Oral History of Sir Antony Hoare

Interviewed by:
Jonathan P. Bowen

Recorded: September 8, 2006
Cambridge, United Kingdom

CHM Reference number: X3698.2007

Jonathan Bowen: Hello, Tony. Would you like to introduce yourself briefly?

Sir Antony Hoare: I'm Tony Hoare, principal researcher at Microsoft Research Limited in Cambridge. Thank you for coming here to talk to me.

Bowen: Thank you, Tony. I'm looking forward to our talk together. It would be interesting to know, first of all, how you grew up, and what your mother and father did.

Hoare: My father was a colonial civil servant, and my mother was the daughter of a tea planter in Ceylon. She was called out to Ceylon to act as social secretary for my grandfather, and they met in Ceylon, married there, and I was born there.

Bowen: And do you have any memories of Ceylon?

Hoare: Oh, yes, I have quite vivid memories of going to school there. In those days it was still quite a wild place, and we used to go out to the country -- indeed into the forest -- to see animals and elephants and tigers. Had quite exciting adventures there in the school party.

Bowen: And you had brothers and sisters?

Hoare: I have two younger brothers and two younger sisters. My second brother was also born in Ceylon.

Bowen: And you all got on well together? You were a happy family?

Hoare: Oh, yes, in the end anyway.

Bowen: Yes, like all families. Yes.

Hoare: We still have the opportunity to meet quite frequently.

Bowen: What were your earliest thoughts about what you wanted to do when you grew up? Did you have aspirations?

Hoare: I thought I would like to be a writer. I didn't know quite what I was going to be writing, but at school I was a rather studious and uncommunicative child, and so everybody called me "Professor".

Bowen: Quite portentous, yes.

Hoare: And so it turned out.

Bowen: Yes. Yes.

Hoare: Certainly, being a professor has been a great privilege and a pleasure, of course, for a larger part of my life.

Bowen: Who were your idols when you were growing up?

Hoare: At school, I used to go to the school library and read quite a lot, and I found the works of Bernard Shaw very inspiring. He's of course an iconoclast, so he would appeal to an adolescent. And also the works of Bertrand Russell, who wrote on social matters as well as philosophical and mathematical matters, and I was interested in all of those aspects of his work.

Bowen: What was your school life like? This was in Ceylon, was it?

Hoare: I'm talking now, we returned to England in 1945, so the major part of my school life was in England.

Bowen: Yes.

Hoare: I attended the King's School in Canterbury, which is where I read in the library, and that gave me an interest in philosophy, which I pursued in my university career.

Bowen: What was your first exposure to computers?

Hoare: I began thinking about computers as a sort of philosophical possibility during my undergraduate course. I took an interest in mathematical logic, which is the basis of the formal treatment of computer programming. I was sufficiently interested that one of my few job interviews was with the British Steel,

just after I finished my university course in 1956. I was attracted by their use of computers, how to control a steel milling line. A little later, I attended an interview at Leo Computers Limited, in London. They were building their own computers to look after the clerical operations of their restaurant chain. But I didn't follow up on either of those prospects of employment. In the end, I was employed the company which just happened to meet me and be impressed, I suppose.

Bowen: Was Alan Turing an influence?

Hoare: No. On my philosophy course at Oxford I did read his famous work on the incomputability of certain functions, and his invention of the Turing machine. But I never met him, and I didn't particularly think of him as the great man he was.

Bowen: Was that something you did on your own? Or did they actually recommend...

Hoare: I was asked to write an essay on the subject of the Turing hypothesis for my philosophy course. I spent the day in the science library at Oxford looking. I spent the whole day in fact; I was so fascinated I missed my lunch. But as I said, that was as a topic in philosophy, rather than in computing per se.

Bowen: So classics was quite a wide ranging subject.

Hoare: The course that I was taking at Oxford was called "Lit Hum", and it included ancient Latin and ancient Greek, language and literature, classical, ancient history, and philosophy, both ancient and modern. It was modern philosophy that interested me most.

Bowen: I believe you were at Merton College. How did you choose Merton?

Hoare: Oh, Merton College was the college that my father went to. He had studied the same course in the '20's, and so that, in those days at least, was considered a sufficient reason for choosing one rather than another.

Bowen: Was there pressure on you to that sort of thing?

Hoare: No. No, not at all. One felt that maybe it would be slightly easier to get in if you had family connections.

Bowen: Yes. What was college life like in those days?

Hoare: College life was wonderful. I think that the atmosphere of colleges in those days encouraged quite intense and long lasting friendships among people studying in the same courses, and different courses. I've kept up some of them all my life.

Bowen: Are there any particular people you'd like to mention?

Hoare: Perhaps Michael Jackson, who has worked all of his life in the area of computing, along lines very similar to myself. I think I excited his first interest in mathematical logic as an object of study, so maybe I had an influence on him as well.

Bowen: Indeed, do you remember your first meeting?

Hoare: First meeting with Michael Jackson, no. I do remember one meeting when I explained to him how the ancestral relation worked, the transitive closure of a relation in mathematical logic.

Bowen: So were they quite serious meetings you had together?

Hoare: Yes, I was meeting him regularly as a sort of tutor for several weeks on end.

Bowen: It was just an informal arrangement?

Hoare: Completely informal, yes, yes.

Bowen: Was that normal thing to happen at Oxford at that time?

Hoare: I wouldn't suppose it was abnormal.

Bowen: So you moved from classics to computing. What was the first program you ever wrote?

Hoare: I attended a programming course, which was run at Oxford University when I was a graduate student, and it was run by Lesley Fox, who was a very distinguished numerical analyst. He was in fact my Head of department when I first went to Oxford in 1977. But back in 1958 I attended a course in

Mercury Autocode, which was the programming language used on a computer that the University was just purchasing from Ferranti. I wrote a program. I chose my own exercise program. It was a solution of two person game, using a technique which I found in a book on game theory by von Neumann and Morgenstern. It did, I think-- well, I don't know whether it worked or not. It certainly ran to the end, but I forgot to put in any check on whether the answers it produced was correct, and the calculations were too difficult for me to do by hand afterwards.

Bowen: What was programming like in those days?

Hoare: Very different from today. The programs were all prepared on punched cards or paper tape. It might take a day even to get them punched up from the coding sheets, and then submitted to a computer again, probably in a batch, maybe the following day. It would take a long time if there were any faults in the program, to find out where they were.

Bowen: You just submitted a hand-written program and somebody typed it up for you?

Hoare: That's right, yes. Yes. We weren't so good at typing in those days.

Bowen: Yes, very different from today. After Oxford what did you go on to do?

Hoare: I did national service, which was compulsory in those days, in the Royal Navy, studying modern Russian. Military Russian, in fact. I used to know the names of all the parts of a ship in Russian, even if I didn't know what the actual parts of the ship were. Then I went back to university to study a professional qualification in statistics for a year. I continued my graduate career as a visiting student at Moscow State University for another year.

Bowen: What was it like in Russia at the time?

Hoare: Russia in 1960. Very exciting times. That was the time when Khrushchev was making overtures to Eisenhower, and it looked as though they were going to be friends. Then in the middle of the year a spy plane was shot down over Russia, and everything completely froze over again. It was very noticeable that, all my friends knew that relations on the international level had soured, and they were more careful afterwards than before, shall we say, in our meetings and conversations.

Bowen: How easy was it to visit Russia at the time?

Hoare: The visit was organized by the British Counsel as an exchange studentship. Twenty British graduate students and twenty Russian graduate students were involved, so it was quite easy to go. It was a very interesting cultural experience, of course, to be separated from family and friends for such a long period, but we got on well with our Russian contemporaries, and with each other.

Bowen: How big a group was it?

Hoare: There were twenty altogether, and about six of them stayed in Leningrad University, so there were about fourteen at the time at Moscow State University.

Bowen: Did you mix with Russians much?

Hoare: Russian students? Yes, indeed.

Bowen: Was everything quite free in that respect? Or did you feel restricted?

Hoare: I felt quite free, and most of relations with Russian friends were… you wouldn't… No political problems obtruded But they were very suspicious of each other. We learned quite early on that you never introduce one Russian friend to another, because each of them thinks the other one is the informer.

Bowen: So there was some tension you could feel.

Hoare: That, definitely, yes. And we knew that our rooms were bugged, so we would never talk about Russian friends inside our own rooms.

Bowen: So the political situation was all around you. Not spoken, so much as...

Hoare: Yes, but one can forget.

Bowen: Yes, yes. When you got back from Russia, what did you do next?

Hoare: I met my future employers, actually, in Russia. I was being employed as an interpreter at an exhibition in Moscow, where Elliott Brothers, which at that time made small computers -- scientific computers -- were exhibiting and selling their computer in Moscow at the time. Because of my interest in

computers, I spent quite a bit of time at their stand, rather than the other stands in the exhibition. They gave me a lift back home in the van that had brought the computer over, and offered me employment when I came back, with an additional 100 pounds a year on my salary because I knew Russian.

Bowen: So that was a benefit. Did you have a formal interview?

Hoare: No, I never had a formal interview.

Bowen: So it was all very informal. What was the situation at Elliott Brothers at the time? How big was it?

Hoare: Well, it was very small. The computing division at Elliott Brothers -- which was a much larger company, about 14,000 employees -- the computing division was below 1,000. We were designing, and making, and programming small computers, which were selling in the British and European market.

Bowen: At this time I think you first started your interest in Algol, an early programming language, Algol 60. How did that come about?

Hoare: The computer manufacturer was embarking on the design of a new and very much faster computer, and they thought they would celebrate by inventing a new language to program it in. So as a recent employee, with six months experience, I was put to designing the language. Fortunately I happened to see a copy of the “Report on the Algorithmic Language Algol 60”, and I was able to recommend to the company to implement that, rather than inventing a language of their own. That proved a very good commercial decision, as well.

Bowen: Were they receptive to that?

Hoare: Oh, yes, certainly yes. They all came on an Algol course in Brighton with me: the general manager of the division, and the sales manager of the division, and, indeed, the other programmer, who eventually came to work on the same project, and who eventually I married.

Bowen: Yes, I was going to mention Jill. Do have any early memories? What was your first meeting like?

Hoare: <Laughs> That I can't remember. She had experience on writing a complier before, which in those days was quite an unusual experience, so I very much respected her ability, and her contributions to the project.

Bowen: So she taught you your early compilation...

Hoare: She taught me. And particularly she was a much better disciplined programmer than I ever was.

Bowen: Developing the Algol 60 compiler, how many were on the team to do that?

Hoare: There were about three or four most of time. Me, my wife, Jeff Hillmore, and one or two other people for shorter periods.

Bowen: What was it like to develop a compiler in those days?

Hoare: Well, we sort of… None of us really knew whether we could do it or not. It was: you just start at the beginning, you start writing code, practically, and...

Bowen: What sort of specifications did you have to start with?

Hoare: I did read the literature. The Communications of the ACM had an issue which was devoted to Algol 60, and I found that very helpful in setting out the principles of the design. But perhaps the main specification was one which I wrote myself in Algol 60, as a description of the algorithm and compiling procedures, which were then implemented in the machine code of the computer.

Bowen: The syntax was formally defined at the time?

Hoare: The syntax was formally defined. The grammar of the language was written up in a way that was, I think, completely unambiguous, or certainly sufficient to enable anybody who was interested in writing a compiler that would indeed compile this language, and not some other language.

Bowen: And the semantics, was that...

Hoare: The semantics was a little less formally defined. It used ordinary English to describe what the effect of executing a program would be. But it was very well written by Peter Naur, and it was sufficient to enable us to write a compiler without ever consulting the original designers of the language. And it was sufficient for programmers in the language to write programs, which in the end actually ran on our compiler, without ever consulting us or the original designers of the language. It was a really very remarkable achievement, rather beyond what maybe we can achieve these days in the design of languages.

Bowen: What was going on in other areas, in terms of defining and writing compilers? Were you very aware of other activities?

Hoare: No, we didn't correspond with other people writing compilers, even for Algol, in those days. We didn't know each other.

Bowen: There wasn't a community.

Hoare: There was no real scientific community which one could join to talk over problems with other people who encountered the same problems. No, I think we worked pretty well on our own.

Bowen: So you started your professional career in industry, but you spend most of your time in academia. How did that transition happen from Elliott Brothers?

Hoare: Well, my company, like many small computer manufacturers in those days, wasn't really viable in the long term. It was taken over by a longer company, and then the joint conglomerate was taken over by even larger one, so the opportunities and the nature of the work changed rather radically. I sort of felt that there wasn't very much more I could do for them. A chance came to move over into academic life, and I took it.

Bowen: You literally spotted it in an advertisement.

Hoare: I spotted it in an advertisement in the newspaper with a closing date, I think, which was one day ahead. I drafted a letter of application, and said to myself, "Well, if I catch the evening post, I'll post it, otherwise I won't."

Bowen: So it really was a chance <inaudible>

Hoare: Yes, yes. Like so many of these things, it was an opportunity which I was very lucky in being able to take advantage of it.

Bowen: Yes. And the back of all this, you were developing interesting algorithms, like the Quicksort algorithm that everyone knows now.

Hoare: Oh, yes.

Bowen: When did that fit in?

Hoare: Well, that was so. I think Quicksort is the only really interesting algorithm that I ever developed, and I had already developed that when I was studying at Moscow State University. I got a letter from the National Physical Laboratory, which at that time was just starting a project for the automatic translation of Russian into English. They heard of me somehow indirectly, and so they wrote to me offering me a job as Senior Scientific Officer to work on this project I thought I might as well find out what's going on in this area, and so I looked up the Russian literature on the subject. I met several of the people in Moscow who were working on the machine translation. I wrote my first published article in Russian, in a journal called <Russian journal name>, the Machine Translation. That certainly excited my interest in the topic of computers and languages.

Bowen: But you were asked to write a sorting algorithm? How did it actually come about? Why did you come up with...

Hoare: No. In those days, the dictionary, in which you had to look up in order to translate from Russian to English, was stored on a long magnetic tape, and it took several minutes for the tape to be read from the beginning to the end. Well, this dictionary was stored in alphabetical order, starting with A and ending with Z in English, and therefore it paid to sort the words of the sentence into the same alphabetical order before consulting the dictionary so that you could look up all the words in the sentence on a single pass of the magnetic tape. You didn't have to rewind the tape in order to look up the next word, because it was already in the right alphabetic order.

Bowen: But there were existing sorting algorithms. Were they already using those?

Hoare: Oh, I didn't know anything about what existed in those days.

Bowen: So this was literally out of the blue.

Hoare: Yes, I thought with my knowledge of Mercury Autocode, I'll be able to think up how I would conduct this preliminary sort. After a few moments, I thought of the obvious algorithm, which is now called bubble sort, and rejected that, because that was obviously rather slow, and thought of Quicksort as the second thing I thought of. It didn't occur to me that this was anything very difficult. It was all an interesting exercise in programming.

Bowen: Yes. Others found it slightly more difficult. So you invented Quicksort early on. You've obviously worked on Algol, and Niklaus Wirth was involved with Algol, as well. How did you come to meet him, and how did that collaboration work?

Hoare: After completion of the Algol compiler that I wrote for Elliott Brothers, I was invited to join the Algol Committee, a committee of the International Federation of Information Processing, which was devoted at that time to inventing new programming languages. Niklaus Wirth was also invited to join that committee. We used to meet two or three times a year in various nice places in Europe. Niklaus was actually commissioned to write a draft of the next version of Algol, which he did, very well. I made a lot of comments on the draft, and suggested a number of additional features that could be included, and the proposals were read and studied with considerable interest by my colleagues on the committee.

Bowen: And that's something Elliott Brothers supported.

Hoare: Yes, I've always been lucky that my employer has supported my scientific activities, as well as my professional life.

Bowen: I think that's partly because you're good at combining the two.

Hoare: Thank you.

Bowen: After your move to Queen's University in Belfast, which is in 1968, you wrote a very important paper on the axiomatic approach to computer programming, which has since become known as "Hoare Logic".

Hoare: Yes.

Bowen: Would you like to just say something about what led up to that?

Hoare: Yes, I was interested, as indeed many people were at that time, in making good the perceived deficiency of the Algol report, that while the syntax was extremely carefully and formally defined, the semantics was left a little vaguer. We were pursuing the goal of trying to get an equally good formalization of the semantics of the language, a goal which it think still is pretty advanced and maybe beyond our grasp. I put forward the view that we actually didn't want the specification to be too precise. We didn't want the specification of a programming language to concentrate in too much detail on the way in which the programs were executed, but rather we should set limits on the uncertainty of the execution of the programs, to allow different implementations, to implement the language in different ways. In particular, in those days, the word lengths of the computers, and all the arithmetic of all of the computers, was different. So the idea that I put forward, based on the ideas of mathematical logic which I studied at university, was that I put forward a set of axioms which describe the properties of the implementation without describing exactly how it worked. It would be possible, I hoped, to state those properties sufficiently precisely that programmers would be able to write programs using only those properties, and leave the implementers the freedom to implement the language in different ways, but at the same time taking responsibility for the fact that their implementation actually satisfied the properties that the program was relying on. Well, this idea is-- well, I still put it forward. I haven't abandoned it, but it didn't turn out to be very popular, among language designers anyway. The "Axiomatic Basis" [An Axiomatic Basis for Computing Programming, Communications of the ACM, V12, N10, 1969] was my first publication on the subject, very much resulting from my study of the work of Bob Floyd. I think he showed how one could actually use the axiomatic definition of a language to write and verify the correctness of programs written in that language. That's what I followed up.

Bowen: So that was, in a sense, a turning point, certainly I think in computing, perhaps also in your scientific career. In the '70's you moved on, and you worked on a very well known book in computer science on structured programming which was a collaboration with Dijkstra and Dahl, two other very well known computer scientists. How did that collaboration come about?

Hoare: I met Dijkstra and Dahl, I think it was at an IFIP working conference in 1972 on formal language, definition languages. The topic of the conference was pursuing that question about the semantics of languages. Dijkstra was the other person writing an Algol compiler at the same time as I was, so we had that in common, and Dahl was inventing a new simulation language which he called Simula, in which he introduced the ideas of object oriented programming, which would later have a great influence on programming and programming languages. Now all three of us, had written draft monographs on our favorite topics, the one by Dijkstra called "Notes on Structured Programming," and one by Dahl on hierarchal program structures, and my own notes were on data structuring. I thought it would be interesting to collect these three together and publish them as a single book. I spent quite some time trying to understand what Dahl had written. He was a very dense writer. Well, dense in the sense of a very brilliant writer.

Bowen: Yes.

Hoare: I had great fun trying to simplify some of his really brilliant ideas on how to structure programs in the large. I was, at the time, editor of a series for Academic Press, and so that was a natural place where I should get it published.

Bowen: You were effectively an editor for that book. Did you work on Dijkstra's material as well?

Hoare: I did not work on Dijkstra's material, no. That served to be pretty clear and well written itself.

Bowen: Elegant I'm sure, yes. Obviously that had a great influence on structured programming through the '70s. The next year you worked on an axiomatic definition of Pascal. What sort of involvement did you have with Pascal?

Hoare: Pascal was the language that was designed by my friend Niklaus Wirth, after the language we had designed together in the early '60's had not been recommended by our parent committee. He designed it as a teaching language. We used to meet quite frequently to talk about the design, and make suggestion on how to simplify it occasionally. My research on applying the axiomatic method to programming language semantics had made quite considerable progress by then, and I thought it was time to see whether I could tackle a complete language using this style of definition. Well, the easy bits I managed to do. There was still quite a lot of challenges left over, which we just omitted from the definition of the language.

Bowen: In another aspect, you were also looking, at the time, in the context of operating systems, at monitors. I believe you collaborated, or at least discussed these with Hansen and others.

Hoare: Yes. In 1972 I organized a conference in Belfast which was sponsored by ICT, which was then the main -- ICL, I think it already was -- which was the main British computer manufacturer. We assembled quite a brilliant group of scientists to talk about operating system techniques. I was interested in exploring the ideas of concurrency -- parallel programming -- simultaneous operation of several programs incorporating on a single task. I was interested in studying that from an axiomatic point of view. Could I define axioms that would enable people to safely write concurrent programs in the same way as they can write sequential programs today? The topic came up at the workshop, and in fact we devoted an afternoon to discussing the emerging ideas of monitors. I and Per Brinch Hansen were the main people to contribute to that discussion.

Bowen: Is that something that led on to your work on communicating sequential processes?

Hoare: I was indeed, yes. Yes.

Bowen: How did the two link?

Hoare: I think it was the change in technology that actually spurred me to move from monitors to communicating processes. It was perfectly possible to regard a monitor as an isolated program, such as might be executed by a single stand-alone computer doing nothing else. So the idea of a free-standing process, as it came to be known, was already inherent in the monitor concept. But in the monitor concept it was mixed up with a lot of complexity associated with method calls. The idea of a communicating process is that instead of calling its components as subroutine, or a method, you'd actually communicate the values that you wanted to transmit to it by some input or output channel, and it would communicate its results back again by a similar medium. The reason for this, as I say, was a technological advance in the hardware: the advent of the microprocessor. The microprocessor at that time was a cheap and small machine, with not very much store, but in principle capable of communicating with other microprocessors of a similar nature along wires. I thought some simple metal wires, which would communicate signals from the pin one of these machines to the pin of another. It was easy to predict that the fastest and cheapest way of making a large and fast computer would be to assemble a large number of very cheap microprocessors together, and allow them to cooperate with each other on a single task by communicating along wires that connected them. For this, a new method, I should say, discipline of programming, a new paradigm, a new architecture of programs would be appropriate. And perhaps a new language for expressing the programs. That was how the communicating sequential processes came to take a leading role in my subsequent research.

Bowen: This was also the time when you moved from Queen's University, Belfast, to Oxford.

Hoare: That's right. The motivation for my move to Oxford was indeed partially supplied by my idea to study the Oxford ideas on the semantics of programming languages again. I hoped that it would be possible, using the Oxford techniques of defining semantics, to clarify the exact meaning of communicating processes in a way that would be helpful to people writing programs in that idiom.

Bowen: And of course this was a very sad time, when Chris Strachey died, but I believe you knew him before.

Hoare: Yes, I knew Christopher before. I had visited him at Oxford in fact, as a member of a visiting panel commissioned by the Science Research Council to make a report on the work that he was doing at the programming research group. It was a great loss.

Bowen: I think one thing you both share in common is the connection of theory and practice.

Hoare: Yes. When eventually I took over his chair at Oxford, quite literally sitting in his chair and sitting at his desk, I happened to open the drawer, and come across a final report on one of his research projects, in which he put forward his ideals for keeping the theory and practice of programming and computer science very much in step with each other. He said the theory could easily become sterile, and the practice could easily become ad hoc and unreliable, if you weren't able to keep one firmly based on the other, and the other firmly studying problems with the practical uses of computers. He said his goal was to make sure that, in his group, the programming research group, the division between theory and practice could never arise.

Bowen: Well, I think you helped with that afterwards.

Hoare: I hope so, yes.

Bowen: About this time you also started another very well known series, the Prentice Hall International Series on Computer Science, sometimes known as the orange book series. How did that...

Hoare: No, red and white.

Bowen: Red and white, sorry. How did that come about?

Hoare: Well, I suppose that Henry Hirschberg, who was a commissioning editor for the UK branch of Prentice Hall, must have heard of me, perhaps through the academic press connection. Even before I moved to Oxford he had entertained me quite lavishly to a very nice lunch, and invited me to edit a series for Prentice Hall. I thought it seemed a very attractive proposition, and the terms were very good. The prospect of working for the US branch of the company was attractive, because there was a hope that one would be able to sell the books a little more vigorously in America, rather than just in Europe.

Bowen: It's certainly been a very good series for computer science, I think.

Hoare: Thank you.

Bowen: Of course you continued your work on CSP, getting a lot more formal <inaudible>. In particular, you worked with people like Zhou Chao Chen, and Eric Hehner, and Brookes, and Roscoe. Would you like to say anything about any of your collaborations?

Hoare: I'd like to say a lot about all of them, if we have time.

Bowen: Yes, yes.

Hoare: I think the first two were Steve Brookes and Bill Roscoe, who were mathematics graduates of Oxford University. They were recruited by Joe Stoy as doctoral students in the programming research group. I managed to interest them in communicating processes and in constructing a formal model for them. Most of the detailed mathematical work behind that model, and the proof of all the theorems necessary, were done by Steve and Bill, who both, I'm glad to say, have gone on to make brilliant careers in computer science. Bill Roscoe is my successor as head of department in Oxford now.

Bowen: This led up to your CSP book in your own red and white book series in 1985, but you were also working on other things. You were very much promoting a new formal notation that became known as Z at Oxford, although maybe you weren't directly working on it. I understand you were involved in inviting Jean-Raymond Abrial for instance, who very much promoted Z and defined Z. Would you like to say something about how that started?

Hoare: Yes, it'd be a pleasure. I met Jean-Raymond Abrial at a summer school organized by Electricite de France in a lovely little village in France called Labrae sonnalp [sp]. I was very impressed by his research, and indeed his lecturing. He was a brilliant lecturer. I invited him to spend some research time at Oxford, and got a research grant for him from the Research Council. He did marvelous work in the development of Z. This is a sort of specialization of the techniques of logic and set theory, which were originally thought of as the foundation of mathematics in the earlier part of the last century, and now being suggested as a suitable notation and conceptual framework for specifying the desired properties of computer systems. The formal theory was adapted for this purpose and tried out in a number of small and other realistic and large sized examples, and found to be surprisingly effective at describing what a program is intended to do, rather than how the program works, how the program actually does it.

Bowen: I think you were involved with encouraging IBM to actually take up the ideas. How did that come about?

Hoare: Again, it was very much a matter of chance. I was invited to give a keynote address at the British Computer Society Symposium on professionalism, I think it was, in programming. I talked a bit about formalization and verification, and put forward a conjecture, fairly tentatively, that maybe the time was right to scale these things up by trial use in industry. In the audience there happened to be quite a senior director from IBM at Hursley. He came up to me after the lecture, and invited me to put my commitment where my mouth was, and actually do something in collaboration with IBM. That made me gulp a bit, because I knew that -- well, I had the impression -- that IBM had produced some pretty complicated software, and this really would be a challenge, and there was a good chance that we would fall flat on our faces. But you can't turn down an opportunity like that. So Abrial and Ib Sorensen and other colleagues set to work, and actually produced some very useful analyses for them, using Z, to help them with an

ongoing project for restructuring and rewriting parts of their software system, a software system called CICS, C-I-C-S. It stands for Customer Information Control System. It was one of the most profitable and influential software products in those days, so it was an important project.

Bowen: That led to a Queen's award for technology. Another strand of your work led to another Queen's award, which was the work on Occam and the transputer, which came out of your CSP work. How did that connect?

Hoare: Well, the founder of the INMOS, the British company which developed the transputer and developed… Well, the transputer was as embodiment of the ideas that I described earlier, of building microprocessors which could communicate with each other along wires which would stretch between their terminals. In fact, this was pretty unusual groundbreaking architecture in those days. Most attempts to connect these microprocessors together were very high cost. They cost far more than… well, ten times as much as the actual processors themselves. INMOS managed to get this down to negligible costs. Well, he had the vision that the CSP ideas were ripe for industrial exploitation, and he made that the basis of the language for programming transputers, which was called Occam. We continued to develop contacts between the company INMOS and Oxford University. When they came to develop the second version of their transputer, one that had a floating point unit attached to it -- well, not attached to it, rather actually built into it; that was another first, a processor which actually included a floating point unit on the same chip. The question arose as to whether the design of the floating point unit was correct. My colleagues at Oxford, Bill Roscoe and Jeff Barrett, actually used formal models of communicating processes, and techniques of program verification, to check that their designs for the implementation of the IEEE floating point specification were in fact correct. It turned out, maybe with one possible exception that the engineers claimed they knew about anyway, the formal verification possibly discovered one flaw in their design, and it was removed. The company went ahead to produce their masks and their computer without conducting any further tests. This actually saved them a lot of money. They estimated it enabled them to deliver the hardware one year earlier than would otherwise have happened. They applied for and won a Queen's award for technological achievement, in conjunction with Oxford University Computing Laboratory, for that achievement. It really is, it still stands out as one of the first applications of formal methods to hardware design, long before Intel became very interested in the same problems.

Bowen: Of course, you were also very interested in the use of algebra, for CSP algebraic laws, also leading up to work on laws of programming.

Hoare: Yes, the algebraic approach. Algebra's a particular format for writing axioms. If one is able to express the properties one wants of a system, or a programming language, in the form of algebraic equations, this is a very simple and very elegant, and intuitively pleasing way of describing the properties of the operators that you're interested in. People learn algebra, of course, by applying it to numbers and

operators like plus and minus and times and divide. But we have learned to apply algebra also to the operators of a programming language, sequential composition, conditionals, and to considerable extent even concurrent composition, allowing two programs to run in parallel.

Bowen: In the early '90's and late '80's, you started thinking about provably correct systems, and amongst that work, you worked on duration calculus with Zhou Chao Chen. How did that come about?

Hoare: I spent 1987 as a visitor on sabbatical at the University of Texas at Austin. I spent quite a time with a company founded by Bob Boyer and J [Strother] Moore, called Computational Logic Inc., or CLINC for short. They were making a living by selling prospects for verification of software and hardware. The hardware manufacturing companies, and the security agencies, were quite interested in the possibility of verification as a means of boosting security and confidence in computer systems. They had constructed a verification of a small processor, a microprocessor, an assembler, a low level machine language for the processor, a higher level language which was compiled to machine code, together with a proof of the correctness of the compiler, and also a verification condition generator, which enabled and helped programmers to prove the correctness of programs actually written for this machine. This project was called The Stack, because all the components of it were built one on top of the other: The assembler on top of the hardware, the compiler on top of the assembler, and so on. I thought that we ought to be doing something similar in Europe to try to catch up with our American colleagues. At that time the European Commission was just beginning to support basic research under its Esprit Programme. So I put forward this idea, as one of the things that we might contribute, this time perhaps applying the stack to an existing processor, that's the transputer, and an existing language, that is the Occam language, rather than one being invented for the purpose. Well, of course the important thing is that European connection enabled us to enlist the help of a number of brilliant European colleagues who went on to do all the actual work. One of them was Zhou Chao Chen, a Chinese visiting scholar, whom I knew well because he'd visited Oxford in the early 1980's. He picked up ideas and developed them into the Duration Calculus, about which he's recently published a book.

Bowen: You also worked with He Jifeng, another Chinese colleague, for many years. How did he get involved with Oxford in the Programming Research group?

Hoare: Well, He Jifeng sort of turned up. He was sent, by people who knew him presumably at the East China Normal University, to Oxford with a full support for his visit, for a short period. I put him in an office with a number of colleagues -- Jeff Sanders, Carroll Morgan -- and they started talking. He was an amazingly quick person. He doesn't speak English very well. His accent is still very Chinese. So it's very surprising that he understands it so well. We were constantly finding that we'd start explaining some rather difficult technical idea, and when we reached a point, he'd say, "Yes." And we thought he couldn't possibly have understood it. So we would explain it again, and he'd say, "Yes." Then we realized he was understanding everything the whole time. That's the way he was; very quick on the uptake. It's been

a very great privilege to work with him for a decade. In fact we're still working together, at a distance. He's in Shanghai, and I'm here, and we're hoping to publish an article sometime maybe in the near future.

Bowen: Of course, you published a book together on unified theories of programming, which tries to take an algebraic approach and other different semantic approaches. Would you like to say something about this?

Hoare: Yes. Unifying theories is a sort of hobby horse of mine. Other material scientific disciplines take the unification of theories as a challenge which is worth pursuing in its own right. In computing, perhaps hitherto at least, we haven't been sufficiently mature to do that. As a result, we have rather a large number of theories of programming, all of which seem to be rivals to each other, and indeed perhaps are sometimes promoted as rivals to each other. But if you look at it more deeply, I felt that one would be able to see that they really all are variants on the same theme, and that the underlying mathematics, if you looked at it in the right way, would in fact turn out to be the same. The advantage of doing this, well, partly because it's sort of a scientific challenge in its right, as I said before, and partly because I think the diversity of theories is a genuine inhibition to people who might otherwise be very happy to apply one of the theories involved, if only the experts could agree which one it was going to be. Well, of course they never can, and indeed each theory does have distinctive merits. But the unification of theories, I believe, is an activity which enables distinct theories to retain their specific merits, and at the same time, for you to use them in combination in a safe and secure way. That's the goal of unification. Jifeng and I worked for nearly ten years to try to give just an example of the idea of unification as applied to the theory of programming.

Bowen: Of course, eventually you had to "retire", and if I could put that in quotes, I would. Now you're at Microsoft in Cambridge. So you started your career in industry, and you've finished your career in industry, and you had a wonderful symposium in Oxford to celebrate that with six ACM award winners, cheering award winners I believe, with yourself one of them. So what are your goals at Microsoft?

Hoare: Well, Microsoft have given me, as indeed they give all their researchers, a very free hand to set our own goals. At the moment I'm working on two topics. One of them is concurrency, and the other one is in pursuit of my lifetime's goal of verification of programs. The topic of concurrency, I've set myself the challenge of understanding and formalizing methods for programming concurrent programs, where the programs actually share the store of the same computer, rather than being executed on distinct computers as they are in the computing sequential process architecture. Again, the motivation for studying this different form of concurrency is the advance of hardware technology. It now appears that the only way in which processors can get faster is for them to include more processing units on the same chip, and sharing the same store.

<crew talk>

Bowen: So, Tony, you're now officially retired and have moved from Oxford to Microsoft in Cambridge, but you're still very active in computer science. What are you working on now?

Hoare: I'm working on two things. One of them is a return to the theme of concurrency, which is taking a new form these days. The ideas are basically inspired again by developments in the hardware technology. Intel has announced that in the future, the only possible improvements in the speed of their processors is going to arise from multiplicity of processing units. In other words, they're going to put many computers onto a single chip, all sharing the same logic, and the same memory. So now in order to exploit the extra speed of having many processors rather than just one, it will be necessary, again, to change our programming style, and to change the design of our programs and algorithms to exploit the new architecture. I'd always felt that parallel programs that actually shared main memory, and could interleave their actions at a very fine level of granularity, just a single memory access, they were far too difficult for me. I could see no real prospect of working out a theory that would help people to write correct programs to exploit this capability. Of course, I thought it would be interesting to try again, and see whether with the experience that has been built up -- experience of formalization over the last 20 or 30 years -- could be applied effectively to this extremely complicated form of programming. I remembered that Cliff Jones, who'd been my student at Oxford many years ago, had tackled the same problem, and he'd come up with an idea which he calls the rely-guarantee method, which I think might turn out to be the key to writing correct multiprocessor programs and knowing that they are correct. So I've been pursuing those ideas with some colleagues actually from Cambridge University; some graduate students and researchers have been helping to explore those ideas. At the same time, John Reynolds, who's at Carnegie-Mellon University, an old friend of mine, he's worked out a new logic in which the concepts of concurrency are taken away from just computer programs, and actually imported into the very fabric of the logic itself. It's called "Separation Logic", and it completely surprised me as a way of establishing program correctness, that one should use a new sort of logic to do so. I've always assumed that we would be using the same logic as mathematicians and logicians for the last 100 years.

Bowen: The other thing that you observed at Microsoft, though, the use of assertions, which leads back to some of your work in the 1960's.

Hoare: Oh, yes, yes. I was very interested to see whether the ideas of assertions, which had been put forward by Bob Floyd… And in fact before him; already in 1947 the idea of an assertion was described by Alan Turing in a lecture he gave to the London Mathematical Society. This idea of assertions lies at the very basis of proving programs correct, and at the very basis of my ideas for defining the semantics of programming languages. I already knew when I entered academic life way back in 1968, that these ideas would be unlikely to find commercial exploitation, really, throughout my academic career, 30 years or so. I could look forward to 30 years of research interrupted by anybody who actually wanted to apply its

results, So I thought that at the end of that period, when I retired, it'd be very interesting to see whether the positive side of my prediction would also come true, that these ideas would begin to be applied. And indeed, even since I've joined Microsoft in 1999, I've seen quite a bit of expansion in their use of assertions and other techniques for improving confidence in the reliability of programs. There are program analysis tools now, which certainly stop far short of actually proving correctness of programs, but they're certainly very good at detecting certain kinds of errors, some of them quite dangerous errors which make the software vulnerable to intrusions, to virus attacks, have been detected and removed, as a result of the use of formal techniques of program analysis.

Bowen: Looking forward somewhat, you'd been involved with one of the grand challenges in computer science at the moment: looking at verifying compilers. Where do you see that going?

Hoare: Well, yes, that's the other main thing that has been keeping me busy in the last years. The idea of verifying computer programs is still an idea for the future, clearly, although there are beginnings of using scientific technology to make the programs more reliable. The full scale verification, the full functional verification of a computer program against formally specified requirements, is still something we have to look forward to in the future. But the progress that we've made, and the progress that has been made by researchers in the various technologies needed -- the use of computers to do logic and to do proofs -- progress has really been quite spectacular in the last ten years. So I've been dreaming again that a concerted scientific endeavor by my academic ex-colleagues might actually make a very significant advance towards improving programming practice, and making programs themselves more reliable and more trustworthy. So I've been trying to persuade people that this is an interesting challenge, and that they should take it up and work together towards its solution.

Bowen: Looking back over your career, what were the most important lessons you think you've learned during that period?

Hoare: Oh, I've learned many, many lessons, about people, I suppose. Really, it's all about people. How to enable them best to achieve their own objectives, or align their objectives with those of their colleagues. Every case has to be treated on its own, in its own circumstances. I don't think there's any… There's no single big lesson that I could summarize in a few words.

Bowen: What was your proudest moment during your career?

Hoare: I think it was my inauguration as a fellow of the Royal Society, the highest accolade which can be given by the UK scientific community to one of their members. It's a very nice ceremony. You sign the same members' book that was originally signed by Newton and others. They organize a very nice reception, and I was able to invite my father and my mother to be there with me. I'm sure they never

thought of me as a scientist at any time, in fact. It's particularly ironic that when I followed up the offer of employment at the National Physical Laboratory to work on their project for translation of Russian, I was told that their offer of a job as a Senior Scientific Officer had been a mistake. They couldn't offer me such a high position. In fact they couldn't offer me any position in the scientific civil service at all, because I wasn't a scientist; I had no scientific degree. So they offered me only a temporary civil servant position, which I was not sorry to decline. So this was a way of paying them back for not recognizing my scientific potential. I don't blame them, of course.

Bowen: I think you were one of the first computer scientists to be in the Royal Society.

Hoare: Well, I have distinguished predecessors. Alan Turing of course was a fellow of the Society.

Bowen: Yes.

Hoare: Maurice Wilkes, who was the professor at Cambridge and built one of the first stored-program for digital computers, and Tom Killburn, were all fellows of the Royal Society. But I was the first, perhaps, of the second wave of members. Now I think we have some very, very distinguished colleagues in the society.

Bowen: Looking back, what do you see as the impact of your contributions?

Hoare: Well, I think this interview's shown that I've really sort of rather flitted around from one place to another. If there exists such a subject as the theory of programming, I should like to feel like I've been one of its founders. The theory of programming as of course the theory that is applicable in practice, but one which has a mathematical and logical, and intellectual interest in its own right. That's what I would like to be remembered for.

Bowen: Who were your main role models?

Hoare: When I first moved into academic life I was very much inspired by Edsger Dijkstra, both personally and professionally. It seemed to me that he had very high academic standards and very high academic ideals. He was very different from me in many respects. He had no desire to understand notions of management, or of budgeting, or finance, or any of those things. He was going to devote his life, as indeed he did, to the advancement of science, and that was his sole criteria. I, of course, had been a manager, and as Head of department, I continued to be a manager for a very long time, in fact more or less until I retired. Now I don't have any management responsibility, and it's lovely.

Bowen: Yes. You have been involved with education as well, and I think your knighthood was services to education, which we haven't touched on, and you've introduced courses at Oxford. There was no computer science at Oxford when you first started. Is there anything you'd like to say about your teaching.

Hoare: Well, I right from the beginning took the view that computer science, and in particular computer programming, was a very good kind of education in training the mind in habits of rigorous thinking. A deep concentration is really needed in order to write a program that actually works, and that perhaps it might play the same sort of role in education that the study of the classical languages, Latin and Greek used to play in previous centuries, and unfortunately no longer do. So I'd always hoped that programming and computer science would be regarded as subjects fit for education; interesting and challenging even for people who weren't going to eventually apply the knowledge professionally. Unfortunately, or fortunately perhaps, the demand for professional programmers has been so high ever since I joined the university, that the opportunity to take the subject purely as a broad education is hardly significant, even to this day.

Bowen: You also encouraged postgraduate students, who have been at Marktoberdorf lectures. Do you remember how those started, and what's your involvement been with those over the years?

Hoare: Marktoberdorf Summer School is an institution which I really, really value my association with. I've been a director of the school for nearly 20 years perhaps, and I've attended and lectured at nearly every school. They happen every two years, and they invite graduate students -- well, mainly graduate students-- to come and hear leading lecturers from around the world talk about their favorite subjects in a sort of tutorial way. The lecturers have really have been extraordinarily giving of themselves, first of all spending two weeks living and eating with all of the students at a rather remote village in Bavaria, and in presenting their material in a way that is attractive and informative to young computer scientists. We've published proceedings of the school every two years, and I think they form a sort of memorial of some very interesting people, and very interesting computer science.

Bowen: We've talked a lot about the past, but have you got any thoughts for the future of computer science?

Hoare: The future of computer science. Well, I expect the future to be as wonderful as the past has been. There's still an enormous amount of interesting work to do. The practical work on the applications of computers just keeps leaping ahead. Things that we thought were impossible only a few years ago are now commonplace on the market; low cost consumer items. When I first joined Oxford University in 1977, I knew that the majority of people in this country and in the world had never seen a computer, and that if you asked them, they would say they never expected to see one. That was 1977. Even after the transputer had been shown to be possible, I predicted that it would never be possible to digital zoom,

never possible to take a two dimensional picture, and enlarge it or contract it the way that a zoom lens does. Quite wrong again. That's now just taken for granted on every digital camera on the market, and I see no reason why that should stop. As far as the fundamental science is concerned, we still certainly do not know how to prove programs correct. We need a lot of steady progress in this area, which one can foresee, and a lot of breakthroughs where people suddenly find there's a simple way to do something which everybody hereto has thought to be far too difficult. I certainly wish my successors and anybody studying or working in this area the best of success. I'm sure they will have an interesting professional experience during their working lives, and, well, what more can I say?

Bowen: Well, thank you very much for a fascinating talk through all your achievements, and many congratulations on your Fellowship with the Computer History Museum. Thank you, Tony.

Hoare: Thank you very much.

END OF INTERVIEW

