University Education on Computers

Summary of a Panel Discussion

Reino Kurki-Suonio[1] (chair), Oddur Benediktsson[2], Janis Bubenko, Jr.[3], Ingemar Dahlstrand[4], Christian Gram[5], and John Impagliazzo[6]

[1] Tampere University of Technology (Emeritus), Finland, reino.kurki-suonio@tut.fi
[2] University of Iceland, oddur@hi.is
[3] Royal Institute of Technology (Emeritus), Stockholm, Sweden, janis@dsv.su.se
[4] University of Lund, Sweden (Emeritus)
[5] Technical University of Denmark (Emeritus), Copenhagen, Denmark, chr.gram@ddf.dk
[6] Qatar University, Doha, Qatar, john@qu.edu.qa

Abstract: Following a session on university education, this panel discussed early Nordic visions and experiences on university computing education, contrasting them to today's needs and the international development at that time. This report gives short papers by the panelists (their opening statements), and a brief summary (the chair's interpretation) of the views that were raised in the ensuing discussion.

Keywords: Nordic university computing education, university computing education

1. Introductory Remarks

by *Reino Kurki-Suonio*

More than forty years have passed since computer science and related topics were introduced as academic disciplines, even though universities had already used computers for some time and students had already experienced programming courses. The first professors in these fields had started their work in computing practice more than forty-five years ago. To obtain a quantitative idea of the way the world has changed since then, notice that the factor computed by Moore's law for forty-five years is no less than one billion; that is, it is a factor of ten to the ninth!

Trying to imagine ourselves in those times, we remember also that computers were centralized facilities, operated by special personnel in the batch mode. Computers were far too expensive and it was difficult to justify their use for educational purposes or for computing-related research; they were primarily purchased for more "serious" use as tools in number crunching and/or administrative data processing.

Another major difference is that in its infancy the world of computing was still rather homogeneous. Until the mid-1970s, "all" computer people – system and application developers, academicians and practitioners – gathered at IFIP world congresses, for instance, and they were still listening to each other with interest and with more or less good understanding. In other words, the many disciplines that were inspired by information technology were just emerging and had not yet developed their separating paradigms, no specialized conference or workshop series existed, and the gulf between theory and practice was still narrow.

This panel presentation has provided a spectrum of viewpoints of computing at universities from different countries and from different periods. The overall collection total of these points of view has created an interesting dialogue useful to computing history and history specialists.

2. Early Development in Finland

by *Reino Kurki-Suonio*

My own university career started in 1965 at the University of Tampere and continued from 1980 until 2002 at Tampere University of Technology. In both places, I had the privilege of developing degree-based education in computing from scratch. When I became involved with this, I had five years experience in application design and implementation as well as in programming education at Finnish Cable Works, one of the roots of today's Nokia. My ideas of computing education were, however, strongly influenced by a post-doctoral year at Carnegie Institute of Technology (now Carnegie Mellon University), where the computer science department was just beginning to start in a formal way.

As discussed in more detail in [8], in Finland the first chair in computing was established by a surprise move in 1965 by the University of Tampere – then still a School of Social Sciences – at which point I suddenly found myself involved in designing an academic curriculum for an emerging discipline. This activity was not a result of gradual development. Since we did not have much of a model to use, I had a rather free hand in the curriculum design.

Contrary to what we had heard about the controversy between a tool and a discipline in Norway [10], this was not a problem in Tampere since the university did not have natural sciences or technology with large number crunching needs. Later, however, this controversy was strongly reflected in the acquisition of computers, as is apparent in [11]. In any case, in computing education we were definitely going for a new discipline, which we felt to be of fundamental importance to human civilization.

For the core of this discipline, I considered expressing of complex algorithmic processes. It was clear that, as an academic discipline in an area with much practical importance, the curriculum should combine practical skills with theoretical understanding. Of course, this "motherhood" statement was never easy to implement, since "one man's theory is another man's practice". In addition, much of what practitioners criticized as being too theoretical in our curriculum is now pure practice.

Although the field then was much more homogeneous than today, computing practice had two important lines of separation: one between s*cientific computing* and *administrative data processing*, and the other between *programming* and *system analysis and design*. In my mind, the emerging discipline should do away with these differences. I felt that the discipline is much more fundamental than using computers as tools in certain applications, and that results in its core areas are application-independent. Additionally, I also strongly opposed the common view of practitioners that programming is a low-level activity of coding, or the technical mastery of one or more programming languages.

In designing and then implementing the curriculum, my colleague, Miikka Jahnukainen, assisted me. He had already been involved with a plan to educate system analysts and designers for administrative data processing. We felt that it was a good idea to expose the students to our complementary views on what was most essential for the students. I could concentrate on algorithmic processes, data structures, principles of programming languages and operating systems, and other aspects of the young computer science, whereas Jahnukainen's approach better prepared students for the more mundane practices of the ADP departments and led to the Scandinavian direction of "systemeering". However, to my disappointment, the two views seldom merged successfully in the students' minds. My idealism was also shaken by the experience that so many of the students – especially ones who wanted to specialize in administrative system design – had tremendous difficulties in passing the courses that I considered to be the core of the discipline.

In any case, when further Finnish universities followed us in starting their computer science and related departments, we had already gained some experiences that they could utilize, in addition to the international models that then started to be available.

3. A Swedish Perspective

by *Janis Bubenko, Jr.*

In the early days of computing, the 1950s and the 1960s, researchers and practitioners had different visions about computing in the future. Swedish researchers' vision was the continued use of large computers, precise application problem formulation in high-level, declarative languages followed by "code generation" and optimization. We believed in the development of advanced tools for design and generation of information systems. We also believed we would/could develop a comprehensive "theory of information systems development". However, we totally underestimated the complexity of such an undertaking. It is important to note that, in the 1960s, our vision of future information technology did not include (1) personal and personally owned and portable computers, (2) data communication development and the internet, (3) security threats and problems, and (4) the development of commercial off-the-shelf (COTS) software and hardware. We could hardly have imagined that these things were possible.

Today, university education in information technology has become specialized in many different directions such as theoretical aspects (computer science), databases, information systems, software engineering, requirements engineering, human-computer interaction, and many other specialties. However, in most of these specialties we still consider certain fundamental topics as essential. Some of them include modeling of "object systems" (applications) using different types of models (conceptual, object-oriented, process, rules, etc.), programming languages of different kinds, algorithms and data structures, mathematics and logics, and design, testing and proofs of programs.

Major changes during the last twenty years that have affected the needs of academic computing education are the personal computer and advances in software and hardware technology, advances in telecommunications, the internet technology including search technologies. These developments have changed our vision of the way we can build future systems and the way our vision affects the way future systems may influence our daily lives.

What should be the proper role of computer science and other theoretical bases in computing education and research today? The complexity of systems is increasing. Systems increasingly experience "bad input" and hostile attacks. The need to build systems with "a correct and safe (robust) behavior" is increasing. The need for interfaces designed in such a way that non-computer experts can use them is obvious.

As a contrast, the use of formal methods, mathematics, and logic in computing education seems decreasing. In some Swedish colleges, for dubious reasons, some advanced theoretical topics in computing have been "dropped" in order to attract more students to information technology. We should never forget that systems and program development is much more than "front page design". Unfortunately, few companies of today understand the importance and need for higher-level theoretical knowledge within the computing field.

4. University Computer Science Education in Denmark Before 1970

by *Christian Gram*

4.1 The Very Beginning

Before 1962, no regular curriculum in computer science existed at universities. However, we do know that universities offered several extra courses for both students and academic staff. Some of the topics included programming in assembly language, Algol or FORTRAN, which departments then supplemented with courses in numerical technical calculation.

The first regular courses for students emerged in the early 1960s at the universities and they centered on departments of mathematics; at the technical universities, they centered on departments of electronics. As an example, the first computer-related course at Copenhagen University was "Mathematics 4", and the

contents of the course (a) Programming in Algol, (b) Numerical analysis of problems in linear equation solving, numerical integration, and root finding.

4.2 The First Plan

Through his work on compiler construction and on EDP applications, Peter Naur became convinced that some common basic principles lay behind all data processing and use of computation. In 1966, while he worked as a senior consultant at Regnecentralen, he published a red booklet with 64 pages called "A Plan for a Course in Datalogy and Datamatics" [9]. The booklet outlined what a general course in computer science should contain. The preamble stated that "Datalogy is as fundamental as language and mathematics in education", and some knowledge of programming must be taught early. The plan proceeded by describing in some detail six major areas:

- Concepts and methodology for datalogy; computers; data processes.
- Single data elements; dealing with data representation; numbers and arithmetics; classification and choices.
- Medium size data sets; problems concerned with searching and sorting; sequential analysis of text; arithmetic expressions; list structures.
- Communication between man and computer; format of input data; output representations; dialogue between man and machine.
- Large data sets and file transactions; processing efficiency; utilization of sequential secondary storage; searching on secondary storage media.
- Development of large programs; consideration of safety problems; ways to plan and develop large programs.

Using this plan as the list of contents, Naur and a group of colleagues at Regnecentralen planned to write a textbook containing eighteen chapters. It was our conviction at the time that new textbooks were essential; the management at Regnecentralen supported this belief. In 1967-69, the group wrote thirteen of the planned eighteen chapters; however, they never finished the last five chapters. The project stopped because in 1969, Copenhagen University appointed Naur as professor in Datalogy. At the same time, Regnecentralen moved toward a more business-oriented direction. However, the material they developed influenced the computer science curricula created in the late 1960s.

4.3 Comparison with Today

The tables below show a comparison between computer science courses of today and courses in 1968. The column "Typical 2007 Courses" contains course titles from a typical computer science curriculum 2007. The column "Corresponding Titles 1968" shows, where similar courses existed already around 1968 and where methods, principles, or technology were still under development.

Table 1 mentions some of today's courses, which are more or less similar to courses that already had existed in the late 1960s. Table 2 contains several modern courses that had no obvious parallel in the old days. For many of the

courses, the technology was not yet available; for other courses, the theory was still under development; in a few cases, the topics simply did not exist in the 1960s.

Table 1

Typical 2007 Courses	**Corresponding Titles 1968**	**Remarks**
Intro to Mathematics	(same)	
Linear Algebra	(same)	
Advanced Algorithms	(same)	
Compilers	(same)	
Types and Programming Languages	Programming Languages	
Functional Programming	LISP	LISP was the only functional language
Advanced Databases	Databases	
Operating Systems	(same)	
Object-oriented Programming and Design	Design of EDP Systems	The term "object-oriented" was not invented, but design was dealt with much the same way as today
Optimizing in Production Planning	System Analysis, Optimization	Not exactly the same course, but much of the same flavor
Computer Architecture	(same)	Very similar courses, even if technology differed
Man-Machine Interaction	Input/Output Formatting	The term "interaction" in today's meaning was not possible; the emphasis was on user-friendly input/output
Artificial Intelligence	(same)	Courses existed in late 1960s, but the contents were much different from today's courses

Table 2

Typical 2007 Courses	**Corresponding Titles 1968**	**Remarks**
Intro to Graphics		No graphical media existed
Intro to Image Processing		No means for image manipulation existed
Logic: Models and Proofs		Prolog courses began to appear in curricula in the 1970s
Software Engineering		The term was not invented until 1968
Computation and Deduction		Theoretical computer science courses were not established

Cryptography and Security		Problems around secure EDP were not yet on the agenda
Reversible Computation		Theories were not developed
Algorithmic Geometry		Mathematicians had not yet started to use computers in geometry
Data net	(Data Transmission)	The term "data net" was not invented, but one-to-one transmission was used and taught
Intro to Distributed Systems		Distributed systems were not invented
Chip Design	(Circuit Analysis and Design)	Systems and methods for chip design did not exist
Robot Experimentation		Robots did not really appear in courses until the 1980s

5. Experiences in Lund

by Ingemar Dahlstrand

In 1985, I entered academia at Lund Technical College (LTH), so I did not take part in the early build-up, except for programming courses in machine code and later Algol. When I did become a teacher after much practical experience in industry, I found that the education offered at LTH was at a strong level, both practically and theoretically. In the 1960s, I had thought that getting research started was more urgent than mass education because computer scientists outside the Stockholm area had a rather poor job market. The question always had emerged as to whether computer science was a science in its own right. My response is emphatically "yes". This is the first time in civilization that we learn to instruct a completely obedient apparatus, and we are finding it surprisingly difficult. Our department at LTH offered programs in both numerical analysis and in computer science, but that was for historical reasons.

Our students sometimes complained that they wanted to learn C++ because that was what industry used. Actually, industry asked the faculty to teach students foundations and problem solving; for commercial usage, industry was prepared to teach specialized topics themselves. A computer scientist should know the difference between a good method and a poor one, even if he or she must use the latter for a while. We had a seminar once at a national conference; it started out with the question: Does computer science build upon its foundations such as computability, program proving, and the Turing machine? I do not think it always does.

6. The Start of Computer Science Education in Iceland

by Oddur Benediktsson

With the acquisition of an IBM 1620 Model 2 computer in 1964, the University of Iceland entered the computer age. Programming became part of the engineering curriculum in the following year. The programming language used was

FORTRAN II. Programming became a required component of an "applied mathematics" course in the engineering curriculum. At that time, only the first three years of the engineering studies could be completed in Iceland; students would go abroad to finish their studies.

In the academic year 1972-73, a new three-year sequence of study, the BS in Applied Mathematics, became a curriculum in the Mathematics Department at the University of Iceland. The core curriculum consisted of mathematical analysis, algebra, and statistics, in addition to computer science, numerical analysis, and operations research. The curriculum was partly based on the recommendations of the ACM Curriculum Committee on Computer Science "Curriculum 68: Recommendations for the Undergraduate Program in Computer Science" [2].

Computer science became a separate three-year BS degree program at the University of Iceland in 1976. The Mathematics Department housed the degree program in computer science; the program remained there for the subsequent ten years, before becoming an independent department.

The following table shows the first computer language taught to engineering and science students at the university and the computer systems used.

Table 3

Period	Computer system	First language
1965 – 1975	IBM 1620	FORTRAN II
1976 – 1978	IBM 360/30 and PDP 11	FORTRAN IV
1979 – 1982	DEC VAX-11	FORTRAN 77
1983 – 1986	DEC VAX-11 and PCs on net	FORTRAN 77 and Modula-2
1987 – 1990	DEC VAX-11 and PCs on net	FORTRAN 77 and Turbo Pascal
1990 – 1996	Unix servers and PCs on net	C++ and Turbo Pascal
1997 – 2006	Unix servers and PCs on net	Java and MATLAB

It was noted that the first computer language taught at a university could have a profound effect on the students involved, since the first language often becomes a tool used for the entire working life.

7. A U.S. Perspective with International Overtones

by John Impagliazzo

During the 1950s and the early 1960s, the United States began to generate courses associated with data processing primarily targeted toward technical (two-year) colleges. By 1965, ACM had published a paper that was a preview of the well-known Computing Curriculum'68. By the 1970s, we witnessed literature regarding graduate and undergraduate information systems programs, which culminated with Computing Curriculum'78. By the 1980s, we saw literature on discrete mathematics and programming courses, now coined as CS1 and CS2 as well as curricula recommendations for information systems and computer engineering.

7.1 The "First" U.S. Computer Science Department

It is always dangerous to speculate "firsts" when it comes to history, particularly computing history, because many institutions of higher learning explore innovative learning and teaching, particularly during the 1960s. Notwithstanding, it does appear that Purdue University was a leader at least in one area. The Purdue website states:

"The first Department of Computer Sciences in the United States was established at Purdue University in October 1962. There are three natural phases in its history. In the 1960s the effort was to define courses, degree programs, and indirectly the field itself."

During that time, the university hired five faculty members in the first year for its graduate program, which was part of the Division of Mathematical Sciences within the departments of mathematics and statistics. At first, computer science was an option in the mathematics and later became a separate B.S. degree in 1967.

7.2 Emergence of a National Computing Curriculum

By the mid-1960s, much activity ensued in curriculum development. ACM had established a Curriculum Committee on Computer Science (C3S). This group had been considering curriculum problems for approximately three years. During the early part of this period, the committee held a number of informal sessions with computer people at various national meetings. In the latter part of this three-year period, ACM formally organized the committee, where it made a definite effort to arrive at concrete suggestions for a curriculum. In 1965, the group published a paper [1], which became the precursor to Curriculum'68.

Other movements began to emerge during the 1960s. In February of 1967, the President created a Science Advisory Commission (SAC) that focused on the use of computers in higher education. The Computer Sciences in Electrical Engineering (COSINE) Committee explored the ways in which computer science would be part of electrical engineering that then led to the establishment of a Commission on Engineering Education in September of 1967 in Washington DC [5]. The question of recognition became a topic of discussion concerning whether the emerging discipline of computing was legitimate in its own right. Lofti Zadeh placed a marker on that topic with his landmark paper on the subject [12].

7.3 ACM Curriculum'68

The synergies that existed in the mid-1960s gave rise the very well known publication of the ACM Curriculum'68: Recommendations for Academic Programs in Computer Science. It was no accident that Curriculum`68 closely resembled the degree program at Purdue; indeed, Purdue was a test bed for developing recommendation. The published computer science curriculum contained three divisions for computer science that included:

- Information Structure and Processes (data structures, programming languages, methods of computations),

- Information Processing Systems (computer design and organization, translators and interpreters, computer and operating systems, special purpose systems), and
- Methodologies (numerical mathematics, data processing and file management, symbol manipulation, text processing, computer graphics, simulation, information retrieval, artificial intelligence, process control, instructional systems).

Curriculum'68 also included recommendations for mathematics and the sciences. From the mathematical sciences, the curriculum recommended elementary analysis, linear algebra, differential equations, algebraic structures, numerical analysis, applied mathematics, optimization theory, combinatorics, mathematical logic, number theory, probability and statistics, operational analysis. From the physical and engineering sciences the curriculum recommended general physics, basic electronics, circuit analysis and design, thermodynamics, system mechanics, field theory, digital and pulse circuits, coding and information theory, communication and control theory, and quantum mechanics.

Curriculum'68 enjoyed a high degree of initial success. Many universities, nationally and internationally, that had an interest in establishing a computer science department began using it as a reference, at least as a starting point. However, it was not long before the recommendation showed some of its frailties and began to receive criticism. By 1974, published documents called for a revision of Curriculum'68 [6]. This paper claimed among things that the 1968 report did not address the nature of computer science, it did not address the subject matter for a complete bachelor's program, and it did not address articulation between technical and university programs. In addition, specific courses mentioned such as discrete structures, switching theory, and sequential machines seemed isolated, and many courses *not* mentioned in the 1968 report already existed in many computing programs.

A follow up article in 1976 [7] addressed what a computer science major should be able to do rather what courses a student should take. These attributes included an ability to (1) write correct, documented, readable programs in a reasonable time, (2) determine whether written programs are reasonably efficient and well organized, (3) know what types of problems are amenable to computer solution, (4) make reasonable judgments about hardware; and (5) pursue in depth training in one or more application areas. The strong undercurrent toward curriculum reform soon led to a formal revision of the battered 1968 report.

7.4 ACM Curriculum'78

To address the needs of the computing community, ACM created a new committee to overhaul the former curriculum report. The committee created a new report called Curriculum'78 [3] and developed themes of concentration that included computer programming I and II, computer systems, computer organization, file processing, operating systems and architecture, data structures and algorithm analysis, programming languages (overview and theory), computers

and society, database management systems, artificial intelligence, algorithms, software design, automata, computability, formal languages, and numerical mathematics. Curriculum'78 was similar to Curriculum'68; however, the new version stressed greater adherence on software and treated hardware in a more general way. Many universities around the world adopted the framework Curriculum'78. After three decades of use and the development of new technologies, with few modifications, many computing programs in existence today reflect a strong association with the curriculum report from 1978. The curriculum seems to have endured the test of time.

7.5 Further Evolutions

Despite its level of success, Curriculum'78 soon was to come under scrutiny and would not be satisfactory to the greater computing community. The 1980s witnessed a flood of new curricula recommendations, particularly from the Data Processing Management Association (DPMA), which today is the Association for Information Technology Professionals (AITP), and from the Computer Society of the Institute for Electrical and Electronic Engineers (IEEE). Some of these reports include:

- DPMA Educational Programs and Information Systems (1981).
- ACM Information Systems Recommendations – Undergraduate & Graduate Programs (1983).
- IEEE Computer Society Model Curriculum – Computer Engineering (1983).
- DPMA Information Technology and Systems (1984).
- DPMA Associate (Two-Year) Level Model Curriculum – Information Systems (1985).
- DPMA Model Curriculum – Information Systems (1985).

The mid-1980s witnessed great debates on the subject of computing. Some of the debate centered upon whether computer science was indeed a science as opposed to being a part of engineering or a mathematics discipline – or neither of these. The culmination of the debates resulted in a new computer science curriculum recommendations called Curriculum'91 [4]. The next fifteen years saw major changes in curricula development on all computing areas. The details of these developments are beyond the scope of this narrative.

8. Discussion Summary

Arne Sølvberg from the Norwegian University of Science and Technology, Trondheim, briefly commented on the background of Norwegian computing history in the discussion. He identified two important Scandinavian sources of inspiration in the early development: work on programming languages and compilers by Peter Naur's group in Denmark, and Langefors' approach to information systems engineering in Sweden. As a major change to the early

situation, Sølvberg indicated that computing departments are now well established and they no longer have to defend their existence. Although there is not so much change in the foundational courses, we find that much of the earlier curriculum content appears in other disciplines, which has an effect on the relationship of computing departments to other departments.

This led to a discussion on some factors that call for changes in today's programs and may even affect their viability. We see diminishing numbers of students and increased problems in getting good ones; we must address the interdisciplinary nature of computing and students' ability to use computers, even though they need not know how they work inside. As a response, people suggested that we must create new kinds of programs, where computing may be combined with other areas such as art. Instead of programming languages, we may have to use multimedia as the central role in the new computing approaches.

A brief exchange between Ingemar Dahlstrand and Janis Bubenko, Jr., brought up a contrast that is important in computing education. In computer science, we are interested in making the machine do exactly what we want it to do, whereas in system design a major problem is to determine what we want the computer to do.

Regarding new kinds of programs, Enn Tyugu referred to specialized computing programs, as those in bioinformatics. Christian Gram mentioned the rise of "IT high schools" and an IT university in Denmark where "computer science" becomes an add-on to a professional education in another area.

As for the diversity of computing-related programs, John Impagliazzo mentioned that according to a survey conducted a few years ago, universities in the U.K. have more than five thousand different titles for names of computing programs; the ACM/IEEE Computing Curricula 2005 discusses only five basic models for them: Computer Science, Computer Engineering, Information Systems, Information Technology, and Software Engineering. Bud Lawson emphasized that from the viewpoint of systems engineering, traditional programs concentrate just on how to deal with computers, which is only one component in total systems.

The discussion ended with an understanding that the relatively homogeneous early views on university education in computing are transitioning by tremendous diversification. Instead of trying to place the study of computing into a well-defined place in a structured classification of university disciplines, we now need to view it as an interdisciplinary area. Such a transition would require important organizational changes that will bring specialists together from different kinds of computing-related areas and that will encourage interaction and cooperation among them.

References

[1] ACM Curriculum Committee on Computer Science, "An undergraduate program in computer science—preliminary recommendations", *Communications of the ACM*, 8(9):543-552, September 1965.

[2] ACM Curriculum Committee on Computer Science, "Curriculum'68: Recommendations for the undergraduate program in computer science", *Communications of the ACM*, 11(3):151-197, March 1968.
[3] ACM Curriculum Committee on Computer Science, "Curriculum'78: Recommendations for the undergraduate program in computer science", *Communications of the ACM*, 22(3):147-166, March 1979.
[4] ACM/IEEECS, Computing Curricula 1991, Report of the ACM/IEEE-CS Joint Curriculum Task Force, IEEE Computer Society Press [ISBN 0-8186-2220-2] and ACM Press [ISBN 0-8979-381-7], February 1991.
[5] Coates, Clarence L., et al, "An Undergraduate Computer Engineering Option for electrical Engineering", *Proceedings of the IEEE*, Vol. 59, No. 6, June 1971.
[6] Engel, Gerald, et al. "Initial Report: The Revision of Curriculum'68", *ACM SIGCSE Bulletin*, September 1974.
[7] Engel, Gerald , "The Revision of Curriculum'68: An Abstract", *ACM SIGCSE Bulletin*, July 1976.
[8] Kurki-Suonio, Reino, Birth of computer science education and research in Finland, *History of Nordic Computing*, J. Bubenko, Jr., J. Impagliazzo, A. Sølvberg (Springer), 2005, pp. 111-121.
[9] Naur, Peter, *Plan for et kursus i datalogi og datamatik.* A/S Regnecentralen, Copenhagen March 1966.
[10] Nordal, Ola, "A tool or science? The history of computing at the Norwegian University of Science and Technology", In this volume.
[11] Nykänen, Panu and Andersin, Hans, "Scientific computers at the Helsinki University of Technology during post pioneering phase", In this volume.
[12] Zadeh, Lofti A., "Computer science as a discipline", *Journal of Engineering Education*, 58(8):913-916, April 1968.