Stories from a Life in Interesting Times

Antony Hoare

Today in this fine hall, it is a great privilege to reflect on the ways in which my education and experience have shaped me and prepared me for a career in advanced technology. In this commemorative lecture, I want to pay tribute to those who have most influenced my philosophy of life, the intellectual leaders of the present day as well as the more distant past. Many of them have already stood on this proud platform as recipients of the Kyoto Prize in earlier years. My thanks are joined to theirs in gratitude for the honour of the award.

I start my story in 1947, when I entered the King's School at Canterbury for my secondary education. For the next five years I lived and learned in the beautiful precincts of Canterbury cathedral, in sight of its tall and elegant tower, and in earshot of its ringing bells.

In my first year of study, I followed ten subjects, including English language and literature, history, French language, the classical languages Latin and Greek, and a double dose of mathematics, elementary and advanced. This left no time to study any science subject at all, except as a hobby—one lesson per week. I very much enjoyed the mathematics, but in our second year we had to choose just two subjects for continued study. In those days, it was expected that any boy, who was good at Latin and Greek, diluted only with a little scripture or French as a subsidiary subject.

Every week our homework would include a translation of a passage from a classical author into English, and more difficult, a composition in Latin or Greek of a paragraph of English prose. Of course, I had to learn the grammar of those languages. It was rather more complicated than English, because each occurrence of every noun and every adjective was classified as one of three genders and one of five or six cases in either singular or plural. There were four or five regular patterns for calculating the form of the word, but there were also many irregular forms that had to be learnt individually. In every sentence in Latin or Greek, every adjective has to agree with the noun that it qualifies in number, gender and case. I always checked my work carefully

according to the rules, but still I kept making mistakes—howlers, they were called—perhaps because my teachers found the errors so easy to detect. In later life when I came to study the design and implementation of artificial languages for communication and control of computers, I was very keen that their grammatical rules should be as simple and regular as possible; and I ensured that the computer itself, like my teachers, could rapidly detect and reliably report on any violation of the rules in its input program.

Translations and compositions were for me an enjoyable exercise. The challenge is to understand the purpose, the content, the progression of thought and the style of a text in one language, and then to reconstruct the same message in a different language, whose sentences have different structures and whose words have different connotations. I learnt the simple rules of rhetoric: of balance and of contrast, of progression and of variation, of thesis and of antithesis. This last sentence itself is a balanced and contrasted illustration of the very rules that it describes; it is progression of three contrasts. I had to rewrite it carefully many times. As always the best results are obtained from trying many alternative formulations before choosing the best. And that is exactly what I have enjoyed doing in all my later scientific writing and teaching. The challenge is to explain the ideas as clearly and as memorably as possible, and to present my arguments and counterarguments fairly, but still in the strongest possible light. I have tried to pass on the same skills to my students, for writing up the results of their own research.

In spite of my official studies of the classics, I fortunately did not wholly abandon my interests in mathematics, taking a special interest in probabilities as a means of explaining uncertain knowledge; also as a means of winning more often in card games.

In the school library, I discovered a book called *Mathematics for the Million* by Lancelot Hogben, which I found very interesting. I am glad that I pursued mathematics as a hobby, as something that one actually does to solve interesting problems, rather than as a fixed body of knowledge that one just learns to reproduce in examinations. If I had chosen mathematics instead of classics as my main examined subject, I might well have fallen out of love with it; certainly I would not have learnt any of the branches of mathematics that I now consider interesting and relevant for programming theory and computer science.

Also in the school library, I came across a thick volume of *History of Western*

Philosophy, by Bertrand Russell, from which I read large sections, starting with early Greek philosophy and proceeding to nearly modern times. That awakened in me an interest in philosophy, which remains to the present day. Among other philosophical works of Russell, I took an especial interest in his *Introduction to Mathematical Philosophy*, a less formal version of his joint work with Whitehead reported in the three volumes of *Principia Mathematica*. Its achievement was to reduce all of mathematical reasoning to simple logical forms, and all mathematical concepts to the single, uniform and apparently simple concept of a set.

Many years later as a professor at Oxford University, I wanted to extend mathematical reasoning to prove the correctness of computer programs, and to do so without commitment to any particular branch of established mathematics. In a collaboration lead by Jean-Raymond Abrial, we showed that set theory, which Russell proved adequate for so many mathematical purposes, was also well adapted to the specification of programs, and for proofs of the correctness of their design and implementation. First steps towards industrialization of this discovery were taken in collaboration with the computer company IBM.

In 1952 I entered Merton College to study at Oxford University. The reason for the choice was family tradition. My father also studied there, the same subjects as me. In my first two years, I continued with Latin and Greek, adding the composition of verse to my accomplishments. My reading also got faster, because I had to take in almost the whole of the works of Homer and Virgil, and selected works of Cicero, Juvenal, Horace and Euripides.

In my spare time, I continued to pursue my philosophical interests, particularly in logic and the foundations of mathematics. In my first term, I joined with a few classical and mathematical friends in a small study group, which used to meet for coffee in the late evenings after completing our official assignments and just before going to bed. We all bought copies of the textbook *Mathematical Logic* by Willard Van Orman Quine [photo 1], Kyoto Prize laureate for Philosophy in 1996. I still have my copy. From this book we learnt that a mathematical proof is nothing but a text, a sequence of lines rather like a record of the moves in a game of chess. The correctness of a proof can be checked, just like the correctness of a game of chess or the grammar of a Latin sentence, by examining just one line or move or word at a time. That is best done without understanding anything about the subject matter of the proof, the strategy

of the game, or the meaning of the sentence. In the case of a proof, each line must either be a copy of an axiom, or a special case of a previous line, or it must follow from two previous lines by a rule known by its Latin name as *modus ponens*. This formal view of the structure of a logical proof is now the basis of all attempts to get computers to assist in the application of mathematics, or the construction or checking of mathematical proofs. Of course, it is not possible to prove the correctness of the axioms, the lines that do not have to be justified at all. They just have to be accepted as 'self-evident'; or less pretentiously, as the rules of the game that the mathematician has chosen to play; effectively, they define the primitive concepts of the branch of mathematics which is developed from them.

Twenty-five years later, I returned to this view of proofs in the first published article of my academic life on "An Axiomatic Basis for Computer Programming." I was inspired by the ideal of my friend, the great computer scientist Edsger W. Dijkstra, that the design of a computer program should be accompanied, even driven, by the obligation to demonstrate its correctness. Following Robert Floyd's suggestion, I extended this obligation to the designer of a computer programming language, which should therefore assist the programmer in the construction of the necessary proofs. So a good way to describe the meaning of a good programming language is just to describe the axioms on which all proofs are based. The simplicity of the axioms, and their ease of application, are an objective criterion for the quality of the programming language design. This insight was later to help Niklaus Wirth in the design of the well-known teaching language PASCAL.

My course of studies at Oxford was called Literae Humaniories, and it followed an ancient tradition. After two years, we started two new subjects, ancient history and philosophy, which we studied from the original authors: Thucydides and Tacitus, Plato and Aristotle. We also studied more modern European philosophers, Descartes, Hume, and Kant. It was on my philosophy course that I was first introduced to the workings of a computer, in the shape of the famous Turing Machine, invented by Alan Turing in 1937. Among contemporary philosophers, we encountered Karl Popper [photo 2] (Laureate of the Kyoto Prize in 1992) as the author of the doctrine that the meaning of a scientific hypothesis lies in the experiments carefully designed to falsify it. His teaching inspired the construction of my later theories of distributed computing networks; I defined the meaning of program quite directly as the set of observations that

reveal all the ways in which the program might go wrong.

At that time, young men in Britain underwent a two-year period of compulsory national military service. It was abolished shortly after. Because of my qualification in Latin and Greek, I was easily accepted to learn the Russian language in the Royal Navy (I was told that family connections also helped: my uncle was a naval captain). We went through the standard military drills, we learnt to fire guns on the firing range, and we went for a few day trips on a destroyer and a minesweeper, just enough to make us feel very sick; but we never spent even a single night at sea, and certainly never came anywhere near to military action.

The main instruction in Russian was given on dry land, in an offshoot of the London University School of Slavonic Studies. We had a very thorough grounding in the formal rules of Russian grammar, which was at least as complicated as Latin or Greek, with nouns and adjectives declining through six cases, though fortunately there were only two genders. We were warned that grammatical mistakes were the quickest way to earn expulsion from the school, loss of our officer rank, and return back to the hardship and drills of conventional military life. So I repeated the care that I had exercised in Latin and Greek compositions to check every gender and every case against all the rules I had learned; but I still suffered from frequent oversights, as I had in my classical Latin and Greek translations.

The main difference from classical languages was that we learnt Russian as a spoken language, and in speaking there is no time to calculate and apply the grammatical rules at all. It was therefore quite a surprise to me to find myself after a while speaking correct Russian most of the time without even thinking of the grammar, in just the same way that young children learn to speak the language. Even the incredibly complex rules governing Russian numerals soon came quite naturally to me. I did not know how to explain the phenomenon, and I still don't.

On completion of my Russian course, I definitely decided that I had been a student long enough, and I was going to work in industry. But I did not wish to move into accountancy, or into an administrative or managerial career, which were the normal prospects for a graduate in the humanities. I wanted a scientific or a technical job; and to obtain and keep such a job, I felt I needed a relevant technical qualification. I found that I could obtain one by a single year's graduate study back in Oxford, on a course leading to a certificate in statistics. I had some difficulty in persuading the statisticians

at Oxford that I had sufficient competence in mathematics to undertake their courses. Later, when I came to accept students on graduate courses in computing, I always showed special sympathy for those applicants coming from background in the humanities, especially languages.

At Oxford, there is a tradition that students of any subject could attend lectures in any other subject, and I was soon attending a course on the philosophy of mathematics given by Hao Wang, who had just written a program on an early IBM 704 computer to check all the proofs in the first nine chapters of Russell and Whitehead's *Principia Mathematica*. It was a remarkable programming achievement in 1957, fulfilling in practice the speculations of Turing, the objectives of the early logicians, and the dreams of the seventeenth century German Philosopher Leibniz. It was at these lectures in Oxford that I first met a Japanese student, Hide Ishiguro, who has since forged a distinguished career as a philosopher in the United States and in Japan.

I also attended my first and only computer programming course. The instructor was the numerical analyst Leslie Fox, who was to be my head of department twenty years later, when I came back to Oxford as a professor in the Faculty of Mathematics. He taught us the only available, rather primitive, high-level programming language called Mercury Autocode. As my first program I coded an approximate method of solving the probabilistic value of a two-person zero-sum game like the children's game paper-scissors-stone. This was because I was reading a book on the theory of games by von Neuman and Morgenstern. I was very excited as I saw my program read in; it was translated and run for a few seconds, and produced as output the six numbers that I had planned. But I never discovered whether the numbers were correct, because I omitted to insert in the program a check that would have easily told me. Such checks are nowadays known as assertions. I later came to recommend them as the basis on which one could prove that the program is correct.

During the last term of my statistics course at Oxford, I saw on my college notice board an advertisement issued by the British Council, the official body which fosters cultural and educational links between Britain and other countries of the world. They had just negotiated an agreement with the Soviet Union for an exchange of twenty university students per year, at graduate level; and they invited applications against a time limit—which expired the very next day. On the spur of the moment I decided to take the chance. If I could organize an application in twenty-four hours, I would submit

it; and as chance would have it, I succeeded.

The main attraction was the adventure. In 1960, the Soviet Union was still largely inaccessible to foreigners, and I would be among the first group of foreign students to go there, with the opportunity to experience and examine a dramatically opposed political system. The Russian culture had evolved quite separately and in isolation from the Greek and Latin influences which reentered Europe at the time of the Renaissance. I also wanted the chance to exercise and increase my skills in the Russian language. And finally, I wanted to study more of the theoretical foundations of probability theory, in which I knew that the great Russian mathematician Andrei Nicolaevich Kolmogorov had been a pioneer. It was in his department in Moscow State University that I was registered to study.

At first it was hard. It requires a lot of concentration to listen to a lecture in a foreign language for a whole hour, and lectures in Moscow were two hours long. Even the benches we had to sit on were hard. Gradually, I acclimatized to the lectures in Russian. I spent a whole day trying to buy a seat-cushion in Moscow, but failed; so I got my parents to send me one by post. Having solved both these problems, I got down to the real problem—the mathematics itself was the hardest of all, especially for a newcomer to start studying abstract measure theory at graduate level. Fortunately, I found in the university library an excellent book by Halmos. It introduced a range of techniques from set theory, lattice theory and analysis of approximations and limits that I was to encounter again later in my study of domain theory, as applied to the study of computer programming languages.

While I was in Moscow, I received a letter which set my life in the direction it has followed ever since. It was from the National Physical Laboratory, the primary British government research laboratory, located in Teddington near London. I was invited to join the laboratory as a senior scientific officer to help on a new project to program a computer for automatic translation from the Russian language into English.

That gave me sufficient excuse to relax my mathematical studies, and start to learn a bit about the current Russian efforts in machine translation, from English into Russian, of course. I read the early issues of the Russian journal *Mashinnii Pereod (Machine Translation)*, and I met and talked to several of the authors. In fact, that was where I published the very first scientific article I ever wrote. It was in Russian, typed

on a Russian typewriter borrowed from a friend. But I never saw any Soviet computer—they were at that time too secret to show to foreigners.

My study of machine translation took me to the Lenin Library in Moscow, the only library that stocked the relevant work on syntax analysis by Noam Chomsky, Kyoto Laureate for Cognitive Science in the year 1988. His theories of the syntax of language are a precise expression of the rules to be used by a computer in checking the grammar of sentences. His important idea was that of recursive definition, that is, a definition that contains a copy of the actual word being defined. For example, in a grammar for a sentence in English, a ‘subordinate clause’ is defined as containing a verb and possibly some nouns and even some other subordinate clauses. Although this violates normal Aristotelian rules for definitions, when used with care it conveys a very precise meaning. In the programming world, Chomsky’s ideas were first taken up by John Backus and used by Peter Naur in the design of the international algorithmic language ALGOL 60. Since then, they have had an enormous impact on the design of other programming languages, on computational linguistics, and on computing science in general.

One of the first tasks of machine translation is to find the dictionary entry for each word of a sentence in the source language. The dictionary was stored in alphabetical order on a long magnetic tape, which could take several minutes for the computer to read or even just to scan from end to end. It would be very inefficient to do the whole scan separately for each word in the sentence. A more efficient idea is first to sort the words of the sentence also into alphabetical order, so that all of them can be looked up on a single scan of the tape, picking up each dictionary entry as it passes under the head of the tape reader. It was to solve this problem on a machine with a very small main memory that I discovered the sorting algorithm Quicksort.

Towards the end of my visit to Moscow, I received another unexpected letter, this time from my uncle, who had retired from the Royal Navy and was then the general manager of the Scientific Instrument Manufacturers’ Association of Great Britain. He was organizing an exhibition in the center of Moscow at which the manufacturers could display and sell their products. I was invited to act as an interpreter for the exhibitors and lecturers from England, enabling them to communicate with the Russian public and with the scientists and potential purchasers visiting the exhibition.

One of the exhibitors was Elliott Brothers Ltd., whose computing division at

Borehamwood manufactured scientific mini-computers. At Moscow they demonstrated their latest computer, called the Elliott 803, and I spent most of my time on that exhibition stand. At the end of the exhibition, the managing director of the computing division offered me a free lift back to England in the empty van which had brought the computer to Moscow: my knowledge of the Russian language, the people, and the bureaucracy would be of great service to the driver on the trip. I was also invited to consider an offer of permanent employment.

On return to England, my first job interview was with the National Physical Laboratory. There I saw the famous pilot ACE computer designed originally in consultation with Alan Turing. Its cabinets filled a vast hall; its function unit was made from thermionic valves, its few dozen immediate access registers were stored as acoustic delays in tanks filled with the metal mercury, and its main storage was on magnetic drums. Its electric power had to be produced by its own motor generator. And it cost millions of dollars, in today's monetary values. On further discussion of the proposed terms of my appointment, it turned out that the offer of the rank of senior scientific officer was a mistake. In fact, I would not be a scientific officer at all. I would be in a technician grade, as an experimental officer. And when they discovered that I did not have a scientific degree, they explained that I could only be employed on a temporary basis, and I could never hope for a permanent employment in the Scientific Civil Service. I am told that the laboratory was surprised when I declined this far-from-enticing career prospect. But the most important reason for declining was that I had lost faith in the ability of computers to translate natural languages.

Instead, I took my first job at Elliott Brothers as a programmer, writing library programs in decimal machine code for the 803 computer that I had first seen in Moscow. I really enjoyed the challenge of writing and rewriting the frequently executed parts of the program, so that they would be as fast as possible, and use efficiently the very limited memory. My only worry at the time was that my boss was only four years older than me, and his boss too was still a young man. I thought that I had obviously missed the first great wave of expansion of computing, and it would be a long time before I had any prospect of promotion. How was I to know that I would live in such interesting times? That during my working lifetime, I would see the speed of computers growing by about a million times? That they would become a thousand times cheaper and more reliable and consume a thousand times less power? That they would be a million times

smaller but contain a thousand times more internal storage capacity. That their number would grow to hundreds of millions, spread throughout the world. This astronomical increase in cost-effectiveness of computer hardware has led to a proportionate increase in the demand for programmers with the skills to put the increased power to effective use. And yet the principles of programming, which I first learned by experience forty years ago in industry, and which were the subject of all my later teaching and research in universities, are just as relevant today as they ever were, and even more so.

In my spare time from programming at Elliott's, I continued to think about the sorting algorithm that I had discovered in Moscow, to answer the question how fast it was. I wrote down a set of simultaneous difference equations governing the average number of comparisons and exchanges required. One Sunday, I was lazing on my sofa, playing idly around with the formulae. I stopped abruptly when I found that I had proved yet again the obvious fact that zero equals zero. That is what quite often happens to an amateur mathematician; and the best advice is to go back and check carefully all the previous working. When I did this, I found to my surprise that I had made no mistake: indeed I had found the correct formula that solved the equations, and gave the average speed of the sorting program. It was quite as fast as I had hoped. That encouraged me to write up an account of my method in a scientific article entitled "Quicksort," published in the *British Computer Journal* in 1962.

Ten years later, I visited Stanford University at the invitation of Don Knuth, laureate of the Kyoto Prize for Information Science in 1996 [photo 3]. He told me he had been encouraged by my analysis of the average case complexity of Quicksort, and he and his students later applied their much more powerful analytic techniques to reveal much more about the statistical distribution of the sorting times of this algorithm.

When I had been in my job at Elliotts for six months, I was given the task of designing a new high-level programming language for the new and faster members of the company's range of computers. By great good fortune, there came into my hands a copy of Peter Naur's Report on the algorithmic language ALGOL 60; and we decided to implement a subset of that [photo 4]. By an even greater good fortune, I had as my colleague the programmer Jill Pym, who had moved on to this project after implementing the autocode for Elliott's previous computer. Shortly afterwards we were married, and happily we still are.

In the early days in 1961, writing a translator for a programming language

was not obviously simpler than that of translating a natural language. Fortunately my company had a library with a good collection of journals, which I used to browse in nearly every week. There I found the issue of the *Communications of the ACM* of April 1960, the issue that is devoted to the implementation of programming languages. The article that most impressed me was one on recursive functions of symbolic expressions, written by John McCarthy, laureate of the Kyoto Prize for Computer Science and Artificial Intelligence in 1988. He gave a marvelously clear description of the first version of the purely functional programming language, LISP. It incorporated the technique of recursive programming, whereby a subroutine of a program could be defined with the help of a recursive call upon its own definition. I was amazed at the very simple LISP program that McCarthy wrote to define by interpretation any other LISP program submitted to it as data. It was much simpler than the elaborate constructions devised by Alan Turing to perform the same task for his machine. I was equally impressed at the first example in the programming manual for version 1.5 of LISP. It was a remarkably simple expression of the very algorithm that Hao Wang had used earlier to check the theorems in the first nine chapters of Russell and Whitehead's *Principia Mathematica*.

Ten years later, I worked the summer of a visit to Stanford University in John McCarthy's artificial intelligence laboratory. McCarthy's ideas on functional programming, on data structures, on axioms and on non-determinism have been the historical trigger of many later developments of the theory of these subjects.

After a thorough study of the relevant articles by McCarthy and others, I discovered the key that enabled me to design and write the Elliott ALGOL translator. It was the concept of recursion, which I had first encountered in the work of Chomsky, and met again in LISP and ALGOL 60 itself. I took full advantage of it in programming the published version of my sorting algorithm.

Our projects for the implementation of ALGOL 60 made good progress. After a year or so, I began to suspect it would soon be ready for delivery, and I reported this to my managers. Shortly after, the senior manager of the computing division had to fly off to New York, to try to recover the sale of an Elliott computer to a customer who had cancelled an order at the last minute in favour of an IBM computer. The customer was so impressed by the prospect of an ALGOL compiler on the Elliott 803, that he changed his mind back again in our favour. That recovered sale gave our little project a great

deal of internal kudos, and also something we never had before—a deadline for delivery! I am glad to report we met it; though as usual, a lot of later work was needed to make the compiler more usable in an efficient operating environment.

After the unexpected success of our ALGOL compiler, the company turned to a more ambitious project, to design and implement even more powerful and efficient operating environments for our newer and faster computers. Although I was responsible for their design, and for leading a team of around fifty programmers to implement it, after three years we had to admit that we had failed, and would not be able to deliver any part of our promises to our customers. That failure has been more influential on my subsequent career than any of my more modest successes. It has been the goal of all my subsequent research to make the writing of operating systems as easy as writing programming language translators.

But first we had to recover from the failure, and I was made responsible for that too. It took two years, after which the company turned its attention to the design of yet larger and faster machines, needed to keep up with advances in hardware technology and machine architecture. To help in this, I was removed from my position as technical manager in the development division, and appointed as chief scientist in the computing research division. With a small team of colleagues we conducted research into architectural innovations such as cache memories and paging systems, now known as virtual memory. But we realized that the critical factor in the sale of any new computer was going to be its software, and in particular its operating system. The company had still never delivered one, and still I was not quite sure even what an operating system was. So I went to visit the Cambridge University Mathematical Laboratory, to study the Titan computer and the Titan Operating System which ran there. This software had been written in collaboration with the computer manufacturer by the university staff and lecturers. The project was led by Roger Needham, under the direction of Maurice Wilkes [photo 5], laureate of the 1992 Kyoto Prize for Information Technology. I profited from work at Cambridge on cache memories, as well as insights into the structure and functions of an operating system as a whole.

In 1967, Elliott Brothers was taken over by a larger rival company, and our project to design a new computer was cancelled. The next year we were merged into an even larger company. I must have been looking at job advertisements, or I would not

have seen the vacancy for a chair in computing science at the Queen's University of Belfast. To my surprise, I applied for it; and to my even greater surprise, my application was accepted.

1968 was the start of my university career. I chose as the theme of my life-long research the fundamental problem of correctness of computer programs. I was not worried that the research would be called basic, pure, and long term. Indeed, I predicted that it would only begin to be applied in industry after I had retired from a university career. And so it has been. Last year I reached the academic age limit, and I have taken a position in Cambridge as a senior researcher in the research division of the world's leading software development company Microsoft. Maurice Wilkes is now a close neighbour to my new home in Cambridge, and Roger Needham is my immediate boss at work.

It has been a great pleasure to share with you these stories from a life in interesting times. I hope they have shown that in spite of unusual changes of direction, I have taken advantage of the teaching of great philosophers and logicians, the discoveries of early computing pioneers, together with experience derived from my own success and failure in industry, and they have led me to the exploration of fundamental theories for a new and important branch of technology. If there is any lesson for younger members of the audience, it is this: do not follow in my footsteps or in the footsteps of anybody else. Rather make a path of your own choosing, in directions which interest you most at the time. If you feel most comfortable with the established curriculum or the conventional career, I wish you the best of success in it. But if your interests or opportunities lead you in unusual directions, do not be afraid to stray from the general norm. If you think hard how to generalize your earlier learning and experiences, you will profit from them in the most unexpected ways. Everyone in this wide world has a different background and different interests and a different personality. It is this unbounded variety that we all need to foster and develop, to ensure the continued progress of human knowledge, the continuous renewal of our various cultures, and the ultimate prosperity and happiness of our human race.

photo 1

photo 2

photo 3

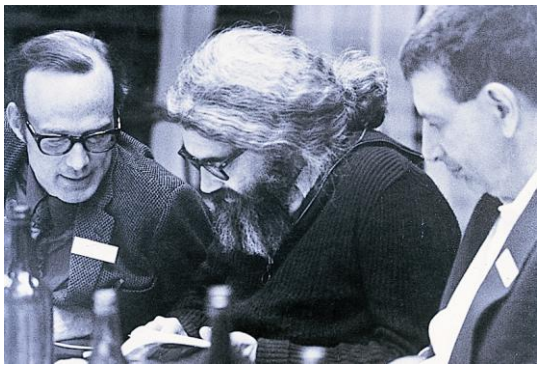

photo 4

photo 5