Dijkstra's Crisis: The End of Algol and Beginning of Software Engineering, 1968-72

Thomas Haigh, thaigh@computer.org, www.tomandmaria.com/tom

Draft for discussion in SOFT-EU Project Meeting, September 2010

"In SHOT meetings and in the journals, it is surprising how few and far between critical references to specific historical arguments are: there is hardly any debate or even serious substantive disagreement." – David Edgerton, 2010.

In 1968 a NATO Software Engineering Conference was held in Garmisch, Germany. Attendees represented a cross section of those involved in programming work and its management. Having never met before they were shocked to realize that identical troubles plagued many different kinds of software. Participants agreed that a "software crisis" was raging. Programming projects were chronically late and over budget, yet often failed to produce useful software. They endorsed a new discipline of software engineering, its contents yet to be defined, as the solution to this crisis. A follow up conference, held in Rome the next year, failed to reach consensus on what specific techniques might constitute the core of this new discipline. Yet software engineering soon became the dominant identity to describe the work and management of programmers.

That, at least, is the composite account one would gain from the almost identical paragraphs repeated again and again in historical works both scholarly and popular. But this growing mass of consensus is, I believe, built on sand. The historical significance of the 1968 NATO Conference, the demographics of its participants, its connection to prior and subsequent events, and its relationship to the software crisis are at risk of being fundamentally misinterpreted.

A sketch of the careers of these concepts in historical writing may help to explain the strange turn our field has taken. Within the scholarly historical literature the software crisis first appeared in Michael Mahoney's landmark 1988 paper "The History of Computing in the History of Technology." [1] In this and subsequent papers he situated the NATO conference within in the histories of both system software and theoretical computer science, an idea captured nicely in the title of "Software: The Self Programming Machine."[2] The crisis and the conference played central roles, and were treated in generally similar ways, in the only three doctoral dissertations devoted to the history of software, at least one of them well underway as Mahoney wrote.[3] In 1996 the NATO conference and attendant software crisis were enshrined in Campbell-Kelly and Aspray's standard overview of the history of computing with a lengthy

[1] Michael S Mahoney, "The History of Computing in the History of Technology", Annals of the History of Computing 10, no. 2 (April 1988):113-125.

[2] Michael S. Mahoney, "Software: The Self Programming Machine", in *From 0 to 1: An Authoritative History of Modern Computing*, ed. Atsushi Akera and Frederik Nebeker (New York: Oxford University Press, 2002):91-100.

[3] Maria Eloina Pelaez Valdez, "A Gift From Pandora's Box: The Software Crisis" (Ph.D., University of Edinburgh, 1988), Stuart S Shapiro, "Computer Software as Technology: an Examination of Technological Development" (Carnegie-Mellon, 1990), Nathan Ensmenger, "From Black Art to Industrial Discipline: The Software Crisis and the Management of Programmers" (Ph.D., University of Pennsylvania, 2001).

treatment and a claim that following the conference "Software writing started to make the transition from being a craft for a long-haired programming priesthood to become a real engineering discipline." [4] The NATO conference and the software crisis have also made their way into popular history with the recent Dreaming in Code.[5]

These later works have tended to separate the conference from the history of computer science and systems software and relate them instead to the broader evolution of programming practice.[6] The work of Nathan Ensmenger's provides the broadest claims yet for the scope of the software crisis and the impact of the 1968 conference, which he has called "perhaps the earliest and best-known attempt to rationalize the production of software development along the lines of traditional industrial manufacturing." [7] Ensmenger's expressed concern is with ordinary commercial users of computers (at that time known as data processing) and thus by implication with application programs such as payroll and accounting systems.[8] He suggests that the impetus to rationalize programming work was coming from "traditional middle-level managers" in these firms, who could use the rhetoric of the software crisis to "provide an unflattering image of the computer specialist vis-à-vis management."[9] According to this account the 1968 NATO Conference was a crucial turning point because it established software engineering as a "compelling solution to the software crisis in part because it was flexible enough to appeal to… industry managers… computer manufactures… academic computer scientists… [and] working programmers."[10]

As the purported scope of the conference and its impact grows it becomes ever harder to square with the actual historical record. Whilst the highly quotable proceedings of the NATO conferences have long called out to historians, the history of Algol has until very recently been regarded as a toxic combination of technical detail, messy failure, academic irrelevance, and bureaucratic tedium best left for participants to hash out amongst themselves. Likewise the existence of the obscure sounding "IFIP Working Group 2.3 on Programming Methodology" has rarely if ever been noted by historians. Yet my argument here may be summarized by the following five propositions:

[4] Martin Campbell-Kelly and William Aspray, *Computer: A History of the Information Machine* (New York, NY: Basic Books, 1996), 201.
[5] Scott Rosenberg, *Dreaming in Code: Two Dozen Programmers, Three Years, 4,732 Bugs, and One Quest for Transcendent Software* (New York: Random House, 2007).
[6] An important exception to this generalization is Donald MacKenzie, *Mechanizing Proof* (Cambridge, MA: MIT Press, 2001), ch. 2.
[7] Nathan Ensmenger and William Aspray, "Software as Labor Process", in *Mapping the History of Computing: Software Issues*, ed. Ulf Hashagen, Reinhard Keil-Slawik, and Arthur L. Norberg (New York: Springer-Verlag, 2002):139-165, page 153.
[8] Ensmenger, "From Black Art to Industrial Discipline", 4-5.
[9] Ibid, 161.
[10] Nathan Ensmenger, "The 'Question of Professionalism' in the Computer Fields", *IEEE Annals of the History of Computing* 23, no. 4 (October-December 2001):56-74, 42-43.

1. The organizers of the NATO conferences came from a dissident faction within the Algol project;
2. the thinking of these men was profoundly shaped by their own experiences programming compilers and other systems software;
3. no managers responsible for corporate application programming attended the conference (though there were many members of industrial research labs);
4. the IFIP Working Group 2.3 was a parallel and more successful means of pursuing the same agenda at the 1968 NATO conference;
5. and the idea of a “software crisis” entered common use not primarily from the 1968 conference itself but following the 1972 Turing Award lecture of one of those same men, Edsger Dijkstra.

Dijkstra’s crisis was rather more idiosyncratic than has generally been acknowledged. He took up the language of crisis to convey the need to replace “Chinese armies” of badly trained and mediocre programmers with an elite corps of “mathematical engineers” modeled on himself.

Algol in 1968

Let us turn first to Algol. In 1968 the Algol project, already a decade old, was the most prestigious group activity in the world of computer science and a key piece of scaffolding around which the international research community had grown.[11] The very first Turing Award, the most prestigious in computer science, was awarded to Alan J. Perlis in 1966 largely for his contributions to the Algol 60 specification. In the decades to come six[12] other members of IFIP Working Group 2.1, a committee founded in 1962 to oversee Algol, were to win the same award: John McCarthy, Edsger Dijkstra, John Backus, C.A.R. Hoare, Niklaus Wirth, and Peter Naur.

Research on programming languages was a topic that grew particularly naturally out of practice. Systems programmers were a fixture of every major computer installation in the late 1950s and early 1960s. Installations developed their own software tools, code libraries, and even operating systems or adapted externally produced code to meet local needs. Systems programming attracted virtuoso programmers, whose interests tended toward to computer itself rather than to its users or applications. The instinct of the systems programmer, when faced a set of tasks, is to look for patterns that reveal an underlying class of problem amenable to a generalized solution. Compilers for general purpose languages are among the most flexible software tools, but in those days any reasonably flexible tool for a task such as report generation, computer aided design, or textual processing tended to include a specialized programming language in place of what we would now think of as a user interface. For the most part, applying a computer to a particular problem meant writing a program in one of an ever growing host of programming languages. And of course computer manufacturers felt increasing pressure to deliver each new machine with a full suite of operating system utilities and compilers, leading them to expand their own systems programming teams. Particularly among those with personal backgrounds in science, or service jobs in universities, this could easily suggest new lines of research. So

[11] The early history of Algol has recently been discussed in David Nofre, "Unravelling Algol: US, Europe, and the Creation of a Programming Language", *IEEE Annals of the History of Computing* 32, no. 2 (April-June 2010):58-68. Its relationship to the development of computer science is explored in H T de Beer, "The History of the Algol Effort" (Technische Universiteit Eindhoven, 2006).
[12] Floyd is also listed in Wikipedia as a member, but I’ve been unable so far to verify this from more reliable sources, though he did create an early and influential Algol 60 compiler.

by 1958 there were many paths toward a research interest in programming languages and compilers. In language remarkably similar to that commonly applied to the 1968 NATO Conference, Alan Perlis described an ACM committee convened in 1958, and soon merged into the Algol effort, as including "representatives of the computer industry, users, universities, and the federal government."[13]

But the path from a research interest in programming languages and compilers to a research position in a corporate lab or computer science department was far from obvious. When the Algol project was launched in 1958 there was no such thing as an academic discipline of computer science. The term itself had been coined, but as one of many possible identities it was not yet clearly distinct from alternatives such as "information processing sciences" (an alternative reflected in the name of IFIP, the International Federation for Information Processing Societies).[14] In the United States, home to most of the world's computers, there were several professional associations and groups active in the nascent field but no university departments or degree programs devoted exclusively to teaching and research in computing. Of course computing research was carried out, but it took place at the margins of other endeavors. Computers were used in corporate engineering groups, government research institutes, campus computer centers, administrative departments, and other settings. The people publishing computing papers at this point might be working as physicists, mathematicians, systems programmers, electrical engineers, or consultants. Their interest in progressively more abstract and theoretical aspects of computing still had to be justified in terms of practical benefits or the interests of better established disciplines.

C.A.R. Hoare recalled that when he started work on his Algol 60 compiler, "we didn't correspond with other people writing compilers, even for Algol, in those days… There was no real scientific community which one could join to talk over problems with other people who encountered the same problems. No, I think we worked pretty well on our own." [15] Of course the Algol project itself had begun a few years earlier to provide exactly that kind of expert community on an international basis, and Hoare himself was soon recognized as an important figure within it. For men such as Hoare, Dijkstra, and Brian Randell participation in the project's meetings, initially as compiler implementers rather than language designers, was a crucial step in their establishment as major figures in the nascent discipline of computer science and their personal transition from systems programmers to software researchers.

Yet despite its storied past and elite membership the project was in trouble. After the early triumph of Algol 60, arguably the most influential programming language in history, many suggestions were made on its extension and improvement. IFIP WG 2.1 was chartered to oversee the maintenance and development of Algol, formalizing the membership and responsibilities of the core project team. By the mid-1960s they had turned toward design of a successor to Algol 60. Things progressed more slowly and less agreeably than expected. According to Hoare's famous account, given years later in his Turing Award lecture, the group took a decisive wrong turn in 1965 when it inexplicably rejected an elegant

[13] Alan J Perlis, "The American Side of the Development of Algol", in *History of Programming Languages*, ed. Richard L Wexelblat (New York: Academic Press, 1981):75-91, page 77.
[14] See, for example, Anonymous, "Is It Overhaul Or Trade-In Time (part I)", *Datamation* 5, no. 4 (July-August 1959):24-33.
[15] C A R Hoare, *Oral History by Jonathan P. Bowen* (Computer History Museum, 2006).

and pragmatic draft design compiled by Niklaus Wirth in favor of an "incomplete and incomprehensible" alternative design for a radically different language submitted by Adriaan van Wijngaarden.[16]

Years of work to complete and discipline this new design led to a series of three meetings in 1968 in which the new language was to be finalized and approved. van Wijngaarden did in the end win approval from the working group, and its parent organization, for his approach to what became known as Algol 68. But he could not persuade many key members of the group to endorse the final proposal, including Hoare, who believed that the committee had been "packed with supporters" of the language (including several of van Wijngaarden's students).[17] Wirth and Naur, one of the leaders of the Algol 60 effort, resigned from the working group shortly after a May 1968 meeting in Zurich where a late draft of the report was discussed.[18] In December Algol 68 was approved by the working group at a meeting in Munich, but several members of the committee had written draft minority reports and were passing them around during the meeting in an attempt to enlist allies. Despite the addition of language in the cover letter making clear that not all of the group members supported its final product seven of its members still signed a minority reported drafted by Dijkstra.

Reading this minority report one is struck by its tight connection to the agenda of the NATO conference. Algol 68 has been widely criticized as overly ambitious, crammed with unnecessary features, and described in an inscrutable notation. A revised report issued in 1974 tided things up somewhat, but the language was rarely implemented and found only niche applications. (Some of its novel features were, however, adopted by more successful languages). In contrast Wirth's rejected earlier draft formed the basis of Pascal, one of the most widely used programming languages of the 1980s. Thus one might expect that the dissident faction would have no shortage of detailed and well informed criticisms to make of its baroque excesses. Indeed such criticisms could readily be found on the pages of the Algol Bulletin during this era.

Yet the minority report fills just one typed page. It brands both Algol 68 and its new language definition method as failures. The paragraph on the language itself asserts that the new complexity and power of third generation computers confronts the programmer with "tasks of completely different and still growing scale and scope" from those of a decade earlier. What is needed, it asserts, is not just a language but an "adequate programming tool that… assists, by structure, the programmer in the <u>reliable creation</u> of sophisticated programs." Because Algol 68 had not incorporated a new "implicit view of the programmer's task" it followed that "regarded as a programming tool, the language must be regarded as obsolete." [19]

The problem identified by the report is not one badly designed programming language but rather the assumption that programming languages should be judged primarily on their ability to elegantly and concisely express well understood meanings. What Dijkstra seems to have wanted instead was a new understanding of languages as tools designed to support or enforce good practice in the design of complex programs. As another participant later wrote, "the whole agenda had changed. The true

[16] C A R Hoare, "The Emperor's Old Clothes", *Communications of the ACM* 24, no. 2 (February 1981):75-83.
[17] Ibid.
[18] C H Lindsey, "A History of Algol 68", in *History of Programming Languages II*, ed. Thomas J. Bergin and Rick G Gibson (New York: ACM Press, 1996):27-83, page 37.
[19] Edsger Wybe Dijkstra et al., "Minority Report", *Algol Bulletin*, March 1970.

problem, as he now saw it, was the reliable creation of programs to perform specified tasks, rather than their expression in some language. I doubt if any programming language, as the term was (and still is) understood, would have satisfied him."[20]

Algol and the NATO Conference

A closer look at the calendar may help to explain this shift in interests, commitments, and expectations on the part of the Algol dissidents. The 1968 NATO Conference on Software Engineering took place in October just as tensions within the Algol project peaked. Wirth and Naur had walked out five months earlier. That very month Van Wijngararden circulated a final draft of the official report, and Doug Ross, Dijkstra, Randell and Hoare must have already been at work on their various draft minority reports.[21] So as they signed Dijkstra's minority report the Algol 68 dissidents were still enthralled with the potential of the new agenda they had explored at the first NATO conference a few months earlier.

A glance at the lists of attendees and organizers of the 1968 NATO conference demonstrates a surprising degree of overlap with the roster of Algol veterans. Friedrich L. Bauer chaired the conference, providing it with its name and obtaining the necessary funding from NATO. He was one of the original founders of the Algol effort, or as it has sometimes been called in this context his "Algol conspiracy." Nine individuals are identified as "group leaders" for design, production, and service. All of these men had well defined connections to the Algol project either as designers of the language, implementers of important Algol compilers, or in the case of J. N. Buxton as creator of a language based on Algol concepts. Of twenty-eight members of IFIP WG 2.1 identified as "active in the original design of Algol 68" thirteen attended one or both of the NATO software engineering conferences. Looked at the other way, of the fifty three full participants at the 1968 meeting (excluding NATO observers and scientific secretaries) twenty eight had made direct contributions to the Algol project as committee members, authors of papers with suggested improvements to Algol, implementers of Algol compilers, or designers of languages explicitly inspired by Algol.[22]

The role of the Algol dissident faction was particularly important. Eleven working group members signed the minority report, resigned in protest earlier in 1968, or circulated their own denunciations of Algol 68 (in the case of Ross). Eight of these men attended at least one of the NATO conferences, and another was Polish and so unlikely to receive an invitation from NATO. On the other hand Van Wijngaarden was conspicuous by his absence in Garmisch and Rome. Only two of the ten other people identified as members of his "inner circle" during the completion of Algol 68 attended either meeting. [23]

As editors of its final report Naur and Randell, two of the Algol 68 refugees, played a particularly important role in shaping the meeting's impact and legacy. Their report is neither a conventional collection of papers presented or an actual transcript of the discussion. Portions of it read like a transcript, but in fact it is more of a Platonic dialog constructed with fragments of the precirculated

[20] Lindsey, "A History of Algol 68", page 40.
[21] D T Ross, "Concerning a Minority Report on Algol 68 (AB 30.2.2)", *Algol Bulletin*, February 1969 shows Ross's correspondence in September and October 1968 with Randell and Hoare trying to get a substantive minority report together.
[22] Between the two conferences these included CPL, JOVIAL, ANALGOL, Algol W, CLISP, Triplex-Algol 60 and MAD.
[23] Lindsey, "A History of Algol 68", page 41.

position papers, at least one document written during the conference, and précis capsules from the verbal discussion. In most cases these summaries were made without reference to the recording. According to the introduction "owing to the high quality of the reporters' notes it was, in general, possible to avoid extensive amounts of tape transcription."[24] A handful of the working papers are reproduced in full, or at least at greater length, in an appendix. So the report as we have it is already a highly processed, well structured document edited to highlight particular topics and publicize certain debates. Participants had even been promised that the original tapes would be destroyed once the report was complete.[25] Doug McIllroy is said to have called the final report ""a triumph of misapplied quotation."[26]

What did the Algol dissidents hope to accomplish with this new initiative to create something called software engineering? I suspect that their intent was to leave behind Algol itself while preserving the core community they had built up during a decade of intensive collaboration and propagating elements of the Algol project they had come to value. These included its international character, the intellectual caliber of its participants, and the diversity of their backgrounds and interests. A lot had changed in the world of computing research in the decade between the initial meetings behind Algol 58 and the crisis now afflicting the group. By 1968 there was such a thing as computer science, at least in the United States. Undergraduate and graduate degree programs were increasingly common and many of these were housed in autonomous university departments at respected universities such as Stanford, Michigan, Pennsylvania, Purdue, and Carnegie Mellon. Academic politics sent computer scientists in search of a field of theory it could call its own, justifying its status as a distinct scientific discipline. 1968 was itself a landmark year. The first Ph.D. in computer science was awarded (to a programming language specialist, naturally) and the Association for Computing Machinery issued its influential Curriculum 68 guidelines for a model undergraduate degree. The ACM had around 24,000 members, and to manage its increasing size and diversity had recently introduced a system of special interest groups which were beginning to run their own conferences and publish their own journals.[27] The system of interest groups had been approved in 1961, and they eventually came to function in many ways as societies in their own right, splintering the identities of computer science as the field grew.[28]As Perlis said the next year at the second NATO conference, "all of us who are in universities at least have a pretty good understanding of the nature of a computer science doctorate program: a healthy dose of logic, automata and computing theory, a diminishing dose of numerical analysis, one or two courses in advanced programming of one

---

[24] Peter Naur and Brian Randell, eds., *Software Engineering: Report on a Conference Sponsored by the NATO Science Committee, Garmisch, Germany, 7th to 11th October 1968* (Brussels: Science Affairs Division, NATO, 1969), 6.

[25] Edsger Wybe Dijkstra, *EWD246: Verslag van het bezoek aan de NATO Conference on Software Engineering; translation by Karel van Oudheusden* (1968); available from http://userweb.cs.utexas.edu/users/EWD/ewd02xx/EWD246.PDF.

[26] Brian Randell, *The 1968/69 NATO Software Engineering Reports, from unpublished proceedings of Dagstuhl-Seminar 9635: "History of Software Engineering," August 26 - 30 1996* (1996); available from http://www.cs.ncl.ac.uk/people/brian.randell/home.formal/NATO/NATOReports/index.html.

[27] Donn B. Parker, "Association for Computing Machinery Promotion of Professional Development", in *Data Processing XIII: Proceedings of the 1968 International Data Processing Conference and Business Exposition*, ed. Data Processing Management Association (Chicago: Data Processing Management Association, 1968):333-335 is the source of the numbers here.

[28] The creation of the ACM SIG system is explored in Walter Carlson, "ACM and Special Interest Groups", *Data Base* 25, no. 2 1994):9-12.

sort or another, some artificial intelligence, and some of this and some of that."[29] The computing research community was less intimate than it had been a decade before and specialists were retreating into their own narrow worlds.

Within this changing world Algol community remained much as before: a tight knit, international community balancing an increasing interest in theory with hands on experience of systems programming and compiler implementation. Dijkstra retained throughout his career a pride in having been "a systems person" in a context where theorists increasingly looked down on the development of actual systems.[30] Like Hoare, Wirth, and many of the other Algol dissidents he maintained this pride even as their work became increasingly mathematical.

This faith in the fruitful interplay of theory and practice in the improvement of systems programming work was at the heart of the agenda behind the NATO Software Engineering conferences. The experiences of their leaders help us to understand this. After all, specifying language syntax and semantics and creating efficient compilers had been a huge challenge facing the original Algol team from 1958 onward, but the challenge had been overcome yielding new theoretical understandings and new compiler techniques as well as a programming language more elegant than any of its predecessors. By the 1970s standard tools such as LEX and YACC, combined with a much improved theoretical framework based around the Chomsky hierarchy of formal grammars, had turned the implementation of a new language into a relatively trivial task. Knowledge of this kind also helped Wirth to design languages to be efficiently implementable and more amenable to mathematical verification.

The interests of the Algol dissidents were shifting from programming language design to the broader challenges of developing large scale systems software. They hoped to apply similar tools in this broader setting. For example, Dijkstra's interests extended far beyond programming languages, and indeed his work on the principles of multitasking operating system design had already yielded the fundamental concept of semaphors (a means of preventing systems from hanging when multiple processes needed to access common resources). But his interests were still more general. In 1968 Dijkstra was already working on the approach to program modularization he called Structured Programming – his major essay on the subject was circulated the next year and his letter denunciation of GOTO had already appeared in Communications of the ACM.[31]

Dijkstra has written that "In 1968 I suffered from a deep depression" for which his work toward "Notes on Structured programming" was a kind of therapy. He recalls the 1968 NATO conference as one of two

[29] J N Buxton and B Randell, eds., *Software Engineering Techniques: Report on a Conference Sponsored by the NATO Science Committee, Rome, Italy, 27th to 31st October 1969* (Brussels: NATO Scientific Affairs Division, 1970), 60.

[30] Edsger Wybe Dijkstra, *Are 'Systems People' Really Necessary: EWD 1095* (1991); available from http://userweb.cs.utexas.edu/users/EWD/transcriptions/EWD10xx/EWD1095.html.

[31] "Notes on Structured Programming" was circulated as EWD249 in 1969, a Technical University Eindhoven technical report report in 1970, and given definitive publication in O J Dahl, E W Dijkstra, and C A R Hoare, *Structured Programming* (New York: Academic Press, 1972). A much shorter "Structured Programming" was precirculated for the Rome conference and included in the proceedings (pages 84-88). Dijkstra's brief dismissal of the GO TO was rushed into print by Wirth, then editing the CACM letters department. Edsger W Dijkstra, "Letters to the Editor: Go To Statement Considered Harmful", *Communications of the ACM* 11, no. 3 (March 1968):147-148.

"external stimuli" toward this recovery (the other being the IFIP WG 2.3 initiative discussed later).[32] He credited the conference with an enlightenment both literal and metaphorical: "For me it was the end of the Middle Ages. It was very sunny."[33] From someone recalling the darkness of depression the metaphor is striking.

What Was "Software" Anyway?

Today we tend to use "software" synonymously with "computer program." This was not the case in the early 1960s, as I have explored in my article "Software in the 1960s as Concept, Service, and Product."[34] Software first became part of the data processing lexicon in the early 1960s, as the complement of hardware. As a 1962 Honeywell advertising supplement entitled "A Few Quick Facts on Software" explained

> Software is a new and important addition to the jargon of computer users and builders. It refers to the automatic programming aids that simplify the task of telling the computer 'hardware' how to do its job. …. Generally there are three basic categories of software: 1) Assembly Systems, 2) Compiler Systems, and 3) Operating Systems.[35]

These three would later be called *systems* software, but at this time no qualification was needed. In 1968 the word software remained closely associated with systems programs, which were increasingly likely to be supplied to computer users by computer manufacturers or specialist firms, rather than with applications programs, which were still overwhelmingly likely to be created within the company using them. Some definitions suggested than a computer service or program was software if purchased from an external company but was not software if developed in house. As Gerard Alberts has pointed out, this highlights the "ware" side of software: its role in commercial exchange.[36]

It should not therefore be surprising that in the proceedings of the NATO software engineering conferences "software" is usually used to imply systems software. Application programs were dismissed in passing by J.N. Buxton, then a professor at the University of Warwick, as outside the scope of discussion: "Of course 99 percent of computers work tolerably satisfactorily; that is obvious. There are thousands of respectable Fortran-oriented installations using many different machines and lots of good data processing applications running quite steadily; we all know that!"[37] The SAGE air defense system, today recognized as the most influential single system in the history of computing, is not mentioned at all. The SABRE airline reservation system, another staple of historical accounts, comes up just once.

[32] Edsger Wybe Dijkstra, *EWD1308: What Led to 'Notes on Structured Programming'* (2001); available from http://userweb.cs.utexas.edu/users/EWD/transcriptions/EWD13xx/EWD1308.html.
[33]Quoted P. 36 of MacKenzie book, but check original in Valdez dissertation.
[34] Thomas Haigh, "Software in the 1960s as Concept, Service, and Product", *IEEE Annals of the History of Computing* 24, no. 1 (January-March 2002):5-13.
[35] Honeywell, "A Few Quick Facts on Software", *Business Automation* 7, no. 1 (January 1962):16-17
[36] Gerard will presumably say this somewhere else in the volume, for a cross reference.
[37] Naur and Randell, eds., *Software Engineering* .

What specific pieces of software are mentioned in the 1968 NATO conference proceedings? The top five are OS/360 (referenced in 16 different paragraphs), Algol (10 paragraphs), PL/1 (7 paragraphs), TSS/360 (6 paragraphs) and Multics (6 pargraphs). A full tabulation[38] would include

- **Operating Systems**: OS/360, TSS/360, CTSS (MIT), Multics, TSS/360, TSS/635 (the OS for an online military system), ECS Scope (CDC) and a range of experimental systems such as Dijkstra's THE, MTS (Michigan), JOSS, CODAS by Parnas and Darringer, and the work of Zucher and Rangall at the IBM Watson Lab.
- **Programming Languages/Compilers:** PL/1, AED (an Algol-based MIT string oriented language), FORTRAN, COBOL, Nebula (for ICT Orion), LISP, SIMULA & SNOBOL.
- **Tools to Support Programming**: Autoflow (ADR), Com Chart, Mercury, TOOL, etc.
- **Real Time Control Applications**: ESS (Bell's Electronic Switching Systems), SABRE.

Discussion was dominated by problems facing the large scale development of operating systems and programming languages. As Wirth noted during the 1969 conference, "It might be thought quite remarkable what proportion of the submitted working material was on the subject of programming language development, although the conference is on software engineering."[39]

Who Was At the Meeting?

In a fairly typical description, Donald MacKenzie wrote that the invitee list for the 1968 NATO Conference was "carefully constructed to include key figures in academia, in the computer industry, in the emerging 'software houses,' as well as a small number of important computer users." [40] Other historians have used similar phrases to suggest that the impact of the conference stemmed in part from its inclusiveness. MacKenzie's description is not incorrect, but it is easy to read into it a much more diverse group of people than the one actually assembled in Garmisch.

It would be a caricature to claim that the conference was nothing more than another meeting of the Algol team. Writing immediately after the meeting, Dijkstra noted that "many met here for the first time… 24 of the 55 attendees I had never seen before…. This made the conference exciting because we were all becoming acquainted with new worlds of experiences." He called this "a kind of antidote against the incest" of IFIP. But the vast majority of full participants were best known for their work on operating systems and compilers. It seems that the organizers developed their list of invitees for the two NATO conferences primarily by broadening out from the core Algol community to include colleagues known for their work on other languages and compilers (including several experts each for APL, Fortran, and PL/1), a few experts on programming language theory and compiler compilers, and a smattering of operating system specialists.

[38] Hmm – what is GP? SYSGEN? IPL? Check for others and make sure are properly situated.
[39] Buxton and Randell, eds., *Software Engineering Techniques*, 18.
[40] Donald MacKenzie, "A View from the Sonnenbichl: On the Historical Sociology of Software and System Dependability", in *Mapping the History of Computing: Software Issues*, ed. Ulf Hashagen, Reinhard Keil-Slawik, and Arthur L. Norberg (New York: Springer-Verlag, 2002):97-122, page 98.

Both of the NATO conference reports include the work address of each participant. This permits an approximate tabulation of the groups represented at the meeting. James E. Tomayko's assertion that "Software engineering seemed of immediate interest to industry, and industry dominated the population at the NATO conference" does not hold up well.[41]

Sector	**1968**	**1969**
Universities	24	26
Corporate Research Labs	10	12
Computer Manufacturer (excluding research centers)	10	11
Computer Software & Services Firms (including timesharing services & online information services)	5	7
Govt. Research Centers or Ministries	3[42]	2
Industrial Company (excluding research centers & computer manufacturers)	0	1
No affiliation given	0	2
Observers, secretaries, etc.	7	21
Total	60	83

There were attendees from many computer companies, from many universities, and from a handful of software companies. But most of the attendees from each side were members of a single emerging community: computer scientists researching systems software. The actual picture is rather less diverse than one might expect. For example IBM was well represented at the conference. Brian Randell and Ascher Opler worked at its Yorktown Heights research center, Albert Endres at its Böblingen lab in Germany, C.P. Enlart at its European Program Library in France, Francois Genuys ran programming language research at IBM France, and John Nash worked on PL/1 at IBM's UK labs in Hursley. [43] Of its

[41] Tomayko even wondered how history might have been different it "more than a sprinkling of academics had attended the Garmisch conference?" James E Tomayko, "Forging a Discipline: An Outline History of Software Engineering", *Annals of Software Engineering* 6 1998):3-18, page quote from pages 6 and 7.

[42] This includes both the Ministry of Technology in London and the "Central Institute for Industrial Research" in Olso – it's not immediately apparent if the latter is academic or government run.

[43] This, of course, raises the problematic separation of "research" and "development," as development of systems software intended for customer use might be done by research groups during this era. In the 1960s flagship corporate research laboratories, of which IBM's Thomas J. Watson research center in Yorktown Heights was a prime example, modeled themselves on elite university research groups and boasted of supporting basic science across a range of disciplines such as physics and economics. However some research centers, including IBM's Hursley and Böblingen labs, had primary responsibility for developing mainstream software tools shipped as official supported software to customers. This is nicely documented in Albert Endres, "IBM Boeblingen's Early Software Contributions", *IEEE Annals of the History of Computing* 26, no. 3 (Jul-Sep 2004):31-41, which shows the

sizable delegation only R. C. Hastings from IBM Rochester, New York came from a mainstream IBM group rather than one of its research centers, and his career left no mark on Google other than his dialog in the conference report.

Many of the industrial participants were computing researchers rather than developers of systems intended for direct commercial use. Almost all were following career paths leading further away from large scale software development and its management. Randell's career path captures a common progression among the attendees. In the late 1950s and early 1960s he worked for English Electric to develop systems software, most notably a much acclaimed Algol compiler. In the mid-1960s he transitioned to a corporate research center at IBM, where he worked on experimental computer architectures and methodologies for operating system design. In 1971 he left business altogether, to become a professor of computer science at the University of Newcastle. Hoare's career was on a similar arc. His Algol compiler was created within a commercial firm and by 1965 he had risen manage a team of around fifteen system software developers. By 1968 he had left for an academic career. Dijkstra followed a similar path, though starting from an academic rather than industrial systems programming group. He was the coauthor of the first Algol 60 compiler, did early work on interrupt handling, and made a major contribution to operating system design with his experimental THE system of the mid-1960s. His career turned progressively toward mathematics and theory, and though his work on concurrent programming and process synchronization was hugely influential he never took part in the development of a large scale commercial system.

IBM's OS/360 was, as we have seen, by far the most frequently discussed software at the conference. Yet none of the conference participants appear to have worked on any aspect of it other than PL/1. Fred Brooks was soon to publish The Mythical Man Month, perhaps the most widely read book ever published on software engineering, based on his experience of managing the project from 1961 to 1965. He was not at the conference, and neither were any of the seven other men he identifies as "key players" in its "conceptual structure."[44] Also absent was his successor, Watts Humphrey, who has become equally famous within software engineering as the creator of the Capability Maturity Model. Neither, to the best of my knowledge, had any of the researchers who attended made direct contributions to IBM's other major software systems of the late 1960s and early 1970s such as CICS and IMS. Another operating system, Multics, was also rather troubled and much discussed. It was a joint industrial/academic project, and two conference participants did have direct involvement despite their location in "research" rather than "development" institutions: R. M. Graham from MIT's Project MAC and Doug McIllroy who was then head of the Bell Labs Computing Techniques Research Department.[45]

Neither were there at the 1968 conference current leaders of major software efforts from Univac, CDC, DEC or any of the other major US companies apart from General Electric. The latter sent Robert W.

---

lab built the main system software for the System/360 Model 20 and implemented PL/1 compiles for low end 360 systems. Yet at a time when significant development work was still being carried out in universities even this does not necessarily separate the interests and activities of industrial researchers from their academic counterparts.

[44] Manfred Broy and Ernst Denert, eds., *Software Pioneers: Contributions to Software Engineering* (Berlin: Springer-Verlag, 2002), 178.

[45] Graham was responsible for many aspects of the Multics Kernel and McIllroy was, like many of those who went on to create Unix, a part of the Multics team until Bell Labs' final withdrawal from the project in 1969. Another Bell labs participant, E.E. David, was a senior manager at the time.

Bemer, who managed system software efforts for Univac in the early 1960s. But Bemer was hardly a typical corporate manager. He had been an implementer of Algol 58 and maintained an interest in its progress, writing a lengthy "Politico-Social History of Algol." Prior to the conference had been shifting his interests strongly toward researching and writing about software production and was soon to give up his managerial title for a consulting role.[46] In contrast European computer manufacturers, perhaps because of their relative smallness and lack of dedicated research divisions, sent several men who were then managers responsible for system software development: J. Berghuis of Philips (who is nevertheless identified as Prof. Dr. in the proceedings and so must also have had some kind of academic identity), Helmut Köhler of AEG Telefunken, Gérard Letellier of SEMA, and F. Sallé of CII.

Dijkstra had been complaining for years about the inadequacy of commercial hardware and software, particularly that produced by IBM. Immediately after the conference he wrote that it was "refreshing to experience" acknowledgement of "software failure" from almost all the participants. According to Dijkstra, "the people from industry were extremely happy that they were at least allowed to express their thoughts! … Some very hard nuts have been cracked open concerning the role of the computer manufacturer in all this, and IBM in particular."[47] This apparent cultural breakthrough becomes easier to understand when we appreciate that Dijkstra had not been hearing from IBM's product managers or marketing experts but from members of its research staff. They shared the values and culture of the academic participants, and the two sides could bond by denouncing the ignorance and incompetence of those producing and selling products for a living. Dijkstra was disillusioned at Rome the next year, describing the conference as "a five-day torture" because "this year it had to be 'positive'" and some participants suggested that "actually it is good enough.'"[48]Shaken by what he saw as a return to complacency, he warned "We may continue to think that programming is not essentially difficult, that it can be done by accurate morons, provided you have enough of them, but then we continue to fool ourselves and no one can do so for a long time unpunished."[49]

The received picture of the 1968 conference as a broadly inclusive gathering of all those with a stake in programming work crumbles further when we consider which groups were not represented there. There were just four people at the 1968 conference from independent companies involved in the production of software or other kinds of computer services. All four firms began as systems software specialists: Alex d'Agapayeff was leader of Computer Analysts and Programmers, a growing software company whose first job had been creation of an Algol compiler; J.D. Babcock ran a small company around a timesharing operating system and interactive PL/1 variant he had developed jointly with IBM a few

[46] Bemer writes "At first I was assigned to work for Dr. John Weil, who was a bit suspicious of me then. His managers always thought I was going to be the manager for all GE software when someone got around to fixing the organization…. I was usually a special assistant to the General Manager in Phoenix." Bob Bemer, *Bob Bemer returns from Paris to Phoenix -- Still with General Electric* (bobbemer.com, [cited 26-APR-01 2001]); available from http://www.bobbemer.com/GEPHX.HTM. Bemer also reports that it was Weil who had been invited to attend the conference, but that he was accepted as a substitute.
[47] Dijkstra, *EWD246: Verslag van het bezoek aan de NATO Conference on Software Engineering; translation by Karel van Oudheusden* .
[48] Edsger Wybe Dijkstra, *Verslag van de tweede "Conference on Software Engineering", georganiseerd door de NATO Science Committee te Rome, 27–31 oktober 1969; translation by Karel van Oudheusden* (1969); available from http://userweb.cs.utexas.edu/users/EWD/transcriptions/EWD02xx/EWD272.html.
[49] Edsger Wybe Dijkstra, *EWD273: The Programming Task Considered as an Intellectual Challenge* (1969); available from http://userweb.cs.utexas.edu/users/EWD/transcriptions/EWD02xx/EWD273.html.

years earlier; Hollis A. Kinslow had recently left behind a career in timesharing research at IBM's Yorktown Heights to run a tiny consulting firm specializing in government work; and Ken Kolence was a founder of Boole and Babbage, then a specialist in performance monitoring utilities.

Major American software houses, such as Informatics or Applied Data Research, were not represented at all. Neither were banks, insurance companies, manufacturing firms or any of the other end user companies in which the vast majority of the world's programmers worked. Few of the attendees had significant experience with applications software other than early experience with scientific computation. There were no specialists in data base management, just then emerging as a crucial kind of systems software and the most important market niche for packaged software companies. Neither were there any specialists on real time applications, such as airline reservation systems. Among the high profile industry figures who might have represented these areas were Charles W. Bachman, Bob Patrick, Dick Canning, Robert V. Head, Frances V. Wagner, Walter M. Carlson and Grace Hopper.

IFIP WG 2.3

Approving the draft design of Algol 68 was not the only thing decided by IFIP's embattled Working Group 2.1 when it met in Munich a few months after the NATO Conference on Software Engineering. The group also voted to establish a new Working Group to pursue formal approaches to software engineering. The new body soon gained official recognition as IFIP WG 2.3 on Programming Methodology. Its members felt that "programmers needed tools other than bigger and better programming languages."[50]According to Randell, "we regarded this group, WG 2.3, as having twin roots, the Algol Committee and the NATO Software Engineering Conference."[51]

Its founding meeting took place in Oslo in July 1969, a few months before the second and final NATO Conference on Software Engineering. Eight men attended this meeting, and one more is also recorded as a founder member. They were O.-J. Dahl, Dijkstra, M.D. McIlroy, Randell, Ross, Gerard Seegmueller, W.M. Turski, Mike Woodger and Manfred Paul. Of these nine, all but one had until recently been members of IFIP WG 2.1 actively working on Algol 68. The long exception was the host, Dahl who was eventually to win the Turing Award for his development of SIMULA-67, a programming language based on Algol 60. Seven of the men attended one or another of the NATO Conferences (and one of the others, Turski, was Polish). Most strikingly of all, six of the nine founders had signed the Algol 68 minority report (or, in the case of Ross, attempted to produce their own). Four more Algol 68 dissidents soon joined the group: Wirth, Hoare, Naur, and Duncan. So in the end only one of the signatories of the minority report was left outside the new working group.[52]

By 1970 the IFIP WG 2.1 responsible for Algol 68 had effectively splintered into two. Twenty eight members of WG 2.1 had contributed to Algol 68. Only thirteen of these remained active in the effort to create a revised version of the specification. New, and significantly less distinguished, members were

[50] M Woodger, "A History of WG2.3", in *Programming Methodology: A Collection of Articles by Members of IFIP WG2.3*, ed. David Gries (New York: Springer-Verlag, 1978):1-6, page quote on page 1.
[51] Brian Randell, "Edsger Dijkstra", in *Ninth IEEE International Workshop on Object-Oriented Real-Time Dependable Systems (WORDS'03F)* (IEEE Computer Society, 2003).
[52] Jan Garwick of Oak Ridge National Lab. As the first nine meetings of WG2.3 were held in Europe and IFIP had no travel funds it is possible that Garwick was invited to join but could not take part for financial reasons.

brought on board to replace them. None of the members of the new WG 2.3 to participate in this effort.[53]

The Algol 68 dissidents had resented the pressure applied to WG 2.1, a large and diverse committee, to agree a draft language design on a tight schedule and make whatever political compromises were necessary to accomplish this. At his final meeting before resignation, Peter Naur had given a paper on "Successes and Failures of the ALGOL Effort" in which he blamed the problems afflicting the Algol project primarily on the IFIP as "the true villain of this unreasonable situation." Naur bemoaned the "extreme clumsiness" of its incorporation as an IFIP working group, its "heavy organizational structure and pretentiousness," and the "hunt for the ultimate monument on which to stick the name Algol."[54] They were also skeptical of the institutional agenda behind the 1969 Rome conference, which according to Randell led to its failure: "it became clear during the conference that the organizers had a hidden agenda, namely that of persuading NATO to fund the setting up of an International Software Engineering Institute. However things did not go according to their plan. The discussion sessions which were meant to provide evidence of strong and extensive support for this proposal were instead marked by considerable skepticism…"[55]

The new group swung to the opposite extreme. According to Mike Woodger, its founding chairman, the group "avoids paper work, reports, voting and the like, and has no explicit 'product.'… Minutes of meetings are not ordinarily kept…. Having too many people at a meeting delays interaction, preventing a proper exchange of views… Having a rigid timetable inhibits the airing of newly produced or developing points of view. "[56] Meetings lasted for five days and were by invitation only. Woodger hoped that the group's success would be judged by "the normal scientific publications of the members."

Working Group 2.3 was something like an elite private member's club, in which conversations could be held frankly and at a high technical level. Membership was a considerable honor, and only a few members were invited from each country. This exclusivity was particularly appreciated by Dijkstra, who had finally found a venue largely free from mediocrity, American dominance, and the dread influence of commercial pressures. In 1976 he was traumatized when several outsiders were invited to join a meeting of the group at St.Pierre-de-Chartreuse. The guests were "insufficiently secure to expose their uncertainty" with honest discussion, and "the meeting was dangerously beginning to look like an ordinary conference with unreferreed papers." On the final morning, when the guests had left, members realized that "the meeting had largely been wasted, and that all of us were totally miserable

[53] Manfred Paul chaired WG 2.1 in the early 1970s and was present at the founding meeting of WG 2.3. However he is not otherwise identified as a member of WG 2.3 and seems not to have attended any further meetings. He may have been present in Oslo as a representative of WG 2.1.
[54] Peter Naur, "Successes and Failures of the Algol Effort", *Algol Bulletin*, July 1968.
[55] Randell, *The 1968/69 NATO Software Engineering Reports* . Who were these "organizers?" The chairman of the Rome conference was P. Ercoli, a prominent figure in Italian computer science. He had few publications and does not seem to have been a specialist in systems software or programming languages, unlike most other attendees. Bauer was "co chairman". Italy was a frequent recipient of NATO funding, and elsewhere in this volume David Nofre explores the earlier process by which Italy successfully maneuvered to be thee location for the UNESCO International Computation Center.
[56] Woodger, "A History of WG2.3", quote on pages 1-2..

about it."[57] One of the interlopers, Mary Shaw, gave a presentation, after which "the discussion was entirely an American affair, and it was noteworthy for the inadequacy with which it was carried out. Among the European participants witnessing this discussion, the overwhelming feeling was one of embarrassment…. none of us did what should have been done… by remarking that this did not seem an adequate way of discussing this topic." Writing to its members he called for "an explicitly stated rule that everybody be outspoken and as clear as possible… in W.G. 2.3 we certainly used to apply such a rule, knowing full well that we would often display what looked like inconsiderable (sic.) behavior…. I also suspect that its former application is largely responsible for W.G. 2.3's former success…"[58]

This approach carried its own perils. According to a later chair of IFIP Technical Committee 2, its parent committee, "WG 2.3 has been a consistent group of famous people… its existence was visible to the outside only by acknowledgements to WG 2.3 in the publications of its members. Within the IFIP bureaucracy the group was therefore considered as an elite discussion group with no output."[59] In "reaction.. to this criticism" the group organized a reprint volume to showcase its output, including classic papers by the likes of Dijkstra, Hoare, Randell, Ross, With, Per Brinch Hansen and David Gries.[60]

The showcase volume gives a good sense of how the Algol dissidents pursued the agenda sketched in their minority report and ideveloped their vision of software engineering expressed at the 1968 NATO conference. As Michael Mahoney wrote "The call for a discipline of 'software engineering' in 1967 meant to some the reduction of programming to a field of applied mathematics." [61] The approach remained a fairly coherent one, at least through the mid-1970s, focused on correctness proofs, structured programming, programming for concurrency, and the development of programming languages to support these activities. Only Randell's interests diverged significantly from this approach to software engineering, looking turning toward fault tolerance.

Dijkstra's Crisis

And so we turn, at last, to the software crisis. Within the historical literature the 1968 NATO conference and the software crisis are inseparable. We might expect their reports to be crammed with discussionof the software crisis. Yet phrase "software crisis" appears only once in the proceedings of the 1968 conference, and that in an editorial aside when the introduction mentions that "There was a considerable amount of debate on what some members chose to call the 'software crisis' or 'software gap.'" [62] The word crisis also appears in quoted dialog between Ken Kolence (who does not like it) and Doug Ross (who does).The phrase was even less ubiquitous at the 1969 conference in Rome. In the proceedings of this conference it again appears only as an editorial interjection, this time in the

[57] Edsger Wybe Dijkstra, *EWD 603: Tripreport E. W. Dijkstra, St. Pierre-de-Chartreuse, 12-19 Dec. 1976* (1976); available from http://userweb.cs.utexas.edu/users/EWD/transcriptions/EWD06xx/EWD603.html.
[58] Edsger W Dijkstra, "EWD 611: On the Fact that the Atlantic Ocean Has Two Sides", in *Selected Writings on Computer Science: A Personal Perspective*, ed. Edsger W. Dijkstra (New York: Springer-Verlag, 1982):268-276
[59] P. Kurki-Suonio, *Technical Committee 2 -- Software: Theory and Practice* (n.d. [cited August 7 2010]); available from http://www.ifip.or.at/36years/t02kurki.html.
[60] David Gries, ed., *Programming Methodology: A Collection of Articles by Members of IFIP WG2.3* (New York: Springer-Verlag, 1978).
[61] Michael S Mahoney, "Computers and Mathematics: The Search for a Discipline of Computer Science", in *The Space of Mathematics*, ed. J. Echeverria, A. Ibarra, and T. Mormann (New York: De Gruyter, 1992):347-361.
[62] Naur and Randell, eds., *Software Engineering*, 120.

introduction: "realization of the full magnitude of the software crisis was the main outcome of the meeting at Garmisch."[63] In the body of that volume the word crisis appears once and once, in a statement from Dijkstra that "we all tell each other and ourselves that software engineering techniques should be improved considerably, because there is a crisis."[64]

This editorial endorsement of "the software crisis" as the main contribution of the 1968 meeting (above even the declaration of software engineering as a new field) points strongly toward the role of the editors in highlighting this part of the discussion. Brian Randell may have primary responsibility for this. He was coeditor of both conference reports, and in recent correspondence suggested of the introduction that "I think I wrote or at least drafted it…."[65]Also Naur, the coeditor of the first report, had initially opposed the inclusion of an introduction but Randell had made the case for its importance.[66]

Yet despite this deliberate push the phrase "software crisis" gained little traction until 1972. [67] Then Dijkstra made it a central theme of his Turing Award lecture "The Humble Programmer."[68] This was probably the single most important turning point in spreading both the phrase itself and a sense of a specific and potentially treatable malady afflicting large scale software development. He uses the exact phrase four times in this short paper, complaining that we are "up to our necks in the software crisis," asking "Is it a wonder that we found ourselves in a software crisis?" and observing that

> only a few years ago… to talk about a software crisis was blasphemy. The turning point was the Conference on Software Engineering in Garmisch, October 1968, a conference that created a sensation as there occurred the first open admission of the software crisis.

If this was blasphemy then the true faith was rare indeed. People had been complaining about software since before the term itself gained currency. Back in the 1950s the SHARE 709 System (SOS), then the most ambitious attempt at an operating system to date, proved late and disappointing.[69] In 1961 Herb Grosch wrote that hardware was generally successful but software problematic, and reminded his readers of "the SOS fiasco in Los Angeles and elsewhere, the Wall Street 501 delays; the Norfolk 220 debacle; gross failures to meet schedule on the part of almost every programming system package whether written in-house or contracted out."[70] In 1966 Carl Hammer, then director of computer sciences at Univac, noted in a public speech that "Software is considered by many the great enigma in the field of electronic computers. It appears as a source of frustration to programmers and it is

[63] Buxton and Randell, eds., *Software Engineering Techniques*, 7.
[64] Ibid, 13.
[65] Email to author, May 5 2010.
[66] Randell, *The 1968/69 NATO Software Engineering Reports* .
[67] The highest profile exception to this is the suggestion that "whether or not one calls it a crisis, there is a serious problem in the software industry" in Bernard Galler, "ACM President's Letter: NATO and Software Engineering", *Communications of the ACM* 12, no. 6 (June 1969):301.
[68] Edsger Wybe Dijkstra, "The Humble Programmer", *Communications of the ACM* 15, no. 10 (October 1972):859-866.
[69] Atsushi Akera, "Voluntarism and the Fruits of Collaboration", *Technology and Culture* 42, no. 4 (October 2001):710-736.
[70] H.R.J. Grosch, "Software in Sickness and Health", *Datamation* 7, no. 7 (July 1961):32-33, page 32.

frequently a matter of great concern to management."[71] The next year Robert V. Head wrote that "To say that the systems software being delivered with third-generation gear falls short of being an unqualified success would perhaps be the understatement of the decade." He bemoaned the computer "manufacturers' difficulties in providing flexible and efficient operating systems, file management systems, and other software aids" to their customers.[72] In 1970 Head listed no less than a dozen separate crises affecting the young software industry, none of them the crisis of large scale system software development associated with "The Software Crisis." Among them are a "crisis of program protection" due to an outdated intellectual property regime and a "crisis of marketing expertism" or rather the lack thereof.[73]

In those days crises were everywhere, reflecting both the cold war heritage of the Suez and Cuban Missile crises and general sense of social upheaval as the turbulence of the late 1960s gave way to the troubles of the 1970s. 1968 was, after all, the year of revolution and upheaval in Tokyo, Paris, Prague, Chicago and Berkeley as well as in the increasingly troubled world of Algol. Dijkstra, Randell, and the other proponents of the software crisis were far from the only people to seize on the rhetorical power of crisis to further their own agendas for computing. In data processing, for example, talk of crisis had a long history. The rhetoric of crisis had even used with respect to punched card machines, when a 1961 article proclaimed a "Crisis in Machine Accounting" because punched card supervisors had failed to take their managerial duties seriously.[74] In 1963 the American Management Association warned that "management is facing an information crisis" because of the "large gap between the possibilities of EDP as a profit-oriented management tool and its actual use in business today."[75] In 1967 a computing specialist at NASA warned of an "ADP Management Crisis", and advocates for the creation the new field of information science warned throughout the decade of an "information crisis." Throughout the 1960s the "totally integrated management information system" had been a widely held dream for the fusion of computers and management, but 1968 was the year of crisis in which *Fortune*, the *Harvard Business Review* and elite consulting firm McKinsey and Company all abandoned their support of the concept and began to publicize its failure.[76] Indeed the 1970 annual conference of the ACM was entitled "Computers and Crisis: How Computers are Shaping Our Future." The software crisis is not mentioned in its proceedings.[77]

[71] Carl Hammer, A Hard Look at Software: Keynote Address, American Management Briefing Session 6379-04, 1966, contained in CBI, Minneapolis
[72] Robert V. Head, "Old Myths and New Realities", *Datamation* 13, no. 9 (September 1967):26-29.
[73] Robert V. Head, "Twelve Crises -- Comments on the Future of the Software Industry", *Datamation* 16, no. 3 (March 1970):124-126.
[74] Arnold E. Keller, "Crisis In Machine Accounting", *Management and Business Automation* 5, no. 6 (June 1961):30-31.
[75] Gabriel N. Stillian, "EDP and Profit Making", in *Control Through Information: A Report on Management Information Systems (AMA Management Bulletin 24)*, ed. Alex W. Rathe (New York: 1963):42-44
[76] Thomas Haigh, "Inventing Information Systems: The Systems Men and the Computer, 1950-1968", *Business History Review* 75, no. 1 (Spring 2001):15-61.
[77] Participants do make reference to various political and environmental crises. The editor is Bob Bemer, who participated in the 1968 NATO Conference. While he does not mention the software crisis he does acknowledge the influence of the conference in another area: the heavily abridged style of the conference report itself and the associated use of tape recording and transcription to capture dialog. Robert W Bemer, "Editor's Preface", in

What, for Dijkstra, was the crisis? I believe it was the separation of practice from theory in the development of commercial systems software. Recall the background shared by many of the most distinguishing contributors to the Nato conference, such as Dijkstra, Hoare, Perlis, Naur, and Wirth. They were exceptionally gifted men, many with strong scientific backgrounds, who in the early- and mid-1960s had produced effective, elegant and reliable pieces of systems software with small teams and tiny budgets while working in research environments or for marginal computer firms. They were not by temperament successful corporate managers or project management specialists. So their instinct was to recoil as word spread of out-of-control industrial development efforts in which millions were being spend to produce unreliable, bloated, and incomplete operating systems and compilers. In his paper for the Rome conference Dijkstra wrote “I have no experience with the Chinese Army approach, nor am I convinces of its virtues.”[78] Fraser, a young researcher at the conference, spoke for many when he said “the designs I have been involved in… have not involved too many people… About large designs I don’t know.”[79] Instead their faith was that with the application of suitably rigorous and mathematical methods a small team could produce high quality systems software without the need for massive teams of programmers and elaborate bureaucratic controls.

Near the end of his life, Dijkstra wrote that

> The ALGOL implementation was one of my proudest achievements. Its major significance was that it was done in 8 months at the investment of less than 3 man-years. The rumour was that IBM's FORTRAN implementation had taken 300 man-years and thus had been a project in size way beyond what a university could ever hope to undertake. By requiring only 1 % of that labour, our compiler returned language implementation to the academic world, and that was its political significance.[80]

Dijkstra’s evolving views are not hard to document. He supplied friends and colleagues with a regular stream of material, photocopied and posted to them. Each item received its own EWD serial number. Their contexts ranged widely, from lengthy preprints of book chapters and technical papers to jokes, brief opinion pieces on the state of computer science, and book reviews. To a modern eye the series resembles nothing so much as a blog. One staple of the series was “trip reports,” in which he summarized in some detail his experiences at academic meetings. The persona Dijkstra presented to the world in this series of semipublic statements was not altogether pleasant. He prided himself on writing honestly, without respect to the feelings of others. For example it is not uncommon to read his account of making an invited visit to a university and learn that he considered its faculty members fools or charlatans. His comments on the work of others are usually critical and frequently dismissive.

Among Dijkstra’s many and passionate dislikes were managers, IBM, tact, the concept of software reliability, Algol 68, Fortran, Cobol, PL/1, political compromises of any kind, Alan Perlis, Harlan D. Mills, word processing, Microsoft, the mainstream mathematical community, System 360, American computer

Proceedings of the 1970 25th annual conference on Computers and crisis: how computers are shaping our future, ed. Robert W Bemer (New York: Association for Computing Machinery, 1970):(unnumbered).

[78] Buxton and Randell, eds., *Software Engineering Techniques*, 88.

[79] Naur and Randell, eds., *Software Engineering*, 50.

[80] Edsger Wybe Dijkstra, *EWD1298: Under the spell of Leibniz's Dream* (2000); available from http://userweb.cs.utexas.edu/users/EWD/transcriptions/EWD12xx/EWD1298.html.

science, lawyers, IBM's famous advertisements featuring "Susie Meyers" the secretary who could program in PL/1, logicians, user friendliness, PowerPoint, and anything justified on the grounds of practicality or cost effectiveness. A complete list of his dislikes would encompass a clear majority of the things, ideas, and people he writes about.

Yet his single biggest complaint, and one underlying many of the others, was against the idea that computer scientists must adjust their expectations to a world in which most programmers are intellectually mediocre and theory-averse. This is why he called himself a "humble programmer" in his Turing award speech, rather than, as his colleagues might, an arrogant (albeit insecure) academic computer scientist. Trained as a physicist he believed that the proper model for programming came from "mathematical engineering," a concept he claimed was well understood in the Netherlands but which thoroughly confused his American colleagues.[81] By seizing the mantle of programmer, despite his increasingly deep immersion in the world of mathematical theory and formal methods, Dijkstra could insist that he had not left programming behind. If actual programmers failed to grasp the importance of his new work then it was they who were betraying the true essence of programming.

Dijkstra gave a particularly clear statement of this in 1977, in an open letter circulated to fellow members of Working Group 2.3. According to Dijkstra, in the late 1960s it "became abundantly clear that we did not know how to program well enough." As a result "people concerned with Programming Methodology," by which he presumably means him and his fellow working group members, discovered that programming was "a tough engineering discipline with a strong mathematical flavor." However this inarguable conclusion was sometimes ignored, "because of the unattractiveness of its implications, such as,"

> (1) good programming is probably beyond the intellectual abilities of today's 'average programmer'
>
> (2) to do, hic et nunc, the job well enough with today's army of practitioners, many of whom have been lured into a profession well beyond their intellectual abilities, is an insoluble problem
>
> (3) our only hope is that, by revealing the intellectual contents of programming, we will make the subject attractive to the type of students it deserves, so that a next generation of better qualified programmers may gradually replace the current one.[82]

According to Dijkstra, only "political or emotional reasons" could explain disagreement with these "technical" observations. He conceded that they were "unattractive" and had "severe" social implications and no quick solutions. Clearly it would not be easy to replace almost the entire programming workforce. But he dreamed of a scientific revolution, in which programming would win recognition as a mathematical discipline and the incompetent hordes currently messing things up would be swept away and replaced by an elite corps of scientists versed in new and more formal methods of software development.

[81] Dijkstra used the phrase "mathematical engineer" in the discussion at Garmisch, and it also appears frequently in his writing from the late 1960s onward to describe the kind of programmer he was attempting to train. Naur and Randell, eds., *Software Engineering*, 127.
[82] Dijkstra, "EWD 611: On the Fact that the Atlantic Ocean Has Two Sides" .

Dijkstra seized on talk of a crisis at the 1969 NATO Conference on Software Engineering and spread to a broad audience with his 1972 Turing Award as a way to win support for his critique of the failure of computer companies to recognize the mathematical nature of software development and their insistence on hiring insufficiently intelligent people to do it. Only if the true magnitude of the crisis was appreciated would their clueless managers be forced to concede the necessity of this purge. The instrumental utility of the crisis concept to Dijkstra and his colleagues was not lost on those present at the NATO Conference. According to Donald MacKenzie, Albert Endres later expressed the view that "the notion of a 'software crisis' was an exaggeration that theorists had constructed to justify their work."[83]

He neither knew nor cared very much for the work of ordinary data processing programmers writing administrative programs in COBOL within user firms. Explaining why he had spared COBOL , the dominant data processing language, from a recent jeremiad against FORTRAN, the dominant language of scientific computing, Dijkstra explained "I would like to give a short explanation of my silence about COBOL. The point is that I have neither first-hand nor second-hand experience with COBOL's influence on its users."[84] The ignorance was mutual. The data processing literature of the early 1970s contained to references to software engineering and references to the software crisis remained largely within the academic and systems programming community. One of the earliest such references, in 1975, noted that "few data processing managers today even know of Dijkstra." [85]

Conclusions

In conjunction with the earlier chapters of this book, this epilogue has sketched an important trajectory running from the early days of systems programming practice in the mid-1950s through Algol, the Algol 68 minority report, the NATO Conferences on Software Engineering, and the IFIP Working Group 2.3 on Programming Methodology to culminate both in the prominent tradition of formal methods within computer science and the promotion of a "software crisis" as a justification for the quixotic project of remaking programming as a kind of applied mathematics.

Once enough detailed work has been done we are likely to discover that the NATO conferences deserve an important role in some of what Mahoney came to call the "histories of computing(s)" but not in others.[86] For example participants there knew little of ordinary data processing work, and in turn the conference and the associated rhetoric of a software crisis had little direct impact on data processing practice during the 1970s. On the other hand, researching this topic has convinced me that the triptych of Algol, the NATO conferences, and the early years of IFIP WG 2.3 provides important ways to think about the evolution of computer science. Mahoney's work on the origins theoretical computer science treated the development of mathematical models to reason about the behavior of programs as the central thread in this story, looking at the contributions of von Neumann, Hoare, Strachey, McCarthy,

[83] MacKenzie, *Mechanizing Proof*, 37.
[84] Dijkstra, *EWD273: The Programming Task Considered as an Intellectual Challenge* .
[85] Kenneth T. Orr, "Structured Programming: Not a Fad", *Infosystems* 22, no. 11 (November 1975):36-38. Perhaps the earliest mention of these ideas in a managerially oriented data processing publication was Jerry L. Odgin, "The Mongolian Hordes versus Superprogrammer", Infosystems 19, no. 12 (December 1972):20-23.
[86] Michael S. Mahoney, "The Histories of Computing(s)" (paper presented at the Digital Scholarship, Digital Culture, Centre for Computing in the Humanities, King's College, London, 18 March 2004).

and others.[87] He perceptively linked this story to the dominant initial vision of software engineering expressed in 1968.

Integrating the development of theoretical computer science more closely with the evolution of systems software and computing practice offers a different and complementary perspective. Software is, in many ways, a crystallization and embodiment of particular practices. It can be moved from one site to another, making those particles material and mobile. The history of systems software is inseparable from the history of the local groups responsible for its development and the communities of practice that grew up around successful systems. By tracing the careers and experiences of computer scientists such as Hoare, Dijkstra, and Naur we can reveal the importance of their early development work to their later careers and illuminate the centrality of Algol to the early development of computer science.

*

Today the software crisis occupies a much more prominent place in the work of historians than it does within the literature of the software engineering field. The clearest illustration of this comes in comparing two books on the history of software, published by Springer-Verlag within a few months of each other. *Software Pioneers: Contributions to Software Engineering* was based on a conference held in 2001 to which a diverse range of pioneers were invited. Among them were many of the key figures of early software engineering and programming methodologies such as Dijkstra, Hoare, Wirth, and Bauer as well as representatives of other software engineering approaches such as Fred Brooks, Barry Boehm, Tom DeMarco, David Parnas, Michael Jackson, Peter Chen, and Michael Fagan. Each pioneer contributed a classic technical paper to be reprinted and most presented new lectures reflecting on the origins and impact of their ideas. The 1968 NATO Software Engineering Conference was mentioned only once in the body of the text – in Dijkstra's own reflections.[88] Its proceedings were cited just once (a reference to Dijkstra's technical working paper included in Parnas's reprinted 1972 paper on operating system design). The word crisis does not appear, to the best of my knowledge or that of Amazon's search inside the book feature, on a single one of the 728 pages of this hefty volume. In contrast the software pioneers mention Algol on 64 different pages. Bauer, the main sponsor of the NATO conferences, did not mention them in his reflections but instead spoke at length about Algol.

The other volume, *Mapping the History of Computing: Software Issues*, is based on a workshop held in 2000. Historians set the agenda and provided the five main papers. These explore different aspects of software, for example "Software as Economic Artifact" or "Software as Labor Process." Yet, as Mahoney observed in the concluding essay of its proceedings book, three of the five "chose to begin with reference to the NATO conference." His own paper had a different opening but did discuss the NATO conferences and the mathematical tradition of reasoning about the behavior of programs. Thus four out of the five main contributions placed the conferences and their associated crisis close to the heart of their narrative. The exact phrase "software crisis" appears thirty times in its 283 pages.

[87] The most complete version of this story is in Michael S. Mahoney, "Computer Science: The Search for a Mathematical Theory", in *Science in the 20th Century*, ed. John Krige and Dominique Pestre (Amsterdam: Harwood Academic Publishers, 1997).
[88] Edsger Wybe Dijkstra, *EWD1308: What Led to 'Notes on Structured Programming'*, in *Software Pioneers: Contributions to Software Engineering*, ed. Manfred Broy and Ernst Denert (New York: Springer-Verlag, 2001).

Historians are rightly wary of reifying actors' categories in the framing of our own work. Sometimes this is unavoidable, but with the software crisis are in the strange position of having seized upon an actors' category that most of the actors themselves make little much less use of. We have also been too ready to echo Dijkstra's own words by saying something like "a software crisis was declared by attendees at the 1968 NATO Conference on Software Engineering." Given the actual spread of the idea, we might more usefully write "In 1972 a software crisis was proclaimed by Edsger Dijkstra to have been declared in 1968 at the NATO Conference on Software Engineering."

Likewise the 1968 NATO Conference should be seen in context as a largely unsuccessful attempt by a small group of people to preserve a community and an intellectual agenda that had evolved out of the Algol project and was ultimately sustained by the IFIP Working Group 2.3 on Programming Methodology. Earlier discussion, particularly that of Mahoney and MacKenzie, neglected the specific connections of the leading participants to Algol and their parallel development of the new working group but was at least squarely concerned with the development of theoretical computer science and formal methods for software engineering. More recently the conference and the crisis have drifted further away from these roots, evidenced in Tomayko's insistence that the conference included few academics and Ensmenger's attempts to stretch the crisis to encompass the experience of ordinary data processing workers and generalist corporate middle managers in the 1960s. The conference proceedings have been looted for pithy quotes, with which they are indeed stuffed, rather than explored carefully with respect to the biographies of the participants or the agenda of the editors.

Of course software engineering and, to a lesser extent, the software crisis are concepts that have taken on lives of their own and exerted considerable influence in the world. But this took place primarily from the late 1970s onward, and with an increasingly tangential relationship to the circumstances of their creation and the ideas and interests of their original proponents. The approaches favored by the Algol dissidents are today known as "formal methods." They played a progressively more marginal role within software engineering as it evolved from the late-1970s onward and the "software engineering" tag was freely borrowed and applied to a broad range of different ideas and products. This left many of the original enthusiasts thoroughly disillusioned. Randell later recalled that despite the collapse of the original initiative "the software engineering bandwagon began to roll as many people started to use the term to describe their work, to my mind often with very little justification."[89] Dijkstra, as usual, was harsher, writing in 1993 that "the programming manager has found the euphemism with which to lend an air of respectability to what he does: 'software engineering'." Recalling his initial embrace of the term, he noted that "As soon as the term arrived in the USA, it was relived of all its technical content… mathematics was widely considered impractical and irrelevant… but for the managing community it was a godsend."[90] Dijkstra's contempt for "the managing community" seems to have anticipated that of the comic strip Dilbert, in which the efforts of engineers are constantly being undermined by their perverse, proudly ignorant, and cliché-loving bosses.

Without appreciating this history it is hard to understand the connection between the title of Wirth's recent article, "A Brief History of Software Engineering" and its content, which outlines the development of programming languages and formal methods. Wirth never acknowledges that "software engineering"

[89] Randell, *The 1968/69 NATO Software Engineering Reports* .

[90] Edsger Wybe Dijkstra, *There is Still A War Going On: EWD 1165* (1993).

had, by the time it gained some kind of disciplinary reality in the 1980s, come to include project management, cost estimation, specialized personnel management, source code management and versioning systems, measurement of organizational capabilities, systems analysis methodologies, requirements analysis, software quality metrics, software testing, and a variety of other areas with no real connection to mathematics or theoretical computer science.[91] Most, though not all, of these approaches were missing (or at least marginalized) at the 1968 conference.[92] Many, such as the use of formal project management methodologies the documentation of user requirements, have little connection to the agendas expressed at the conference but can be traced back to earlier practice both in the cold war world of systems engineering and the more humdrum sphere of administrative systems and procedures work. We will also need to better document Bauer's own role in the early development of software engineering – he chose the name "software engineering" for the conferences, established their format, and secured funding from NATO. Yet he did not join his fellow Algol veterans in IFIP WG 2.3, and seems to have advanced a broader and more practical agenda for the putative field with a 1972 "Advanced Course on Software Engineering."[93]

So to understand the origins of software engineering as an academic discipline, research area, and identity for a relatively small group of system developers, we will need to move beyond ritual invocation of the 1968 NATO Conference on Software engineering to focus on how the concept was actually appropriated, extended, or modified during the 1970s and 80s. Likewise, putting the conference itself back into its proper historical place will require us to do a great deal of hard work to document the complex and indirect processes by which research work in software engineering and computer science eventually made its way into the software tools and methodologies used by ordinary programmers.[94]

[91] Niklaus Wirth, "A Brief History of Software Engineering", *IEEE Annals of the History of Computing* 30, no. 3 (July-Sept 2008):32-39. The accepted current scope of SW engineering can be taken from the Software Engineering Body of Knowledge, summarized at http://en.wikipedia.org/wiki/Software_Engineering_Body_of_Knowledge.

[92] Some of these topics were discussed at the 1968 NATO conference, particularly software performance evaluation and the possibility of automating testing.

[93] F L Bauer, *Software Engineering: An Advanced Course* (New York: Springer-Verlag, 1973) showcases pragmatic work distinct from the more theoretical tone of the IFIP WG 2.3 members at the time. Bauer stated with respect to this 1972 course that "In the years since 1968, remarkable progress in the technical sense has been made. Software engineering now comprises a bundle of techniques and principles which are hoped to help overcome the software crisis." F L Bauer, "Software and Software Engineering", *SIAM Review* 15, no. 2 (April 1973):460-480 This rush for immediate solutions is what annoyed Dijsktra and his compatriots at the 1969 Rome meeting. On the other hand Dijkstra and Hoare were fixtures of the NATO Marktoberforff Summer Schools that Bauer organized in the mid-1970s, and he quoted Dijkstra frequently in the aforementioned article, so this did not reflect any kind of estrangement.

[94] Again the technical people themselves are ahead of us here, with a stream of publications from the SIGSOFT impact project, thus far ignored by historians, aimed at doing just this. For example Barbara G Ryder, Mary Lou Soffa, and Margaret Burnett, "The Impact of Software Engineering Research on Modern Programming Languages", *ACM Transactions on Software Engineering and Methodology* 14, no. 4 2005):431-477.