spare copy

Tony Hoare

From: Tony Hoare
Sent: 02 July 2000 19:11
To: Tony Hoare

ACCEPTANCE SPEECH

Advanced Technology

Tony Hoare

Your Imperial Highnesses, Ladies and Gentlemen:

It is a great honour for me in the year 2000 to receive the Kyoto prize in Advanced Technology. I am proud to be numbered among the outstanding thinkers, scientists and artists who have received a similar honour from the Inamori Foundation in the century now brought to a close. In its choice of prize-winners, the Foundation continues to show its confidence in Computer Science, not only as an intellectually challenging branch of human endeavour, but as a contributor to its noble ideals of the betterment of the whole of mankind. It is this that serves as a strong encouragement to my friends, students and colleagues in extending the boundaries of our understanding of programming theory, and also to the larger numbers of members and future entrants to professional practice in programming.

For me myself, this prize is a wonderful extra. I have already been fully rewarded by a working lifetime of excitement and achievement, spanning almost the whole of the computer era. My chosen subject of programming has interesting roots in the thinking of philosophers and logicians throughout the ages. But programming as we know it today was brought into existence by the invention and development of the stored-program digital computer. In the forty years that I have spent working both in University and in Industry, I have seen a doubling every two years of the capacity and power of computer hardware, and a similar reduction in cost. In forty years, that multiplies up to an overall improvement of a million fold, and there is still no end in sight. Such continuous improvement throws up a succession of new applications, new challenges, and new opportunities for the programmer. And yet, because my research interests and experience lie in the scientific principles of programming, and because these are inspired by the engineering ideals of integrity of structure and accuracy of detail, my early work has retained much of its relevance, while expanding in its range enormously, and enjoying vastly increased potential for application.

The growth in application of computers is even more amazing and even more important. In the beginning, most of the computer power of the world was devoted to military applications - nuclear weapon simulation, decryption of codes, battlefield logistics, and early warning of missile attack. Gradually civilian computers took over, and became indispensable tools for science, industry, commerce, government and communication. Now at last computer power is primarily in the hands of individuals, who have come to use the world wide web as their preferred medium of personal, cultural, intellectual and social interchange. More than just by the money to be made by electronic commerce, I am impressed by the generosity and altruism of those who freely populate the web-servers with useful and reliable information. Even more generous are those who freely offer their own expert advice and assistance by electronic correspondence with individual enquirers whom they will never meet. I can only applaud current initiatives to make access to the web available to people in the poorer parts of the world, so that the benefits of this altruism can be extended to the whole of mankind, enabling all individuals to participate and contribute in the exchange of ideas, information, education, discovery, and artistic creation. In this way, I hope that in the present century we will see how computing has contributed to the eternal and universal ideals which inspire the founder and the awarders of the Kyoto prize, and all who receive it.

Assertions. Tony Hoare's Laureate Lecture for the Workshop, November 12, 2000.

Summary: an assertion is a Boolean formula written in the text of a program, which the programmer asserts will always be true when that part of the program is executed. It specifies an internal interface between all of the program that comes before it and all that follows it. In the software industry today, assertions are conditionally compiled in test runs of a program, and help in the detection and diagnosis of errors. Alan Turing first proposed assertions as a means of checking a large routine. They were rediscovered independently by Naur as generalised snapshots, and by Floyd, who used them to assign meanings to programs. Floyd suggested that if the internal assertions were strong enough, they would constitute a formal proof of the correctness of a complete program. In this lecture, I will summarise the subsequent development of the idea, and describe some of its practical impact.

In the early seventies, I developed an axiomatic approach for proofs of programs, covering all the main constructions of a high-level programming language – iterations, local variables, procedures and parameters, recursion, and even jumps. Following Dijkstra, I always took a top-down view of the task of software construction, with assertions formulated as part of program specification, and with proofs conducted as part of program design. I hoped that this research would help to reduce the high costs of programming error, and the high risks of using computers in critical applications. But the real attraction for me was that the axioms underlying program proofs would provide an objective and scientific test of the quality of programming language design: a language described by a small collection of obvious rules, easily applied, would be better than one that required many rules with complex side-conditions. In collaboration with Wirth, we tried out the idea on the Pascal language; and later it inspired the design of Euclid by a team in Xerox PARC.

In scaling proof methods from small sequential algorithms to large software systems, it was necessary to extend the power of the assertion language. The Z specification language was developed by Abrial on the basis of Zermelo's set theory, which Frankel showed to be essentially adequate for expression of all concepts known to mathematics. It should therefore be adequate to express all the abstractions useful to computing, and prove the correctness of their representations. Dijkstra dealt with non-determinism, by imagining the choice to be exercised maliciously by a demon. Jones and his fellow designers of VDM included initial as well as final values of program variables. All these ideas were successfully tested by IBM in specifying the internal interfaces of a large system, CICS.

The next challenge was to extend the technology to concurrent programs. Milner suggested that their meaning could be specified by the collection of tests which they passed. Following Popper's criterion of falsifiability, Roscoe and Brookes concentrated on failures of a test, and constructed a non-deterministic model of concurrency, following the paradigm of Communicating Sequential Processes. This was applied industrially by the British start-up microchip Company Inmos in the design of the occam programming language, and the architecture of the transputer which implemented it. Finally, Hehner showed how Roscoe's results could be coded directly in the language of assertions, so that any kind of program, concurrent as well as sequential, could be interpreted as the strongest assertion that describes all its possible behaviours.

This insight has inspired all my subsequent research. With the aid of He Jifeng, it has been applied to wider varieties of programming paradigm and language, including hardware and software, parallel and sequential, declarative and procedural. Ignoring radical differences in syntax, and abstracting from implementation technology, very similar definitions and mathematical laws apply in many different paradigms; perhaps Computing Science has achieved a level of maturity to undertake the challenge that drives the progress of many other sciences, namely unification of theories of programming.

Tony Hoare's Laureate Lecture for the Workshop, November 12, 2000.

Title: Assertions.

Summary: an assertion is a Boolean formula written in the text of a program, which the programmer asserts will always be true when that part of the program is executed. It specifies an internal interface between all of the program that comes before it and all that follows. Turing first proposed assertions as a means of checking a large routine. They were rediscovered independently by Naur (as generalised snapshots) and by Floyd, who used them to assign meanings to programs. Floyd suggested that if the assertions were strong enough, it would be possible to give a formal proof of the correctness of the program. In this lecture, I will summarise the results of subsequent development of the idea, and describe some of their impact in practice.

In my early research papers, I developed a formal axiomatic approach for proofs of programs that use all the main constructions of a high-level programming language – iterations, local variables, procedures and parameters, recursion, and even jumps. Following Dijkstra, I always took a top-down view of the task of software construction, with assertions formulated as part of program specification, and with proofs conducted as part of program design. I hoped that this research would help to reduce the high costs of programming error, and the high risks of using computers in critical applications. But the real attraction for me was that the axioms underlying program proofs would provide an objective and scientific test of the quality of programming language design: a language described by a small collection of obvious rules, easily applied, would be better than one that required many rules with complex side-conditions. These ideals were pursued in the design of the Pascal language, and later Euclid.

In attempting to scale the proof methods from small sequential algorithms to large software systems, it was necessary to extend the power of the assertion language. VDM showed how to include initial as well as final values of program variables; and this idea was successfully tested by IBM on the internal interfaces of a large system, CICS. Dijkstra showed how to deal with non-determinism, by imagining the choice to be exercised maliciously by a demon. Milner suggested that the meaning of a concurrent program could be specified by the collection of tests that could be made on it. Following Popper's criterion of falsifiability, Roscoe and Brookes concentrated on failures of a test, and constructed a formal non-deterministic model of concurrency, following the paradigm of Communicating Sequential Processes. This was applied industrially by Inmos in the design of the occam programming language, and the architecture of the transputer which implemented it. Finally, Hehner showed how Roscoe's results could be coded directly in the language of assertions, so that any kind of program could be interpreted as the strongest assertion that describes all its possible behaviours.

This insight has inspired all my subsequent research. With the constant aid of He Jifeng, I have tried to apply it to wider varieties of programming paradigm and language, including both hardware and software, parallel and sequential, declarative and procedural. On ignoring radical differences in syntax and in implementation technology, it appears that very similar definitions and mathematical laws apply in many different paradigms; this has led us to suggest that Computing Science has achieved such a level of maturity that it makes sense to formulate the same challenge that drives the progress of many other sciences, and to embark upon the process of unifying theories of programming.

CURRICULUM VITAE

CHARLES ANTONY RICHARD HOARE

Date of Birth: 11th January, 1934

Education:

1948-1952	Scholar and Senior Scholar, The King's School, Canterbury
1952-1956	Exhibitioner, Merton College, Oxford
	1st Class Honours, Classical Latin and Greek
	2nd Class Honours, Philosophy and Ancient History.
1956-1958	National Service, Royal Navy. Civil Service Interpreter's qualification in Russian
1958-1959	Certificate in Statistics (with distinction)
1994-1996	British Council Visiting Student, Moscow State University

Employment:

1960-1968	Elliott Bros. (London) Ltd. Programmer, Senior Programmer,
	Chief Engineer, Technical Manager, Chief Scientist
1960-1962	Project Leader in the implementation of ALGOL 60
1968	Chief Consultant, The National Computing Centre, Manchester
1968-1977	Professor of Computing Science, The Queen's University of Belfast
1968-1970	Director of the University Computer Laboratory
1973	Visiting Professor, Stanford University
1976-1977	SERC Senior Fellow
1977-1999	Professor of Computation, then James Martin Professor of Computing at Oxford University
1984-1986, 1991-1993, 1996, 1997, 1997-1998	Director of the University Computing Laboratory
1994-1996	Admiral R. Inman Centennial Chair in Computing Theory, University of Texas at Austin.
1999-	Senior Researcher, Microsoft Research, Cambridge

Consultancies:
Computer Technology Ltd.; GEC Computers Ltd.; International Computers Ltd.; Imperial College London; Syracuse University, New York; US/European Research Office, London; Burroughs Corporation; IDA, Washington, U.S.A.; S.R.I., California, U.S.A.; CII Honeywell Bull, France; INMOS Ltd., Bristol; Rolls Royce Associates, Derby; IBM Ltd., Hursley; MCC Austin, Texas, U.S.A.; European Community Commission, Brussels; CWI, Amsterdam; BNR, Maidstone; DEC Nashua, U.S.A.; Microsoft Research, Cambridge

Research Contracts:

1969-1971	ICL/ACTP: "Operating System Techniques"
1969-1971	SERC: "Proofs of Programs"
1974-1977	SERC: "Application Orientated Languages"
1974-1977	SERC: "Program Proving Techniques"
1978-1987	SERC: "Software Engineering"
1981-1987	SERC: "Distributed Computing"
1985-1986	Computer Board: "The use of computer facilities in teaching"
1986-1988	SERC: "Industrial Software Engineering Unit"
1989-1996	EEC: Esprit BRA "Provably Correct Systems"
1989-1995	EEC: Esprit BRA "Concurrency: Unification and Extension"
1989-1993	SERC: "Safemos"
1990-1993	SERC: "Formal Methods in Medical Diagnosis"
1993-1996	SERC: "Provably Correct Hardware/ Software Design"
1994-1996	SERC: " A Design Calculus for the Resource Analysis of Real-Time System"
1994-1995	SERC: "Models Algebra & Mechanical Support in W"
1995-1998	EPSRC: "Linking Theories of Computing Science"

Service:
Member of SERC Computing Science Committee 1970-1973, Distributed Computing Panel 1977 – 1981; Member of EPSRC College, Computing and Communications 1993-1996 ; Editor of APIC Series in Data Processing (Academic Press) 1968-1977 ; Member of Editorial Boards of: Acta Informatica; IEEE Transactions in Software Engineering; Computer Programming Languages; Computer Journal; Fifth Generation Computing; Mathematical Structures in Computing Science

Member of Board of Directors, IRISA, Rennes, 1976-1978; Chairman, BCS Symposium on Software Engineering, Belfast, 1976 ; Program Co-chairman for International Conference on Reliable Software, Los Angeles, April 1975; Member IFIP W.G. 2.1 (ALGOL) 1962-1974 and W.G. 2.3 (Programming Methodology), 1969- ; Chairman ECMA TC10 (PL/I standardisation) 1967-1971 (approx); Organiser Royal Society discussion meetings: Mathematical Logic and Programming Languages 1982; Scientific Application of Multiprocessors; Mechanised Reasoning and Hardware Design.
Member of Royal Society Sectional Committee 1 (1981-1984) and Sectional Committee 4 (1995/1998) and Industrial Fellowships Panel (1995-); Editor Prentice Hall International Series in Computer Science 1977-1998 (over 100 books published); Director Marktoberdorf Summer School, 1982- ; Director: UT Year of Programming, Austin Texas 1987
Director of the Smith Institute 1996- ; Member of Academic Council, AWE Aldermaston 1998-

Lectures:
Lecturer at International Summer Schools in Villard-de-lans; Le Breau sans Nappe; Marktoberdorf (twelve occasions); Santa Cruz (twice); Newcastle; Belfast; Cambridge; Wollongong; Jindabyne; Moscow; Beijing Graduate School
Visiting Lecturer at Universities of Copenhagen, Oslo, Zurich, Chapel Hill, Austin, Wollongong, Amsterdam, Pisa.
Invited speaker at BCS branch meetings: Belfast, London, Glasgow, Stoke-on-Trent, Newcastle (Co. Down), York, Reading
Invited speaker at ACM branches and specialist groups: Boston, Mass., Phoenix, Arizona, Washington D.C.
Distinguished Lecturer: MIT Boston, Mass., U.S.A., 1976; Wang Institute, 1983; Software Engineering Institute, 1989; Organick Memorial Lectures, Utah 1989; Lee Kuan Yew Distinguished Visitor, Singapore 1992; Intel Lecturer, Jerusalem, 1993; Capstone Lecturer, Ada, Oklahoma, 1993; USC Distinguished Lecturer, 1993; Kliegel Distinguished Lecturer, Caltech, 1993; Huygens Lecture, The Hague 1998
Invited Keynote Speaker: Conference on Software Engineering, Atlanta 1978; International Conference on Foundations of Computing Science, Delhi 1985; First Turing Memorial Lecture, Glasgow 1985; Workshop on Mathematical Foundations of Programming Semantics, Boulder, 1988; Symposium on Logic in Computer Science, Edinburgh 1988; CERN School of Computing, Oxford 1988; TAPSOFT Barcelona 1989; Fifth Generation Computer Systems, Tokyo 1992; World Transputer Congress 1993 (Aachen); Fifth International Forum on the Frontier of Telecommunications Technology, Tokyo 1993; Nordic Workshop on Program Correctness, Turku 1993; Foundations of Software Engineering, Los Angeles 1993; Programming Languages and System Architectures, Zurich 1994; Third International Symposium, FTRTFS, Lübeck 1994; Euromicro, Liverpool 1994; Newton Institute, 1995; International Software Engineering Conference, Berlin 1996 ; Third International Symposium on Formal Methods Europe 1996; BCS Refinement Workshop, Bath 1996; Euro-Par, Passau 1997; POPL99 San Antonio Texas; Formal Methods International Congress 1999. ECOOP, Lisbon 1999; ACSW Canberra 2000; IFIP World Computer Congress, Beijing 2000; ICFEM, York 2000; BAAS Festival of Science 2000; IFM, Schloss Dagstuhl, 2000.
Seminar speaker at Universities and other establishments in: London, Coleraine, Dublin, Glasgow, Edinburgh, Newcastle, Warwick, Swansea, Sheffield, Bath, Cardiff, Moscow, Rehovet, Leningrad, Stanford, Berkeley, Caltech, Denver, Los Angeles, Rennes, Paris, Lisbon, Imperial College, Reading, Oxford, Cambridge.

Prizes:
ACM Programming Systems and Languages Award, 1973
Kyoto Prize, 2000

Distinctions:
Distinguished Fellow of British Computer Society, 1978; Doctor of Science honoris causa., University of Southern California, 1979; ACM Turing Award, 1980; AFIP Harry Goode Memorial Award, 1981; Elected Fellow of the Royal Society, 1982; IEE Faraday Medal, 1985; Honorary Doctor of Science, Warwick University, 1985, Pennsylvania University, 1986, Queen's University of Belfast, 1987; Foreign Member, Accademia dei Lincei, 1988; Honorary Doctor of the University of York, 1989, University of Essex, 1991; Member, Academia Europaea, 1989; IEE Computer Pioneer Award, 1991; Lee Kuan Yew Distinguished Visitor, Singapore 1992; Honorary Doctor of Science, University of Bath, 1993; Honorary Member of the Oxford Innovation Society, 1993; Corresponding Member of the Bavarian Academy of Sciences, 1997; Honorary Doctor of Oxford Brookes University, 2000; Knight Batchelor 2000

Books
[1] Structured Programming. (With E-W. Dijkstra, and 0-J. Dahl) Academic Press 1972.
[2] Communicating Sequential Processes. Prentice Hall International 1985.
[3] Essays in Computing Science. Prentice Hall International 1989.
[4] Unifying Theories of Programming. (With He Jifeng) Prentice Hall International, 1998.

Festschrift
A Classical Mind. Prentice Hall International 1994.

ACCEPTANCE SPEECH

Advanced Technology

Tony Hoare

Your Imperial Highnesses, Ladies and Gentlemen:

It is a great honour for me in the year 2000 to receive the Kyoto prize in Advanced Technology. I am proud to be numbered among the outstanding thinkers, scientists and artists who have received a similar honour from the Inamori Foundation in the century now brought to a close. In its choice of prize-winners, the Foundation continues to show its confidence in Computer Science, not only as an intellectually challenging branch of human endeavour, but as a contributor to its noble ideals of the betterment of the whole of mankind. It is this that serves as a strong encouragement to my friends, students and colleagues in extending the boundaries of our understanding of programming theory, and also to the larger numbers of members and future entrants to professional practice in programming.

For me myself, this prize is a wonderful extra. I have already been fully rewarded by a working lifetime of excitement and achievement. My subject has interesting roots in the thinking of philosophers and logicians throughout the ages. But programming as we know it today was brought into existence by the invention and development of the stored-program digital computer. In the forty years that I have spent working both in University and in Industry, I have seen a doubling every two years of the capacity and power of computer hardware, and a similar reduction in cost. That means an overall improvement of a million fold, and there is still no end in sight. Such continuous improvement throws up a succession of new applications, new challenges, and new opportunities for the programmer. And yet, because my research interests and experience lie in the scientific principles of programming, and because these are inspired by the engineering ideals of integrity of structure and accuracy of detail, my early work has retained much of its relevance, while expanding in its range and enjoying vastly increased potential for application.

The expansion in application of computers is even more amazing and even more important. In the beginning, most of the computer power of the world was devoted to military applications – nuclear weapon simulation, decryption of codes, logistics, and early warning of missile attack. Gradually computers became indispensable tools in science, industry, commerce, government and communication. But computer power is now primarily in the hands of individuals, who have come to use the world wide web as their preferred medium of personal, cultural, intellectual and social interchange. More than by the money to be made by electronic commerce, I am impressed by the generosity and altruism of those who freely populate the web-servers with useful and reliable information, and even more by those who freely offer expert advice and assistance by electronic correspondence with individual enquirers whom they will never meet. I can only applaud current initiatives to make access to the web available to people in the poorer parts of the world, so that the benefits of this altruism can be extended to the whole of mankind, enabling all individuals to participate and contribute in the exchange of ideas, information, education, discovery, and artistic creation. In this way, I hope that in the present century we will see how computing has contributed to the eternal and universal ideals which inspired the foundation of the Kyoto prize.

Profile of laureate Tony Hoare.

1. I am the eldest of a family of five: three boys and two girls. My elder brother Howard has worked all his life as a mathematician at Birmingham University. My younger brother trained as an Engineer at Cambridge, and worked as an industrial manager, and then as a Consultant. My elder sister is a teacher of mathematics, and my younger sister taught religious studies, but is now a priest in the Church of England.

2. As a child, I was studious and a bit unsociable. My school-fellows gave me the premature nickname of 'Professor', which was not always meant kindly. I was a fast and voracious reader. I learnt a lot about the world of science by reading a Children's Encyclopedia, edited by Arthur Mee. My mother taught me typing, Mah Jongg and Bridge at an early age, and I enjoyed playing those games in the family, and also chess. But I never enjoyed physical sport. I learnt knitting, and knitted elaborately patterned woollen dresses for both my sisters when they were babies.

3. I was born in Colombo, Ceylon (now Sri Lanka), but during World War II, white women and children were evacuated from the country. With my mother and brothers, we spent time in Gwelo, Southern Rhodesia (now Zimbabwe), and Durban in South Africa. Towards the end of the war we returned to Ceylon, and shortly after the war my whole family moved to England, where we stayed. My most vivid memories are of travel between these places.

4. My early education was much disrupted. On return to England at age eleven, I went with my brothers to the same primary school, the Dragon School, as my father. There I started learning Latin, Greek, French, and mathematics. After two weeks, I was doing so badly that I had to be put down into a lower year, and even then I was bottom of the class (how proud the family was at the first weekly report that I was no longer bottom!). By the end of term I had reached the top of that class, and in each subsequent term I was promoted to a higher class, till by the beginning of the next year I reached the top class in all subjects. But I was never the top of that. Because of this experience, perhaps I have been less reluctant than some to learn new ideas and languages.

5. My mother raised five children in hard times, and when my cousins were orphaned, she raised three more. She remained an active traveller into old age, and regularly visited her children's families. She left the family house as a legacy to be shared between her seventeen grandchildren; several of them still live in.

6. Like me, my father was a classical scholar at Merton College, Oxford, studying Latin, Greek, philosophy and ancient history. He made his career in the British colonial Civil Service. When Ceylon became independent in 1947, he accepted the offer of early retirement on half pension, and took a series of jobs in England, as a civil servant, as the manager of a Charity for the relief of Leprosy, and as a teacher. He qualified as a Lay preacher in the English Church, and when he retired from all other occupations, he continued to help the priest in our local parish.

7. As a child, I wanted to be an inventor. As a schoolboy, I wanted to be a writer. As a student, I wanted to be a philosopher. But when I took a job as a computer programmer with the Company Elliott Brothers, I never wanted to be anything else. Writing programs is the purest form of invention, and I remember my joy at the discovery of the sorting algorithm quicksort, and the design of a translator for the programming language ALGOL 60. When I moved to academic life, with the aid of excellent joint authors, I wrote over a hundred articles and three books. And in all my thinking and research, I have been inspired by the teachings of the great philosophers. I think, in unexpected ways, I have satisfied all my childhood ambitions.

8. I met my wife Jill at my first job, where she did much of the programming for the ALGOL translator. Our younger son Matthew had a happy childhood, and died twenty years ago of leukaemia; our memories of him are still happy. Our daughter Joanna studied Philosophy and English at Edinburgh University, and has a qualification in Psychotherapy, specialising in the application of Buddhist teaching. She has emigrated recently to set up practice in Wellington, New Zealand. Our elder son Tom studied Engineering Science at Oxford University. He is now a programmer with the British telecoms Company, Marconi in Coventry. His wife Kindy Sandhu is a programming manager in the same Company. They have two daughters, Jhansi and Maya; they are a great joy to their grandparents.

9. My interest in computers was aroused by their implications for philosophy. Perhaps they would throw light on age-old questions of human knowledge, understanding, and reason. Their capabilities as logical machines were fascinating; perhaps they were even capable of doing mathematics or understanding natural language. In my job at Elliott's our main concern was to simplify and extend the practical application of computers in science, commerce, and industrial process control. But even in these practical contexts, the design of computer software and languages is perhaps closer to philosophical speculation than any other branch of technology. I now suspect that the contribution of philosophy to computing is just as interesting as the other way round.

10. As a youngster, my interest in philosophy was roused by Bertrand Russell. His moral philosophy was appealingly unconventional, and his writing style was clear and amusing. His theory of knowledge was highly analytic, and fostered the constructive inclinations of a future computer scientist. Later, I came to appreciate his achievement in the foundation of mathematics. When I became a programmer at Elliott's, my strongest (and continuing) inspiration was the programming language ALGOL 60, for which I designed a compiler.

11. I am now again pursuing my favourite research topics, the principles of programming, and their application in the design of programming tools and languages. I am particularly interested in the practical use of assertions in programming, a topic which I first developed when I moved from Industry to take up a Professorship at the Queen's University, Belfast. Since retiring from my post as Professor at Oxford, I have returned to industrial employment at

Microsoft Research in Cambridge. Working with a group of brilliant colleagues, we have unique opportunities to promote exploitation within the Company of the results of past research of the whole community of scientists working in this field, and to direct our present and future research to goals of long-term benefit of all users of computers.

12. In 1960, when I was a graduate student at Moscow State University, I received an invitation from the National Physical Laboratory to join a British group working on the machine translation of Russian into English. At that time, dictionaries were held on magnetic tape, and for efficient look-up it was necessary to sort the words of a sentence into alphabetical order before scanning the tape. I wondered how a computer would do this. My first idea was simple, but too slow. My second idea was the algorithm which later became famous as quicksort. In my research life, I have frequently had to investigate and reject a dozen bad ideas before reaching one that I could recommend for study or implementation to others. It is the ideas that came quickly to me that I recall with greatest pleasure.

13. The advice that most influenced my scientific life came from my old friend Edsger W. Dijkstra. Computer Scientists work in a field where fashionable buzz-words have often attracted the most immediate attention, and the most generous research funding. But he has always insisted that the role of scientists in Universities is to be detached from political financial and managerial concerns; they must pursue knowledge and understanding for their own sake, and be satisfied with nothing less than excellence. In all my writing, I think of him as the reader over my shoulder, constantly monitoring the clarity of my thinking and my writing style. I still follow his central advice: in selection of research topics, do not compete in the popular races, but choose goals and methods which you alone are qualified to bring to success.

14. For recreation, I read novels, varying between modern thrillers and classical authors like Jane Austen and Charles Dickens. I also fill in crosswords. I enjoy walking in the countryside, sometimes open-air swimming, and on rare occasions when it freezes, open-air skating. And I have travelled in all continents of the world except South America and Antarctica.

15. I admire the Japanese as the heirs to an ancient civilisation, disjoint from the European tradition, but embodying many social and personal virtues that we share, and a tolerance of variety in religion, which Europe has frequently lacked. In Japanese personal etiquette, I infer precise and elegant codes of behaviour, which I fear I often fail to follow. In Japanese classical arts and music, and even in gardening, the pursuit of elegance and precision is paramount, and shows itself in modern times in the design of innovative electronic products, exported throughout the world. I have enjoyed three visits to Tokyo, and especially side trips to Kamakura and to Nara; and I have every good reason to look forward to my next visit to Kyoto and later to Kagoshima.

Stories from a life in interesting times. Commemorative lecture by Tony Hoare: summary.

I have lived in interesting times. My stories will recount the twists of fate and circumstance that moulded my thinking and my career as an information scientist. My deep thanks for the award of the Kyoto prize for the year 2000 will be best expressed if any of my listeners is warned by my mistakes or inspired by my success to contribute further to the goals of the founder of the prize – the betterment of mankind and society.

Born in Sri Lanka in 1934, I spent several years of the last world war in Zimbabwe and South Africa. In 1946, my father retired to England, and I made a late start on a traditional English education. I specialised in classical Greek and Latin, including the languages and the literature, and graduated in1956 from Oxford University in these subjects, together with ancient history and philosophy, which was always my favourite. The Russian language I learnt during two years' national service in the Royal Navy. Then I returned to Oxford as a graduate student for the Certificate in Statistics – my only official scientific or professional qualification. I was attracted to the subject by its implications for the philosophical questions of human knowledge and uncertainty. My final year as a student I spent at Moscow State University, in Kolmogorov's school of probability. There I invented the quicksort algorithm, and investigated the state of the art in computer translation of natural languages. This was the topic of my first scientific article, written and published in Russian.

Back in England in 1960, I decided to stop studying and start working. I joined a small computer manufacturing Company as a programmer. Again, I was attracted to computing by its philosophical implications, though I thought (how wrongly!) that the period of major expansion of the subject was already over. I designed a translator for an artificial language, the international algorithmic language, ALGOL 60. Promoted beyond my capabilities, I then supervised the development of an operating system that never worked, and a machine architecture that was never built. The Company was absorbed into two larger Companies, and I returned to academic life as Professor of Computing Science at the Queen's University, Belfast, to teach the subject that I had learnt by eight years of hard experience in industry.

1968 was just the start of the intensive period of the civil disturbances in Northern Ireland that are only now subsiding. In spite of this, I was able to recruit a strong team to the computing department, to meet a rapidly expanding demand for education in the subject. My research had a practical goal: I wanted to understand the reasons for the success and the failure of my earlier projects in industry. In tackling the problems, I took a long-term, almost philosophical view. I planned a research agenda that would last throughout my academic life, because I knew its results would hardly be ripe for application in industry until after my retirement some thirty years ahead. In 1977, I moved to the chair of Computation at Oxford, where again, from smaller beginnings, I assembled the funds and recruited a team to introduce the subject of computing into the curricula of the University. I retired last year as senior Professor in the Faculty of Mathematics, to move back into industry.

I am now working as a senior researcher for Microsoft Research at Cambridge. Here I will pursue my early hope that the results of pure research, my own and that of others, may be applied by those engaged in the writing of large-scale computer software, for the benefit of the increasing numbers of those who use it. Perhaps in the years to come, that will be the whole of mankind.

Stories from a life in interesting times. Commemorative lecture by Tony Hoare: summary.

I have lived in interesting times. My stories will recount the twists of fate and circumstance that moulded my career as an information scientist. My deep thanks for the award of the Kyoto prize for the year 2000 will be best expressed if any of my listeners is warned by my mistakes or inspired by my success to contribute further to the goals of the founder of the prize – the betterment of mankind and society.

Born in Sri Lanka in 1934, I spent several years of the last world war in Zimbabwe and South Africa. In 1946, my father retired to England, and I made a late start on a traditional English education. I specialised in classical Greek and Latin, including the languages and the literature, and graduated in1956 from Oxford University in these subjects, together with ancient history and philosophy, which was always my favourite. The Russian language I learnt during two years' national service in the Royal Navy. Then I returned to Oxford as a graduate student for the Certificate in Statistics – my only official scientific or professional qualification. I was attracted to the subject by its implications for the philosophical questions of human knowledge and uncertainty. My final year as a student I spent at Moscow State University, in Kolmogorov's school of probability. There I invented the quicksort algorithm, and investigated the state of the art in computer translation of natural languages. This was the topic of my first scientific article, written and published in Russian.

Back in England in 1960, I decided to end my student days by joining a small computer manufacturing Company as a programmer. Again, I was attracted to computing by its philosophical implications, though I thought (how wrongly!) that the period of major expansion of the subject was already over. I designed a translator for an artificial language, the international algorithmic language, ALGOL 60. Promoted beyond my capabilities, I then supervised the development of an operating system that never worked, and a machine architecture that was never built. The Company was absorbed into two larger Companies, and I returned to academic life as Professor of Computing Science at the Queen's University, Belfast, to teach the subject that I had learnt by eight years of hard experience in industry.

1968 was just the start of the intensive period of the civil disturbances in Northern Ireland that are only now subsiding. In spite of this, I was able to recruit a strong team to the computing department, to meet a rapidly expanding demand for education in the subject. My research had a practical goal: I wanted to understand the reasons for the success and the failure of my earlier projects in industry. In tackling the problems, I took a long-term, almost philosophical view. I was thinking of research agenda that would last throughout my academic life, because I knew its results would hardly be ripe for application in industry until after my retirement some thirty years ahead. In 1977, I moved to the chair of Computation at Oxford, where again, from smaller beginnings, I assembled the funds and recruited a strong team to introduce the subject of computing into the curricula of the University. I retired last year as senior Professor in the Faculty of Mathematics, to move back into industry.

I am now working as a senior researcher for Microsoft Research at Cambridge. Here I will pursue my early hope that the results of pure research, my own and that of others, may be applied by those engaged in the writing of large-scale computer software, for the benefit of the increasing numbers of those who use it. Perhaps in the years to come, that will be the whole of mankind.

CURRICULUM VITAE

CHARLES ANTONY RICHARD HOARE

Date of Birth: 11th January, 1934

Education:

1947-1952	Scholar and Senior Scholar, The King's School, Canterbury
1952-1956	Exhibitioner, Merton College, Oxford
	1st Class Honours, Classical Latin and Greek
	2nd Class Honours, Philosophy and Ancient History.
1956-1958	National Service, Royal Navy. Civil Service Interpreter's qualification in Russian
1958-1959	Certificate in Statistics (with distinction)
1959-1960	British Council Visiting Student, Moscow State University

Employment:

1960-1968	Elliott Bros. (London) Ltd. Programmer, Senior Programmer, Chief Engineer, Technical Manager, Chief Scientist
1960-1962	Project Leader in the implementation of ALGOL 60
1968	Chief Consultant, The National Computing Centre, Manchester
1968-1977	Professor of Computing Science, The Queen's University of Belfast
1968-1970	Director of the University Computer Laboratory
1973	Visiting Professor, Stanford University
1976-1977	SERC Senior Fellow
1977-1999	Professor of Computation, then James Martin Professor of Computing at Oxford University
1984-1986, 1991-1993, 1996, 1997, 1997-1998	Director of the University Computing Laboratory
1994-1996	Admiral R. Inman Centennial Chair in Computing Theory, University of Texas at Austin.
Currently	Senior Researcher, Microsoft Research, Cambridge

Consultancies:
Computer Technology Ltd.; GEC Computers Ltd.; International Computers Ltd.; Imperial College London; Syracuse University, New York; US/European Research Office, London; Burroughs Corporation; IDA, Washington, U.S.A.; S.R.I., California, U.S.A.; CII Honeywell Bull, France; INMOS Ltd., Bristol; Rolls Royce Associates, Derby; IBM Ltd., Hursley; MCC Austin, Texas, U.S.A.; European Community Commission, Brussels; CWI, Amsterdam; BNR, Maidstone; DEC Nashua, U.S.A.; Microsoft Research, Cambridge

Research Contracts:
1969-1971 ICL/ACTP: "Operating System Techniques"; 1969-1971 SERC: "Proofs of Programs"; 1974-1977 SERC: "Application Orientated Languages"; 1974-1977SERC: "Program Proving Techniques"; 1978-1987 SERC: "Software Engineering"; 1981-1987 SERC: "Distributed Computing"; 1985-1986 Computer Board: "The use of computer facilities in teaching"; 1986-1988 SERC: "Industrial Software Engineering Unit"; 1989-1996 EEC: Esprit BRA "Provably Correct Systems"; 1989-1995 EEC: Esprit BRA "Concurrency: Unification and Extension"; 1989-1993 SERC: "Safemos"; 1990-1993 SERC: "Formal Methods in Medical Diagnosis"; 1993-1996 SERC: "Provably Correct Hardware/ Software Design"; 1994-1996 SERC: " A Design Calculus for the Resource Analysis of Real-Time System"; 1994-1995 SERC: "Models Algebra & Mechanical Support in W"; 1995-1998 EPSRC: "Linking Theories of Computing Science"

Service:
Member of SERC Computing Science Committee 1970-1973, Distributed Computing Panel 1977 – 1981; Member of EPSRC College, Computing and Communications 1993-1996 ; Editor of APIC Series in Data Processing (Academic Press) 1968-1977 ; Member of Editorial Boards of: Acta Informatica; IEEE Transactions in Software Engineering; Computer Programming Languages; Computer Journal; Fifth Generation Computing; Mathematical Structures in Computing Science Member of Board of Directors, IRISA, Rennes, 1976-1978; Chairman, BCS Symposium on Software Engineering, Belfast, 1976 ; Program Co-chairman for International Conference on Reliable Software, Los Angeles, April 1975; Member IFIP W.G. 2.1 (ALGOL) 1962-1974 and W.G. 2.3 (Programming Methodology), 1969- ; Chairman ECMA TC10 (PL/I standardisation) 1967-1971 (approx); Organiser Royal Society discussion meetings: Mathematical Logic and Programming Languages 1982; Scientific Application of Multiprocessors; Mechanised Reasoning and Hardware Design.

Member of Royal Society Sectional Committee 1 (1981-1984) and Sectional Committee 4 (1995/1998) and Industrial Fellowships Panel (1995-); Editor Prentice Hall International Series in Computer Science 1977-1998 (over 100 books published); Director Marktoberdorf Summer School, 1982- ; Director: UT Year of Programming, Austin Texas 1987 Director of the Smith Institute 1996- ; Member of Academic Council, AWE Aldermaston 1998-

Lectures:
Lecturer at International Summer Schools in Villard-de-lans; Le Breau sans Nappe; Marktoberdorf (twelve occasions); Santa Cruz (twice); Newcastle; Belfast; Cambridge; Wollongong; Jindabyne; Moscow; Beijing Graduate School
Visiting Lecturer at Universities of Copenhagen, Oslo, Zurich, Chapel Hill, Austin, Wollongong, Amsterdam, Pisa.
Invited speaker at BCS branch meetings: Belfast, London, Glasgow, Stoke-on-Trent, Newcastle (Co. Down), York, Reading . Invited speaker at ACM branches and specialist groups: Boston, Mass., Phoenix, Arizona, Washington D.C.

Distinguished Lecturer: MIT Boston, Mass., U.S.A., 1976; Wang Institute, 1983; Software Engineering Institute, 1989; Organick Memorial Lectures, Utah 1989; Lee Kuan Yew Distinguished Visitor, Singapore 1992; Intel Lecturer, Jerusalem, 1993; Capstone Lecturer, Ada, Oklahoma, 1993; USC Distinguished Lecturer, 1993; Kliegel Distinguished Lecturer, Caltech, 1993; Huygens Lecture, The Hague 1998

Invited Keynote Speaker: Conference on Software Engineering, Atlanta 1978; International Conference on Foundations of Computing Science, Delhi 1985; First Turing Memorial Lecture, Glasgow 1985; Workshop on Mathematical Foundations of Programming Semantics, Boulder, 1988; Symposium on Logic in Computer Science, Edinburgh 1988; CERN School of Computing, Oxford 1988; TAPSOFT Barcelona 1989; Fifth Generation Computer Systems, Tokyo 1992; World Transputer Congress 1993 (Aachen); Fifth International Forum on the Frontier of Telecommunications Technology, Tokyo 1993; Nordic Workshop on Program Correctness, Turku 1993; Foundations of Software Engineering, Los Angeles 1993; Programming Languages and System Architectures, Zurich 1994; Third International Symposium, FTRTFS, Lübeck 1994; Euromicro, Liverpool 1994; Newton Institute, 1995; International Software Engineering Conference, Berlin 1996 ; Third International Symposium on Formal Methods Europe 1996; BCS Refinement Workshop, Bath 1996; Euro-Par, Passau 1997; POPL99 San Antonio Texas; Formal Methods International Congress 1999. ECOOP, Lisbon 1999; ACSW Canberra 2000; IFIP World Computer Congress, Beijing 2000; ICFEM, York 2000; BAAS Festival of Science 2000; IFM, Schloss Dagstuhl, 2000.

Seminar speaker at Universities and other establishments in: London, Coleraine, Dublin, Glasgow, Edinburgh, Newcastle, Warwick, Swansea, Sheffield, Bath, Cardiff, Moscow, Kiev, Novosibirsk, Rehovet, Leningrad, Stanford, Berkeley, Caltech, Denver, Los Angeles, Rennes, Paris, Lisbon, Imperial College, Reading, Oxford, Cambridge.

Prizes:
ACM Programming Systems and Languages Award, 1973
Kyoto Prize, 2000

Distinctions:
Distinguished Fellow of British Computer Society, 1978; Doctor of Science honoris causa., University of Southern California, 1979; ACM Turing Award, 1980; AFIP Harry Goode Memorial Award, 1981; Elected Fellow of the Royal Society, 1982; IEE Faraday Medal, 1985; Honorary Doctor of Science, Warwick University, 1985, Pennsylvania University, 1986, Queen's University of Belfast, 1987; Foreign Member, Accademia dei Lincei, 1988; Honorary Doctor of the University of York, 1989, University of Essex, 1991; Member, Academia Europaea, 1989; IEE Computer Pioneer Award, 1991; Lee Kuan Yew Distinguished Visitor, Singapore 1992; Honorary Doctor of Science, University of Bath, 1993; Honorary Member of the Oxford Innovation Society, 1993; Corresponding Member of the Bavarian Academy of Sciences, 1997; Honorary Doctor of Oxford Brookes University, 2000; Knight Batchelor 2000

Books

[1] Structured Programming. (With E-W. Dijkstra, and 0-J. Dahl) Academic Press 1972.
[2] Communicating Sequential Processes. Prentice Hall International 1985.
[3] Essays in Computing Science. Prentice Hall International 1989.
[4] Unifying Theories of Programming. (With He Jifeng) Prentice Hall International, 1998.

Festschrift

A Classical Mind. Prentice Hall International 1994.