was something that all young people were expected to enter and qualify in. And then you would get a certificate. But there was a short period, I don't know when exactly it was, but it certainly was, the regulations of that period were in effect, in, I think 1949, '50, '51, that sort of time, and the regulation said, you cannot be given a School Certificate, the bit of paper, until you are sixteen. So you can't take the exam and, and be given a certificate of your performance in the exam till you're sixteen. So, although, although we did the School Certificate, and did the exams, we didn't get a certificate. And, I went into the classical lower fifth, because, I suppose of the two, the two possibilities, I don't really remember, perhaps the thing about a barrister as what mattered. I had a, I had a facility, quite modest looking back, but some relation to the time, for mathematics. I mean I was the sort of child I'd say who could understand immediately the proof of elementary theorem and Euclidian plane geometry or something of that sort. So I, it was a choice, and I, I made the choice. And so that, that's, I went into the classical lower fifth. But, that created a situation, which I think I may have mentioned in the timeline elsewhere, which was that, there we were in the classical lower fifth; the following year we would be in the classical upper fifth, and the following year in the classical lower sixth, and the following year in the classical upper sixth, or sixth as it was called. But somewhere along there we had to do the School Certificate. And you couldn't say, well I'll just do it in Latin and Greek; you had to have quite a range of subjects. So there was a need to maintain some familiarity with the subjects that we had already passed School Certificate in but hadn't been awarded the certificate. And one of those was mathematics. And, the mathematics teacher, that was either chosen for this task or the fates decreed would be his task, was Christopher Strachey. And, he didn't think that we needed to brush up on our elementary differential equations or anything like that. He thought we ought to know about [inaud] and [inaud], and all sorts of wonderful things like that. And it was a, it was a great revelation, it was marvellous. Brilliant.

[11:56]

I think I read somewhere that you wrote your first program. Is that a slight exaggeration, or is that what happened, with him?

Well, it depends. It depends how you define the word program. What Christopher did was… At the time he was working, well he was a schoolmaster at Harrow, but he was also, before or after, or during that, working at the MPL, National Physical Laboratory, and he had a lot of involvement with the people in Manchester, who were making the, I'm not sure of the chronology, the Manchester Baby, the Manchester Mark 1. And he wrote a, he wrote a program to play draughts, chequers, which was, which was quite well-known, because it was quite a feat. And I remember the delight with which he explained to us that his program was quite correct, but his specification was wrong. And, without delving too deeply into the meaning of the word specification, he had written the program so that when the opponent was about to make a king, I'm sure you're familiar with this, the program would give its pieces away, because that was as good as, it meant you were one down, and that was as good as letting the opponent make a king. And, so he told us with great glee that his program worked, but the specification was wrong. So, why am I saying all this? Because he thought we ought to learn about programming, along with Klein bottles. And, so what he did, I, I realise with hindsight, was rather clever. He devised a very simple, a very simple programming language for small integers essentially, which could be values of variables, which you could name A, B, C and D. And, he imagined, or perhaps he was describing the Manchester Baby, I don't know, he imagined a machine that, for example, I'm not sure this was exactly right, but for example, didn't have multiplication, and didn't have division, it only had addition and subtraction. And it didn't have comparison of two numeric values to see which one was larger, and so on. And he set us to write some programs. So… So I, I wrote programs in his little programming language, all multiplying two integers together, and dividing one integer by another. I think the, the summit of my achievement in the programming world then was comparing two integers, and what I did, which I look back on with, not very well justified pride, but nonetheless, it's pride, was, I made a program which alternately… Now what did it do? [pause] Or maybe… [pause] No, I think, maybe it was to say whether an integer was positive or negative. That's right. Yes. And it made a copy. So you had two copies. And, alternately, it added one to one copy and subtracted one from the other. So you would see which one got to zero first. And I, that's my first algorithm. [laughs]

Very good.

Yes.

[15:35]
So after Harrow, it was on to Oxford, and Classical Mods and Greats.

Yes.

For, perhaps for those who don't know, what does Classical Mods and Greats consist of?

Right. Well, Classical Mods was Latin and Greek, Classical languages, and Latin and Greek literature. That was, that was basically what it was. And one learnt to do things that the seventeenth and eighteenth century schoolboys did, so, the nineteenth century as well, so, why shouldn't we do it in the twentieth century? Learnt to translate English into Latin prose, and Greek prose, and, Latin hexameters, and Greek archaic texts. So, all that. That was, that was Mods, and that was five terms. It was a four-year course, three terms in each year. And, Greats. You were expected to continue to maintain your ability in the languages, and, something perhaps of literature, but, it introduced history, Classical Greek and Roman history, and Classical Greek philosophy. And also modern philosophy. And of course, this was, this was 1958, '57, '58, so the linguistic philosophers like J L Austin were hard at work at Oxford, so there was a lot of that. And in the finals paper for Greats there was a paper called Logic. It was quite interesting I think in a way. The Logic paper was not an exercise in, in propositional or modal logic, or higher-order logic or anything like that. The Logic paper was a paper, which had questions like, question seven, 'Is question seven unfair?' And question nine would be, 'Does God exist?' And, it was essentially an invitation to write a kind of, I suppose they were hoping for some kind of demonstration of intellectual agility, or, something. But it was a lot, it was a lot of fun.

I saw somewhere a quotation which said, 'The Mods examination has a reputation as something of an ordeal.' Did you find it so, or did you…?

Well the examination itself was fourteen papers, I haven't forgotten this, was fourteen papers, of which twelve were three-hour papers, and two were an hour and a half papers. And they were done over something like, seven days. So… It's something like, something like that. So that was…

So that was like an ordeal.

It was, it was quite hard, yes. Yes.

[18:20]
So after Oxford, you obviously decided to change tack a little bit, or maybe pick up the maths that you had perhaps left behind?

Yes.

And went to Cambridge.

Yes.

So how did that come about?

[pause] My father had been aware, was aware then, that I had made this decision between mathematics and Classics, or, we had made it. And he realised that I still had some inclination fir mathematics. And he said, in a fit of extraordinary generosity, he said, 'If you can find yourself a place to read mathematics, I'll pay for it.' And… Because, undergraduate education was free then. And, so, I, I managed somehow, probably by, by a little bit of really quite disgraceful influence among tutors or something of that sort, to get myself an offer of a place at Trinity Hall. And, I wrote to the University Joint Recruiting Board secretary, Mr Jolly, j-o-l-l-y, and said, 'I have a place to read…' This was quite, quite naïve, there was nothing disingenuous about this at all. I wrote, and I said, 'I have a place to read mathematics at Cambridge. May I defer my National Service?' Which of course, everyone knew was going to end, this was 1958, in the next year or two. And we got back a rubber-stamped letter saying, 'Your National Service is deferred.' And again, no naivety, and no devious tactical

calculations. I went off with my friend Andrew from school, and we hitchhiked, his parents lived in Brussels, we hitchhiked around Europe and up into Norway, incommunicado. I mean telephone was unimaginably expensive, no Internet and so on, so forth. And then, when I came back, in early September, there was a pile of letters waiting for me from the University Joint Recruiting Board saying, 'Hang on a moment, hold on a moment.' But we persuaded them that having granted me deferment in June, they couldn't reasonably withdraw it in September. And so I went up to Cambridge.

Right.

I'm being too verbose aren't I.

No, not at all.

This will take us forever if I go on like this.

[20:53]
No, it's all right, we'll… We'll pick up the pace I'm sure. It's all right, it's all right. So you enjoyed the maths degree? And I found maths hard Cambridge I have to say.

Well, I found it very hard as well. I… I did work during what remained of the summer vacation very hard. But frankly, my applicable mathematics education, there weren't any questions in Part I about Klein bottles, and my applicable mathematical education had stopped when I was, literally when I was thirteen. And, if they had had a little bit of elementary differential analysis, and a few Euclidian plane problems, I could have, I, I could have swung it. But, it was, it was too hard for me. I did, I did go through, I did Part I, and I did Part II. But, in a way, it was, in a way it was disappointingly hard, I found it.

You achieved nevertheless…

Well, sort of. Yes. I did get a degree, yes, I did get a degree, yes.

[22:09]

So after Cambridge, what did you, where did you go? You needed a job I guess.

Yes. I went to work for an economic and industrial consultancy. This was, 1961. And they had understood that there were some new-fangled things called computers. They knew that I was interested in them. Not that I had any particular knowledge about them. And they thought, what more can you want? An ignoramus who's interested in computers. That's what we need. And, eventually they, well they started by sending me on a programming course for a truly wonderful machine. I guess this was rather before your time. It was the Bull Gamma 300. Bull was a machine, Bull is a French, was a French computer company. And this was an electromechanical machine. It had the same kind of hardware technology as tabulating machines. It was very sort of, rotary switches and relays and, all that kind of stuff. But, it had a generalised design in the sense that, the main feature of the logic of the machine was an array of, I think they were called selectors, which were hubs which had, I think, twelve sockets into which you could plug the end of a wire, and they had a, each hub was identical in specification, and it had a specification such as, this isn't quite right, but it's right in spirit, if you put a pulse in at socket twelve, at, one time, then, two ticks later, you will get a pulse out at three. I mean it was that, it was that kind of specification. And with this you could wire up a plug board, which could do all sorts of things. And one of the, the memorable example was that, they had a card reader with five reading stations and a punch station. So that the card went past five reading stations. But you had a calculation which you could do in five cycles. You could multi-program the calculation four or five times. So that as the cards moved along, each card got a cycle of the calculation as it went to the next reading station. So, it was a, multi-program. But, of course, I mean 1961, I mean the 7090 as out by then, or certainly the 709, and so on. And it was hopelessly unreliable. And, I don't think anyone ever really made a working system. There was an insurance company which sent people on the course and…

Mm.

[25:11]

So that, that was the, that was the first thing. But that wasn't, that wasn't much effect. But then they sent me to a bank, Philip Hill, Higginson, who were transferring a share registration system from an IBM punch card system, 407 card sorters, and, collators and those things, to an IBM 1401. And, this was an experience just like learning Christopher's programming, which I thought, this is the water, and I think I'm a duck. It was wonderful. The 1401 was just brilliant. Do you know anything about it?

No. No.

It was a very simple machine. It was a character machine. The programming language was literally the machine language, although there were mnemonics for some of the operators. It was, it was equipped with, believe it or not, mixed-radix arithmetic, so that you could do sterling calculations. It had a 4K 6-bit core store, and you could do, you could do lots of lovely toy things with that.

I see now why there's a quotation, which I think you produce much later actually, related to this time, which said, 'I was a careful designer. I realised that program design was hard and results likely to be erroneous.' And the sort of programming you describe is very close to nuts and bolts isn't it, so…

Well, it, it was very close to nuts and bolts, but actually, it was the next experience, which I'll tell you about in a moment, that led to that quotation. The 1401 was, was so simple, and, and the store was so small. I mean the biggest 1401 was 16K, and this was 4K and 6-bit characters. So when you printed out a core dump, you could see the whole thing. You would see, here's the program, here were the instructions, the machine code is what you programmed in, so… And it was like that. And it was small enough, it was small enough, and in a, a slightly quirky way, well enough designed to be, I think, actually really easy, really easy to program. And you couldn't make too much of a mess in a machine that size.
[27:37]
But, after, after that experience, in which I worked with the IBM salesman to develop the programs, do their share registration, and I, I operated the machine for a while when it arrived, because the operator who had been hired hadn't come yet, and I just had a whale of a time. I thought it was wonderful. And it was, I mean it was

wonderful. But that came to an end, of course, eventually, they just wanted to get on with their share registration. And, I went to… As a, as an employee of this industrial consultancy, I was sent to the NAAFI. Does that name ring a bell with you?

It does indeed.

Yes. And they had a payroll. So, I'll spare you my two-week account of the NAAFI payroll, and I'll just condense it to the eleven-hour account, or maybe I can make it a bit shorter. The NAAFI payroll was of course payroll basically for the people who worked in their canteens. Their pay was subject to the Catering Wages Act. The Catering Wages Act recognised that people don't eat meals only between nine in the morning and five in the afternoon. So, the question is, how do you define overtime? And the answer is, at the beginning of each week the employee makes an agreement with their manager about which hours they will work. Hours outside those agreed hours are overtime, whether the total adds up or not. So that, that's the first point. But it meant that, you had to record two lots of hours for each employee each week, or each day. And furthermore, the input to the machine, which I'll describe in a moment, was on punch card, and they had a fine array of, I'm sure you must have seen them, those nice steel bed machines with a travelling carriage, with little punches, you punch holes in the card.

Yes.

So they had to punch all that stuff twice. So, so that was the Catering Wages Act. They were required by their charter to set up a canteen wherever there were 100 military personnel spending the night. So if you sent 101 people out on Salisbury Plain, there had to be a canteen, which came into existence, a pop-up canteen, that's what it was. [both laugh] Finally, there were the ships, the navy ships. Radio communication with the ships was of course at a premium, was needed for things much more serious than payroll. So, what they did was, they paid their people, and recorded it, and brought the records back. I don't remember quite how they brought them back. I don't suppose they actually punched cards or… But they brought the cords back somehow. And, they had paid them, because they had to spend money, on

board ship or at the port. And, when they brought it back, we had to recalculate what had been done, and make the adjustments necessary.

[30:58]

So the reason that I give you this preamble about the NAAFI payroll is that, you can imagine, there were a few complications in the, what today we would call the specification, or the requirement, or something. And, also, well not, not also, this, this was run on a machine called the Honeywell 400, which was a 48-bit word machine, much bigger than the 1401, with lots of tape drives, no disks. And, and it was very, it was very nice. But it was quite quirky. But it had a, it had a teletype operator's console, and from this console the operating system of the Honeywell 400 allowed you to inspect what was in storage, and to change it. So, I wrote these programs, and, and I, I was, I was given a free run of the machine at night, so I had to work nights in order to do this. And, I became aware, as you said in the quotation, that, it wasn't as easy as all that. So what I did was, I introduced… This is 1963. So I'm, I'm entitled to claim some contribution here I think, not that anyone ever knew about it except me, which in, in the terminology of the better educated computer people would be called, introducing assertions. I introduced assertions into the program. And I introduced, of course, naturally, checks to the assertions. And if a check failed, the console, typewriter, typed out 'chaos'. I can still remember it in the funny font that teletype machines had. Said 'chaos', and, and the storage address at which the program had stopped. And when they ran the payroll, which of course had to be… It was a real time system, you had to get it out that day. I would be sitting in the observation room looking at the operators operating, I mean, the operators… [laughs] So I would go, and it would say 'chaos', whatever. And I would sit down, and, oh my God, you know, I would get out my listings, and… And, during the time this continued, which I think was for a few months, I think we got the incidents down to, I, I can't claim zero, but it, I think I was off the hook eventually.

[33:46]

And, as I say, that was, that was when I realised, this programming lark wasn't as easy as it looked at first.

Mm, I can imagine. Sounds quite stressful actually. Although challenging too.

Well it, it was. I was. And it was… But it was… I just loved those machines. I just loved them.

Yes. It's problem-solving isn't it.

It's problem-solving, but it's also the joy of your train set. And on the, on the share registration problem, one of the things they had to do was to print out dividend cheques. Now there's only one printer, so you had to put in the stationery with the cheques for, initially Barclays, Lloyds, Midland, National Provincial, and Westminster. So there were five runs you had to do. But the tapes of the shareholdings and the details of the shareholders were sorted in shareholder ID number, probably something to do with names. But certainly not like this. So you had to make five passes through the, the file. However, the tape drives on this 1401 were, were really lovely, but they were fairly slow. And in particular, to rewind a tape drive, I think, I think I'm right here, there was a slow rewind and a fast rewind. If you did the slow rewind it took thirteen minutes to rewind it. And if you did the fast rewind there was a good chance that when you got to the end of the tape it would put the brakes on. The wheels went round much faster of course. And put the brakes on, and, under angular momentum, the tape would continue to rotate after the hub had stopped, and it would make zigzag holes in the tape somewhere. So, that was fast rewind and destroy. So you couldn't do that. So how could you, how could you do it? So here's the answer. This is my third. I'm remembering my triumphs. When you print the Barclays cheques, you write a copy of the tape, so, there were two tapes, and you're writing a second copy of the master tape. When you get a quarter of the way through the tape, I think it was a quarter, you stop writing the copy, and you slow rewind the copy. When you get to the end of the main tape, the second tape is at its starting point. Now to do the, the Lloyds cheques, you print them from the copy tape, but for every cheque you print, every record you look at, you look at four records from the master tape. So by the time you get to running out of the copy, you've caught up on the master tape. And I remember sitting in the, sitting in the room, watching those taps going backwards and forwards, and thinking, I'm thrilled, this wonderful.

Mm.

And it was wonderful.

Yes.

It was wonderful. I loved it. I really loved it.

[36:58]
Yes. So, 1964.

Mhm.

Maxwell Stamp. You left Maxwell Stamp. I was going to say that came to an end, but actually, they still survive as an organisation I realised.

Do they?

Yes.

I must admit, I haven't checked.

Yes.

Yes.

So, then on to Hoskyns.

Yes.

That's presumably the very start of Hoskyns wasn't it, or very soon afterwards?

The very, the very start. John Hoskyns, and John Pearce had decided to start John Hoskyns Limited, and they were looking for a couple of other people. And, because John Hoskyns, or his father-in-law perhaps, Mott, I think, knew Maxwell Stamp, he

got involved. Max Stamp said, 'I've got a lad that can write programs,' and so Hoskyns said, 'Good.' And also, Alastair de Watteville, who started Centerfile, you know the name?

No. Mhm.

And so, the four of us started. And there was no, there was no office. I dare say John had an office at home, but, and he had a secretary I think, must have had a secretary. And we just got going. And then it, and then it grew.

So, I mentioned before that I've spoken to Ray Harsant, a month or two ago.

Oh, right. Yes.

Ray said at the time he joined, in I think 1968, it was a rapidly growing company, expanding hugely. He said one of the skills that John Hoskins had was raising money to invest in the business.

Yes. Yes, American Express.

Yes, that's right. So it sounds like it was very busy, very fast-moving, lots going on.

Yes.

Would that be true, or is that just hindsight suggesting it was like that?

Well, no, you see this comes back to what I felt obliged to say in my introduction. I think, I was in some sense responsible for, I don't know, a programming group or something that you might call that. And John Pearce was responsible for Hoskyns systems management I think it was called, which engaged in what was called facilities management. Well Ray must have told you all this. Yes. And that was a business enterprise. I mean they had plenty of good technical people. This was a business enterprise. I never even understood that what I was doing was supposed to be a business enterprise. I thought it was just more, more programming fun.

[39:18]
And so, this soldiered on for a bit, making our tiny contribution to things. And then in 1970 there was an explosion of enthusiasm for IT, and, American Express… I remember John, John saying something like, 'You know, the world's gone mad,' he said. 'Nowadays, you can float your company by going public at 70 times last year's losses.' [both laugh] And… And you know, and that was his, his joke. I mean it was probably true. [laughs] And, American Express bought ten per cent of the shares. And for various reasons, partly I guess a misjudgement on John's part, I had some shares. And so I got ten per cent of the value of my shares from American Express. And, that was quite a lot of the, the that much money. In today's terms it wasn't gigantic, but then it looked gigantic. And then it all turned sour. I mean, in the world it turned sour, in the business it turned sour. And at the end of 1970 things were looking actually rather bad. It was… It wasn't, it wasn't quite 2008 but it was, or 7, whenever it was, but it was, it was getting that way for us. And there was a sort of reorganisation, and I was reorganised out of existence. Which I thoroughly deserved. I don't think I was fired, but, it was, you know, you can assist the second under dishwasher, it was a bit like that.

Mhm. Mhm.

[40:57]
And, so, clutching my American Express cheque, I thought, I'm off. Michael Jackson Systems Limited. That's what we want. Ah! [laughs]

Aha.

Ta-ra! [laughs]

Write the brochure. So along the way, you had met Barry Dwyer, I'm sure.

Ah, that was at Hoskyns. Yes.

Yes. I know that name, but, a bit of research, I couldn't find out much about him.

Australia. Australia.

Right.

Yes. Adelaide. University of Adelaide.

So you and he were, co-conspirators at Hoskyns were you?

Well, I would say… He was, he was someone better educated in computing than I was. I didn't really have any computer education at all. But he did. And, I suppose I could say, I had various intuitions, some of which were useful, and probably, most of which weren't. And he could add some sprinkling of actual knowledge and wisdom to it. So we cooperated in that sense, yes.

And the work he did, was that the very beginnings of JSP?

Well, it, it's hard to say. I think we… The things that Barry and I did at Hoskyns, in the latter part. We started something called segmented-level programming, which was really just, it was really just top-down programming with flowcharts and the modules. And, we actually went and gave a seminar about it in New York, and, did various things like this. But, gradually things got a bit more intelligible, and certainly the, some of the early ideas at JSP, certainly must, must be attributed to Barry, there's no doubt about that.
[42:55]
And then when I left, at the end of 1970, just in time to catch the Post Office strike at the beginning of 1971, when I left, I didn't quite know what I was going to do. But, I thought I was going to do this kind of thing. A pre-processor, a COBOL, that allowed you to do some things quite well. And, I was dithering a bit about this. And then, John Hoskyns asked me if I would go and do a consultancy job in the States for a company called Martin Marietta Data Systems.

Michael Jackson Systems Limited.

Yes.

Did you have a particular objective in mind when you set that up, or was it merely a vehicle for pursuing whatever interests came along?

It was the latter. Yes. I realised that I did need somebody else, and, I made an offer to a fellow called Brian Boulter from Hoskyns. And, my offer to him was essentially, 'I'm not sure what we're going to do, but, you can do it for a year, and I'll pay you this much. And maybe at the end of the year I'll say, "I'm sorry Brian, but, the game's up, can't keep you," or we'll carry on.' And in fact, he stayed until, I'm sad to say, his death in 1982 or 3.

Mm.

Mm.

[44:27]
Mm. So, first day in the office then. Did you have an office to start with, or…?

No. I had… Where are we? Let me see. Scott Grove. Well this is Scott Grove[sp?]. Yes. I, I think I built, I built a little extension to our house so that I could have an office. Yes. So I had the office. It was just a bit of my house.

So you sat there on day one, thinking, what do we do, did you, or did you have a whole load of things you already wanted to get done?

No, I had more… You see that was typical of me. What I was saying earlier. I didn't think to myself, you know, this is a marketing problem, and it's all about marketing. I must explore the market, I must survey this and that, and so on. I thought, what's interesting? And the idea I came up with was, was this, this thing here. It was a pre-processor of COBOL programs. And, I thought, well yes, we could do this. And in fact, that's probably why I got Brian on board, because he would be able to do a lot of the programming. Not that I couldn't, but I did have the sense to see that I needed someone else as well. And I've still got, I've still got documents he wrote about it, in '72 I think, something like that.

Mhm.

And so we did this. And, commercially, it wasn't a success at all. But, the programming courses were going, were actually going quite well. And, somehow, while we were doing this, we drifted into, into giving programming courses, and they acquired a certain amount of notoriety among some groups of people. And so, we found ourselves building up a network of licensees. So we had licensee companies in several countries in Europe. And they, they gave our courses; they sent their trainers to our, our trainer's course.
[46:35]
And then, by that time, JSP, which was named by the Swedish licensee, it wasn't our idea, was well enough to find to be able to write a pre-compiler for JSP programs. And that's, that's what we did, and that became the main software offering.

Mm. So that's when you wrote your first book wasn't it, as well, 1975.

Yes, *Principles of Program Design*.

Was that a marketing thing really, or did you just want to write a book and…?

No. What actually happened was that, some word of the programming… I should go back a little bit. At Oxford, I was up at Merton, and I was two years behind Tony Hoare, you know of, and incidentally, he introduced me to the idea that logic wasn't about questions like, does God exist? It was about, what can you do with predicate logic, or… And, so I, I was, I knew, I knew him, and he knew of me and so on. And, through him I think it was, there were some sort of whisperings about program design among the academics surrounding Tony Hoare and Edsger Dijkstra and, Cliff Jones and various other people. And I was invited to a meeting of WG2.3. Do you know about WG2.3?

Mhm.

In Newcastle. So, I went, and I gave a talk, about, about program design. And, constraints were not very severe in those days, so they invited me to join, and I did. [pause] I've completely lost the reason I'm telling you all this. You asked me a question, and I've forgotten what it was.

I asked you about… [pause]

Oh the book.

Yes, yes that's right.

The book.

The course book, yes.

Ys. So, so I had reason to describe the, the method to them, and the courses. And so I had done quite a lot of writing about it, and I gave many of the courses myself. And, they were, they were pretty strenuous. You had, 30 or fewer people in a room for a week from nine to 5.30 every day. The AV equipment was a flipchart board, and subscribing to the view, which, I don't know whether I learnt it from Dijkstra or not, you have to write everything fresh in a course, because, apart from anything else, apart from its effect on the content passing through your mind as you write it, it also means that you can't write faster than they can read. And, and so, I gave many of these courses, how many, I don't remember, but it was quite a lot, in several, many countries. And, so, along with the manuals, the course manuals, and the manuals for the training course, and stuff I had started to write in the fringes of the academic world, in a way the book was, was ready. And Tony Hoare, who at that time was the series editor for the [inaud] series in which, the structured programming book with [inaud] Dijkstra and Hoare [inaud], said, 'Why don't you do a book on your program design course?' He suggested the title, *Principles of Program Design*, so I didn't have to worry about thinking about that. And, I delivered the book. And, it was printed, and some copies were sold, it was wonderful. Mm.

[50:58]

So JSP was different to any of the other methodologies around at that time wasn't it?

Yes.

Did you find it a hard sell so to speak, to convince people this is a good way of doing it, the good way of doing it? I mean did you pitch it as one of many, or did you say, this is what you should be doing?

No, I'm, I'm far too egotistical to have pitched it as one of many. That's, that's just not me. Partly because I'm an egotist, but to be, to be fair to myself, because I recognise my limitations, and I'm not sure that I could give a satisfactory account of anyone else's work, especially if anyone who knew about it was present. So it was just, it was just about, about JSP. Of course it, it evolved a bit over, over a period, but that's, that's what it was about, yes.

Mm. Mm.

And… So… I like to think, I like to think that all the problems that were commonly discussed in courses on modular programming, program design, and so on, in those days, that for many if not most of the problems that people actually dealt with, it was, it was actually really perfectly good. I wrote a piece in the… There was a, there was a conference called 'Software Pioneers' in the year 2000, in Germany. Did you encounter that?

I think that was one of the papers you forwarded, was it?

Oh, right.

Something I've read. Yes.

Right, well then, then you will know. It was, it was tape drives, because people had big files, and you couldn't put big files on disks. It was hopelessly expensive.

Mm.

So, it was all tape files. So programs were run in, in administrative data processing applications of course, were running through sequential tape files, which have regular structures. So, so it really was actually a good idea. And the insights embodied in JSP, I think were, were genuinely useful, really, really useful. And one of the symptoms of, virtue of the approach, I used to claim, and still would, was, at the end of the course, by the end of the course, or by the latest stages of the course, people who had been listening and paying attention would all produce the same answer to the same, the same problem.

Mm. Yes, that's good. Yes.

Of course, of course they were simple problems. They were the sort you could do on a course. But still.

[53:43]
Mm. How did the Government come to adopt it as, is it A Standard?

Yes. Yes. I don't know really. I think, because the Civil Service College invited me to give a course, I get involved in courses at a place in Norwich, and I went up to there. And, in London, in Victoria, Vauxhall Bridge Road I think it was, there's an office, and we gave courses there. And they wanted to adopt it as their standard programming method. And, and basically, that happened.

[54:25]
So, a bit after this, I think JSD started, at least appear in your mind if not the product then.

Yes.

How did that come about? Was it an evolution of JSP, or, just a realisation of other problems out there?

Well, in a way it was an evolution of JSP. It's a slightly sad tale, this, in some ways, but I'll tell you what it was. It was an evolution of JSP, because in, in JSP courses, and in the book, the *Principles of Program Design* book, a notion was introduced that, first, where you have, where you have reality in which there is an important sequential aspect to events, it's, it's critical to understand that sequential aspect, and to understand it with the right span, by which I mean, when does the sequence begin and when does it end? And, in the JSP book, and in various other places, I made the point that that was a notion which you, you could call long-running programs. And, I used to have a song and dance act on the course, which I, I remember with pleasure actually, which was, it went on the following, on the following lines. There was a bank which wanted to have a system, and they, they called up their JSP programmer to ask about the system. And the investigations that the programmer was, they ought to start not too ambitiously, and not too ambitiously meant, they should just have one customer. And furthermore, they should I think have a machine which was ultra-reliable, and would run without switching off indefinitely. And so, now we have the sequence of what happens to this customer, and what we want to print out about it, like, you know, what's the current balance, and, some analysis of past history. And guess what? This is a JSP programme. It's just pure JSP. And then there was a whole song and dance about, I don't know made it into a sort of pantomime tale, about how they wanted to get another customer, and what were you going to do about that? So the answer to that was, well, you have to have a way of switching the machine's attention from one customer to another, but you have to preserve the state of the computation of that.

Mm.

So, you get databases. So there was that, it was that kind of thing. And JSD was built on those notions. But, it was sorely miscalculated in terms of what was either fully… Well perhaps it was quite convincing, but it was, it was miscalculated in terms of what was practical for people who actually felt, well wait a minute, you know, we've got the database now. Where does that leave us? And JSD was an attempt to introduce the notions that I have very briefly sketched a moment ago, into the development of data processing systems. And, to be honest, it wasn't, it wasn't successful. And there was a book about it, there was a book about it.

[57:56]
But, the, the potential users and customers really weren't convinced, and probably they were quite right not to be convinced. So, it sort of limped along, and that's, that's really what happened. And that was a little bit sad.

[58:16]
Hm. So mid-Eighties, I think you're interest started to shift to, it says here, 'My interest shifted to broader software engineering concerns,' in the mid-Eighties.

Yes. Right. Right.

Is that how the problem frame approach started appearing? Is that what you mean by that?

Yes. Yes. [pause] Yes, yes yes, it is. I started… I mean, I started sort of, proposing simple intuitions to myself, like, software development is just a matter of creating descriptions. So the question is, what are you describing, how do you describe it, how do the descriptions come together to make the system, and so on. And I thought a lot about that. And, having sold the company to our Swedish licensees in 1984 or 5, they insisted that I stayed for five years. I did tell them, truthfully with hindsight, that it wasn't in their interests to have me stay, but they didn't see it that way. So I stayed, and I thought about various things, and I learnt small talk, and, did some of this. And, did other things. And thought about these ideas. But also, I got involved with Pamela Zave at AT&T, well, Bell Telephone Laboratories then, because, she and her colleagues, in fact her manager, a man called, hm, I should remember his name, but I… I do remember it, but I can't put my finger on it at the moment. They learnt about JSP, and they thought, we could use this in telephone switching in, that's to say, you know, subscriber services, features, you know, like, well you know what they are.

Mm.

Yes. And so, she got interested. Invited, invited us to send someone to give a course on JSP at Bell Labs, and John Cameron went, who by then was one of the people within JSP. And, did very well, as he would. And, actually, that was after, I think, I

had been invited to go and do a course as early as, could it be 1976? It can't have been as early as that, can it? I don't know. It was a long time ago. But I went to, where were they? [pause] Bell Telephone… I think it must have been at Indian Hill near Chicago. And I went and gave a course there. So, so I, I got involved with Pamela. And, we talked about the problems of telephone switching, because, I don't know whether this is all familiar to you, but, Bell Labs started programming electronic switching for the domestic telephone network. I mean they were, they were at it in 1968, there's a whole account of ESS then. And by the time I had got involved with Pamela, it was, ESS 5 was it? I think. So it had been through various modifications and, and changes. But, not to put too fine a point on it, it was a gigantic mess. And the reason that it was a gigantic mess was that, the features all interacted in horrendous ways. It was, it was really quite… It was quite delightful how hard it was.

[1:02:10]

A little anecdote. Slightly into the future from there. In part of our cooperation, which went on from essentially, 1989 or 1990 to 2001 or 2002, so it was, twelve, thirteen years. At some point we got hold of a little local, what was it called, PABX system, and, and we were playing with it. Discovered a bug. And the bug was this. If Pamela called a number, I could bridge onto the call, so there was something that you could press. I could bridge onto the call on my, my handset. And, if the call then proceeded, and then eventually the callee hung up, then Pamela would hear a dial tone, and I would hear a dial tone. And, we each thought, good, we've got dial tone, I'll now call my wife and say I'm going to be late for dinner, or whatever. We each phoned, dialled a number. The number that was actually called was the interleaving of the digits that we had both done. And we, we said, this wasn't a bug, it was a feature, and it was called cooperative interactive dialling. [both laugh]

[1:03:34]

So anyway, what was I saying? These features were really, really difficult to get right. And they got the software into a mess. They had terrible organisational problems, I don't mean of their organisation, because they had to distribute new versions, they called them generics, of the software, 5VSS, to all their customers regularly. And they had to be working on, on a new system while they were still supporting the previous system. And, roughly, they had got into the situation which Microsoft called infinite bugs, where, when you get a bug, you fix it, and now you

have two instead. And it was, it was like that. And they made various attempts at, at redesigning the architecture of the software. But, they, they couldn't solve that problem. So Pamela really wanted to talk about it. So, I went roughly speaking six times a year for a week at a time. We sat together in Pamela's office, and we talked about phone systems, and about programming, and finite [inaud] and all sorts of things.

[1:04:52]

And then, one day there was a wonderful revelation, which, well which, which came. And that was this… You know, one of the, one of the subscriber services, it's called, what's it called? It's called, the 1800 dialling or something like that. So you can dial 1800 airways, and you get British Airways. So how does this work? Well, the way it works is that, when you dial 1800 something, what your local switch does is, it forwards the call to an 800 server, and there's a machine in the network which is dedicated to doing 800 numbers. And then that one forwards it to British Airways, and they looked it up. And, this was the key to what we called DFC, Distributed Feature Composition. What you do is, you treat the features as if each one had its own hardware, its own, its own computer. Of course you're doing it in software in fact, and you can replicate the features, but then you can put them together, and you can govern their interactions by, by the way they communicate and so on, so forth. And it really, it really made a big impact on the problem of feature interaction in phone systems. Just in time, just in time for voice over IP to undermine the value of the domestic network. [both laugh]

Yes. Well…

But it was fine. But it was, it was fine. And that was, that was… That was lovely. That was lovely. So…

[1:06:47]

Meanwhile, all this was going on, and, and problem frames were developing not so much with Pamela and I at all, really much with Pamela, our problem frames, but, I kept thinking about it, and in 1995, and *Software Requirements & Specifications* was published, a lot of, sometimes light-hearted pieces about software, motivated by the ideas of problem frames, and, prefaced by a frank admission that, this book is called *A Lexicon*, with a number of different articles about different things, because, I am not

in a position to make a coherent approach out of the whole thing yet. And that's where it was in 1995. And now I'm [inaud] closer to her. Yes.

[1:07:48]
So, to come up to today. So, this is still very much an interest of yours, is it?

Absolutely. It's more than an interest, it's an obsession. [laughs] Yes. Yes.

And is the Open University active?

Yes. What happened at the Open University was, I had some contact with them earlier, in the Seventies, Eighties, or, I'm not sure. But then, Bashar Nuseibeh, do you know of him? Bashar Nuseibeh was a, a doctoral student and then a postdoc, and then a, I think a lecturer or a, I think a reader, at Imperial College in computer science. And we became friendly. I had some, because I had some contacts at Imperial College, and so I got to know him a little bit then. Not very much. And he had an opportunity to go to the Open University, and, as it were, introduce research into their mix. And, we talked about it a lot, and he, he did go, and he invited me to, not exactly come along, but come along a little bit later perhaps, or, yeah, roughly like that, as a visiting professor. And so, the visiting was not terribly onerous, but we spent a bit of time together and we did various things. And I interacted with a lot of people there. And I hope I was of some value, but I don't it was a great deal. But it was very, I mean, nominally it was a day a month, so, there wasn't, there wasn't that much that could be done. And that went on until, I think a couple of years ago when I think someone in the OU admin department looked at my birth certificate and said, 'We don't need this guy any more.' [laughs] And, and so, Bashar and I agreed that I would stop what had nominally been called a consultancy, which made it licit to pay me, a pretty modest amount, but it was all right. And so, I still have the visiting chair, but I don't have the pays that that go with it. So that's, that's what's happened there.
[1:10:12]
But out of the problem frames ideas, and I worked on the problem frames ideas there, with two OU people, Jon Hall and his wife Lucia Rapanotti, have you come across them at all? No. Well they wrote some papers on problem frames, and Jon organised an international conference that had, one or two or three appearances on problem

frames, and more recently they've extended to problem oriented engineering, and, generally problem, well, ideas that can vestigially be traced back to problem frames, for just dealing with problems more, more broadly. And they're, they're hard at it on that. Yes.

[1:11:07]
Mhm. [inaud] introduction I remember saying that the problems that JSP, JSD that actually problem frames are trying to address are still with us really. How do we write reliable, accurate, maintainable computer systems?

Mhm.

Do you think in a way we haven't moved on very much, and that there's still an awful lot of unreliable, inaccurate computer systems out there?

Well, I ought not to talk about what's out there, because I am largely inclined to lock myself up in here. But I do sort of, I think lurk is the correct word, on some, some email lists, and, I think there are aspects of software development which are still neglected by people who ought not to be neglecting them. So I wouldn't dare to say anything, for instance, about what the avionics people do, because the safety record of passenger airlines is absolutely astounding, really, really, truly astounding. So I… And they keep their cards close to their chests, they don't publish papers saying, this is what we do, and you can do the same. So, I don't know on this. But I do know, I mean, for instance, in, in areas like requirements, and Pamela and I were active in the very early Nineties of fastening on so-called requirements engineering and starting conferences on it, writing papers on it, and all that kind of stuff. But, I don't think any understanding has been achieved, in general, by the community of developers in general, of what, what requirements are, or ought to be, or how to deal with them. And I think… I mean to sum it up, but if you looked at a little of my stuff, this won't be news to you, I think, one way of thinking of it is that, over-simplifying ludicrously, there's two kinds of software engineering. There's engineering *of* software, and there's engineering *by* software. Engineering *by* software is about embedded systems and cyber-physical systems, where the software is being used to engineer behaviour in the physical world. And, that means that your, your product is actually not the

software, nor is it the execution of the software. It's the behaviour of results in the physical world. That's what, that's what you're really producing. But in order to produce this, you have to have, you have to have machines and software, and you have to understand that the structure of the problem, well, behaviour, is probably not an appropriate structure for the software that you want to execute, and that you have deployed on the hardware that's available. And that understanding seems to be missing in a lot of places where you might expect it to be present.

Mm.

And so I think that's, I think that's what I'd say about that.

Yes. My opinion, for what it's worth, is that, the advent of the personal computer has obviously been of huge benefit generally, and the Internet, and people do things they could never have done before, and so on and so forth.

Right.

And that's marvellous. And yet the overall quality of software that most people see, through websites, the PC interface and so on, is in engineering terms, pretty poor.

[1:14:59]
Yes. And, and what is distressing is that, the same kinds of mistakes are repeated endlessly, and one of the worst examples is Therac-25. Do you know about Therac-25? Therac-25, well, the Therac-20 was a radio therapy machine which treated cancer patients with, I think it was X-rays often, and that was the Therac-20. And, this is, this is… You know, bombarding people with high intensity and high power rays is, is dangerous, obviously. And there were various mechanical interlocks on this system, the Therac-20. Then the company produced the Therac-25, and on Therac-25 the mechanical interlocks were largely superseded by software interlocks. And, the Therac-25, as a result, killed and seriously injured a number of people. And there's a paper about it by Nancy Leveson and, Clifford? Clifford Turner? Clark Turner? Clifford Turner I think, from '86 or thereabouts. And, to my mind one can see reading the paper, although Nancy Leveson and her co-author didn't seem to see the

same thing, on critical reason why, why these disasters happened. However, wind on fifteen years – 25 years, 25 years, and in 2010 there were, reported in the *New York Times* in two editions, there were terrible, terrible deaths of people with radiotherapy machines, which suggested, and I must admit I didn't look any further than the newspaper accounts, which suggested that nothing had been learnt, nothing at all. And a lot of people died. I, I say a lot, I don't know how many people died, but, certainly more than, more than a very, more than two or three or four.

Mm.

[1:17:16]
So that was a… And then the plugger. Do you know about the plugger? The plugger was a device used in Iraq. And, it's a hand-held device in which the holder, who's in the American military, calls down a missile attack on, well, a specific map location. And what you do is, roughly speaking, you enter the, you turn it on, you enter the location, the target location, and then you press the button that says 'fire'. A bit of simplification, but not much. And, and then, some, I suppose relatively nearby, or maybe not, maybe flying around in the air, system connected to this, then fires the missile at the location specified. So on one occasion, and this is all written up in the papers, somebody picked up his plugger, and he entered the location, the target location, and he was about to press the fire button when he noticed that the red light was blinking that said 'battery low'. So, he replaced the battery, and then pressed the, pressed the fire button. Now, when you replace the battery, and the machine boots up again, because, that's what it was doing of course, it resets the location, guess what, to where you are. The GPS system. And so, the missile came, and killed the people using this plugger. Now that's such an elementary failure of component system design. What did the military have to say about it? Our staff should be… 'Our people should be better trained in the correct use of the systems.'

Mm.

[1:19:20]
So, yes, there's plenty of, plenty of reasons for outrage.

Yes. Well, it's possible to get depressed actually isn't it. But… It is possible to get depressed about some of this.

Yes.

But on the other hand, we have achieved a lot with computing and software over the years and…

Oh, yes, absolutely.

I think it's people like yourself who need to keep the beacon of truth and trying to get us all better.

Mm. Well, I, that would be, that would be nice, that would be nice, yes.

Mm. A little quote here from Tony Hoare, who you mentioned before. He was actually talking about trying to write reliable software, rather than accurately correct software.

Yes.

But the sentiment is I think what we've just been talking about. He said, back in 1995, 'It has turned out that the world does not suffer significantly from the type of problem our research was designed to solve.'

Yes.

And I think that's the point isn't it. We're kind of tolerant of these failures, thinking that generally speaking we're still going in the right direction.

Well, I think there's a deeper point in the background to that. And that, that was a quotation from a paper called 'How did software get to be so reliable without proof?' And it was addressed to the people like Tony and his colleagues who thought that the problem with software was correctness. That's to say, you wrote programs that were

not correct, because they did not satisfy their specification, regarded as programs. And therefore, if you could do program proving and correctness proofs, and prove that the program was correct, you would get rid of those errors. And I think what he is saying there is, I have got to admit, I don't know whether he'd agree with this today, I've got to admit that what we, I'll characterise them as, for this, sort of, mathematicians in their approach to computers, what we thought mattered, which was program correctness, actually, it's not the problem. And I think, I think that's absolutely right. The problem is that, the programs that matter that we're talking about are programs that interact with the world. And, this is engineering by software, and what you are producing as a software engineer is the behaviour in the world, and that's where the main effort has to be. So, you know, what was wrong with the plugger? I'm quite sure it had a specification with respect to which its program was correct, but it had a behaviour that was certainly not correct, which was that you could do this sequence of things and commit suicide.

Mhm.

[1:22:21]
So I think he's acknowledging something there that is, is quite dear to my heart, but… Do you know about the Grand Challenge?

No.

You know the Grand Challenges in various areas of science, in, I don't know, twelve years ago I think they started. And one of them was, that the, the [inaud] in software development wanted to lift their eyes from the problem of finding the greatest common deviser of two numbers, and consider how to make an avionics system. Or… I, I'm being sort of, spitefully, [laughs] [inaud]. And so they started a, they started an enterprise to do it. And, I thought this enterprise was absurd to be honest, because, they thought they could use the terms of, of wall analysis, and proof, to extend simply into the physical world, and it just isn't true. And it just isn't true.

Mm. I read something here which I think sums up [inaud] people what we've just been talking about, which is about the problem frames approach.

Uh-huh.

Where it says here, 'It is helpful to recognise the solution is located in the computer and its software, and the problem is in the world outside.'

Mhm.

That's exactly what we're saying isn't it.

Yes. Yes.

And the incorrect recognition of the world outside…

Yes.

…is always going to result in apparently flawed software.

Yes. Yes. Yes.

Now I guess that's what your, a lot of your last work has been about hasn't it, trying to get us away from making that mistake.

I would like… I would like to think that, yes. It's also about, as we said earlier on, it's also about enjoying myself with these computers. They are very complicated, but they are fun. [laughs]

Well, Michael Jackson, I'd like to thank you ever so much, and, certainly, talking about enjoying ourselves, I've enjoyed talking to you, and I hope our listeners of Archives of IT do as well. Is there anything you would like to say by way of conclusion?

No, thank you, really, just, you know, thank you very much for being such a, a tolerant interviewer, and, I've, I've enjoyed talking to you very much. Thank you.

Good. Thank you.

[End of Interview]