Pioneering Software in the 1960s
in Germany, The Netherlands, and Belgium

Theme

Gerard Alberts, Wilfried Brauer, Ulf Hashagen

Software, today, is the ubiquitous support of everyday practice. The early beginnings of software, however, are half a century old. From the coding of the early machines in the 1950s gradually rose the writing of software as an autonomous practice. By 1960 the main areas of research and industrial innovation were operating systems, programming languages and construction of compilers. In the following years the software field developed its own tools, techniques and methodologies. The foundation of software houses signified the emergence of software as an economic activity. The development of software showed its growing pains, from late deliveries, to not meeting specifications, and to straightforward accidents. By the end of the 1960s some perceived the field as being in a crisis, others saw new challenges. The famous Garmisch-Partenkirchen Conference in 1968 marked the self-conscious start of a new discipline called software engineering.

European countries followed their own paths in these early developments of software. Their academics tended to choose the niche of theoretical research, symbolized by abstract reflection on the design of languages. Simultanuously university research teams filled the gap of the absent software departments in the European computer industries, delivering compilers and contributing to operating systems. Germany, The Netherlands and Belgium shared this paradox characteristic of the pioneering age of software.

Topics:

- Designing programming languages
- Construction of compilers and operating systems
- The rise of the software branch: software-houses and application software
- Software tools, techniques and methodologies: towards software engineering in industry and academy

Further abstracts

Opening remarks on the history of software engineering
Paul Klint (CWI Amsterdam)

Designing Languages, Thursday afternoon

Nikolaus Joachim Lehmann from Dresden and the international ALGOL project
Hartmut Petzold (München),
In 1956 N.J. Lehmann (1921-1998) has been appointed as director of the new installed Institut für Maschinelle Rechentechnik at the Technische Hochschule Dresden, which should play the role of the most important institution for design and application of digital electronic computers in the late GDR until 1968, when it was abolished. The institute is comparable not only with the Institut für Praktische Mathematik at the TH Darmstadt, but in a certain way also with the CWI at Amsterdam, the Mathematical Laboratory at the University of Cambridge/England, the Institut für angewandte Mathematik at the ETH Zürich and others. The main differnces were: the Dresden institute was later and did his numerous activities in the isolated situation of the GDR, not accepted by the FDR at that time, and also behind the iron court dividing Europe and the world.
On the other hand the ALGOL-project has been started as an international and intercontinental project in 1957/58 and was later continued under the umbrella of the IFIP, an organisation close to the UNESCO, which included on matter of principle the socialist countries. From the very beginning the ALGOL-protagonists were convinced of their international mission which also should be independent from any company´s interests. The paper will try to give an impression of Lehmann´s situation, his decisions, and also his acting.

Van Wijngaarden and ALGOL 68
Gerard Alberts (Amsterdam),
Starting from a background in applied mechanics Adriaan van Wijngaarden (1916-1987) consciously focused on computing as a career step in the late 1940s. He extended the natural move to numerical analysis to a major step concentrating on programming automatic computers in the 1950s. Taking another leap forward in 1959 he found himself designing programming languages in the 1960s. In a

way his involvement in ALGOL 60 and working on its successor was his third career. At every next stage his work on computers was more sophisticated and more abstract.
Together with Edsger Dijkstra, Aad van Wijngaarden joined the ALGOL effort in 1959. The exciting intellectual cooperation brought them to challenging positions on the design of the language. A language definition allowing recursive procedures was a cherished feat of ALGOL 60.
When in 1962 IFIP confirmed its existence as a living organisation and adopted the ALGOL effort, Working Group 2.1 was formed to prepare a successor language to ALGOL 60. Van Wijngaarden greatly enjoyed this endeavour and repeatedly helped create the atmosphere of an intellectual pressure cooker. Launching staunch ideas, e.g. orthogonal design and two-level grammars, and surpising the working group with broad vistas he simultanuously gained the leading role in construing the new language and provoked opposition. The resulting language design ALGOL 68, was indeed this most elegant, most mathematical and most inaccessible of languages, and indeed came with a minority report derogating its achievement.
It is fair to say that ALGOL 68 was Van Wijngaardens language. Not being much of a programmer of software developer himself, he did lead the way in a part of the pioneering efforts of software, and apparently had lots of fun doing so.

Early language design and compiler development at IBM Europe; a personal retrospect
Albert Endres (Sindelfingen)
Until about 1970, programming languages and their compilers were perhaps the most active system software area. Due to its technical position at that time, IBM made significant contributions to this field. This retrospective concentrates on the two languages Algol 60 and PL/I, because with them compiler development reached an historical peak within IBM's European laboratories. As a consequence of IBM's "unbundling" decision in 1969, programming language activity within IBM's European laboratories decreased considerably, and other software activities were initiated. Much knowledge about software development was acquired from this early language activity. Some of the lessons learned are still useful today.

Comments on languages and the ALGOL effort
Helena Durnova (Brno)

Compilers, and operating systems in academic-industrial cooperation, Friday morning

Compiler construction as an academic enterprise in the ALCOR-Group
Hans Langmaack (Kiel)
The result of several years of efforts to create an international algorithmic language was ALGOL60 (ALGOrithmic Language 60) agreed upon and published in 1960 by thirteen representatives from seven countries. In order to unify even the translating programs (compilers) and to exchange experiences as well as programs several institutions joined in 1959 to form the ALCOR group (ALgol COnverteR). The different translators were based on common logical plans originating from the work of the ZMMD group (Zürich, München, Mainz, Darmstadt) which was begun in 1957. The translators differed only with respect to mode of operations and certain capacity restrictions. In a meeting in Copenhagen in February 1959 a number of European computer installators interested in ALGOL agreed upon a common five channel punched tape code, called ALCOR CCIT, deviating only slightly from the international teletype code CCIT2. In 1962 the ALCOR group consisted of six University institutions and ten computer manufacturers from Switzerland, Austria, The Netherlands, USA and Germany.
First ALGOL60-compilers of the ALCOR group became operational and were delivered to users at the end of 1961/beginning of 1962, e.g. for the Zuse Z22 (M. Paul), PERM (G. Seegmüller), ERMETH (H. R. Schwarz), MAILÜFTERL (P. Lucas, H. Bekic), ZEBRA (W. L. van der Poel, van der Mey) , Siemens 2002 (U. Hill-Samelson, H. Langmaack).
1962, when a second phase of ALCOR cooperation began where computers had got a little larger and faster, some compiler construction groups started to implement full ALGOL60 combined with machine independent optimization, especially recursive address calculation for subscripted variables in loops, an important case of strength reduction.

Progress in ALGOL 60 implementation: two successive MC-systems compared
Frans Kruseman Aretz (Eindhoven)
We compare two ALGOL 60 implementations, both developed at the Mathematical Centre at Amsterdam, forerunner of the CWI. They were designed for Electrologica hardware.
The first one, for the Electrologica X1, by Dijkstra and Zonneveld, was completed in 1960 and probably the first working ALGOL system in the world. It was designed in a purely academic

activity for a machine that did not support ALGOL 60 implementation at all. The Electrologica X8, successor of, and upwards compatible with the X1, had, on the other hand, a number of extensions chosen specifically with an eye to ALGOL 60. The system, written by Nederkoorn and Kruseman Aretz, was completed in 1965 before the first delivery of an X8. We compare the progress in hardware and software in that period of 5 years.

Comments on compiler construction
Gerhard Goos (Karlsruhe),

Operating Systems at Telefunken
Hans-R Wiehle (München)
Abstract
...

Dijkstra and the THE operating system
Edsger Dijkstra's road to a science of programming, 1951-1968
Adrienne van den Bogaard (Delft)
In this lecture Dijkstra's early work will be discussed. In 1951, Dijkstra went to Cambridge to Wilkes' first programming course as a 21 year old student in physics. This year marks the beginning of Dijkstra's career in what would be called computer science. 1968 is a remarkeable year for more than one reason: In 1968 he published his THE-multiprogramming system article which was very much read. The NATO-conference on software engineering was also held in 1968 and Dijkstra was one of the participants. What was Dijkstra's intellectual project in this period? His most important drive was the fear of errors made by computing machines. This can already be read in papers in the early 1950s. The guiding principle in Dijkstra's solutions to this problem was his attempt to abstract from the machine. In earlier years he defined that as the search for general problems in programming. Later, he would start using the phrase "Science of Programming." And in later work he started to use the phrase "Abstraction", and started to use logic and mathematics to create a Discipline of Programming.

Comments on operating systems
Klaus-Peter Löhr (Berlin)
The rise of the software industry, Friday afternoon

Volmac
Jan Mol (Amsterdam)
Abstract
...

The creation and rise of the German software industry: some remarks and case studies
Timo Leimbach (München),
This presentation will not only show how an independent German software industry was created. It will also show which major driving forces stood behind this development and how the founded firms interact successful with the development of a market, which was primary dominated by hardware firms in this time.
In a first step I want to describe Michael Porters well-known theory about the creation of new industries, before I start to apply this theory in the case of the German Software industry. Porter named two causes for the process of creating new industries: 1. general socioeconomic changes and 2. technological innovations and/or changes in the structure of costs or demand. We can find both causes at the end of the 1960s. The socioeconomic change was the Unbundling of IBM, caused by the Anti-Trust-Action in the United States. In Germany it was officially implemented in July 1972, but since the announcement in 1969 this decision marked one of the milestones for the developing software industry, not only in Germany. The second point is what some of the eye witness call the "Software Crisis". The spread of computers in commercial firms since the early 1960s, the fast development of the hardware capabilities and the creation of the high-level programming languages led directly into a rise of more and more ambitious software projects. This demand had two consequences: first a lot of these projects failed and second the hardware companies could not satisfy all of these new demands, especially concerning application software. All together this situation created a perfect ground for founding new specialized companies. In order to the model of Porter the presentation will outline all the typical characteristics for such a development like strategic and technological uncertainty, founding of spin-off or state-run interventions. Based on this I want to show

with some case studies like ADV/Orga, SAP or Softlab how independent software firms were created and how they interacted with this market environment. Due to this facts the presentation will end with some remarks on the structure of the German software industry, which emerged from this development.

Computer design and software development in Belgium before 1970; a personal retrospect
Jacques Loeckx (Köln)
In a first part of the talk the computer science activities in Belgium between 1950 and 1962 are shortly discussed: the construction of the first Belgian computer, the creation of the "Comité d'Etudes et d'Exploitation des Calculateurs Electroniques", the construction of a computer for the First National City Bank and the creation of an IRSIA-committee on electronic devices. While most of these activities were concerned with hardware problems, the research on software started in Belgium with the creation of the MBLE Research Laboratory in 1963. In the second part of the talk the different activities of this laboratory between 1963 and 1970 and the contacts with Dutch and German research groups are presented in some detail.
In both parts of the talk the prominent role played by Professor Vitold Belevitch in the development of computer science research in Belgium is highlighted.

The beginnings of the Belgian software industry
From pins & tricks to operating systems: the beginnings of modern software at the Université de Louvain, 1950s-1960s
Sandra Mols (Manchester)
In this paper, I explore the early days of programming practices in Belgium. The main part of the paper is a case study on computational and programming practices at the Université de Louvain, Louvain, Belgium, during the 1950s-1960s. During the period in question, this university was a rather outdated university as regards to computing machines, a situation that would persist until the mid 1960s. Against expectations, programming practices, and even, somewhat modern software practices would emerge from the use of the outdated computing machines in use. These machines were an Elliott E 101 (1958-1962), a NCR-Burroughs E 802 (1962- *c*1972), a low-key, by American standards, IBM 1620 (1962-*c*1970), and an IBM 360 (1966-mid-1970s). These programming practices had a usual 'material' form however: they consisted in physical manipulations of pins, rods, electronic circuits, buttons, paper tape, etc. I show how, with experience, all these practices ingrained users of these machines with intimate knowledge of their machines and led to excel at developing sophisticated such material practices for the implementation of their mathematical problems. By the early 1960s, as equipment became more sophisticated, these material skills led to the writing of assembler, local autocodes and exploration of modern software development, such as ALGOL. In the same time, one can also see a continuous importance of direct material contact with computers per se as such intimacy seems directly related to the development of expert skills at the writing of more intangible programmes and software.

Software tools, techniques and methodologies:
towards software engineering in industry and academy, Saturday morning

Project management, system management
Jan Berghuis (Bennekom)
In 1964 I transferred to Philips Computer Industry, in Apeldoorn. Before 1964 my activities had been the application of computing facilities e.g. to problems of industry, water management, spaceflight. At Philips our target was to develop and produce a computer system competing with the IBM 360 series; in a fixed time and at a certain budget. Our experience, however, was that the development of such a product mostly missed its goal in product, time and money.
At the start we had only 70 people in Apeldoorn; programming was extremely weak, we had to educate and train new people. IBM employed 2000 men to develop the 360 software. In search of a method to educate people and at the same time to learn to develop systems, a deal was struck with Computer Sciences Corporation (USA). Roy Nutt of CSC was my direct partner in this cooperation. System management was succesfully applied in developing the software for Philips P1000. Moreover in 1967 I lectured these System Management methods at the Delft University.

The Zebraclub
Willem van der Poel (Delft),
Abstract
...

Decision tables, past and present
Maurice Verhelst (Leuven)
The origin of the decision table technique dates back to November 1957 when General Electric initiated a research effort called the “Integrated Systems Project”. They also developed a software tool, called TABSOL, to convert the produced tables to a computer program. At almost the same time another American company, Sutherland, also invented the decision table.
During the next five years, the idea of a decision table caught the attention of committees of CODASYL and ACM, and as a consequence several proposals were made for automatic conversion tools to programs. Starting at the mid sixties, this lead to a number of academic articles (mostly in Communications of the ACM) in which algorithms were developed for transferring decision tables to programs in which running time and/or memory space was minimized.
In the seventies, with computers becoming faster and faster and new memory techniques evolving, the interest shifted from optimizing algorithms to sound methods for making decision tables, and the idea came up, that decision tables are not only useful in the process of programming, but also in analysis and design of information systems, and even in many other areas of human endeavor, such as law making, faultless human application of procedures, etc.
In the mid seventies, Codasyl installed a Decision Table Task Group charging it “to determine why such a valuable technique as the decision table was not in greater use, and what could be done to improve its usability.” Five years later, this emanated in the Codasyl Report “A modern appraisal of decision tables”. The task group defined the form and content of decision tables such as to be compatible with the principles of structured programming and design.
Very surprisingly, in the eighties and the nineties, the decision table was re-invented several times by professionals who did not know of its existence, mostly in the area of artificial intelligence and knowledge bases. Sometimes, this lead to a step backward, in the sense that the new tables which were presented had the same defects as the original ones of the sixties. Several examples of such “bad” tables will be given during the presentation.
During the talk, the notion of a “decision table” as it emanated from the work of the Codasyl report will be introduced, and it will be shown how and in which areas “good” decision tables are useful in various areas of human endeavor.

Software Engineering approaches before the notion
Hans-Dieter Hellige (Bremen)
As a critique of the still dominant view of the history of software engineering, which emphasizes the Conferences of Garmisch and Rome as the crucial paradigm shift in software development, the paper sketches the approaches to software engineering in the fifties and sixties:
- the software manufacturing in the military and governmental software contractor sector, which was influenced by operations research and system engineering and put the focus on a team-oriented division of labour, and
- the concept of software architecture in the corporate software sector, which combined hierarchical organization principles and structural methods with an integrated design philosophy.
Comparing the early and the actual critique of rigid top down approaches and waterfall-like process modelling, the paper pleads for a non-linear view of progress in software development history.

updated 18 October 2006