What's in a name? Origins, transpositions and transformations of the triptych Algorithm -Code -Program

Liesbeth de Mol, Maarten Bullynck

What's in a name?

Origins, transpositions and transformations of the triptych Algorithm - Code - Program.

Liesbeth De Mol

Maarten Bullynck

Abstract

The purpose of this chapter is to focus on three connected and basic notions of computing, Algorithm - Code - Program, and to study their historical origins, their re-appropriations and transformations in different discourses and practices. Our main purpose is to show how these interconnected terms themselves have a complex history and cannot be fixed to one meaning only, even today.[1]

1 Introduction

While words such as "code", "program" and "algorithm" were once specific to specialist technical discourse, they now belong to popular discourse and refer on an everyday basis to what lies "underneath" technological products like Amazon, Facebook or Google. "Coding" and "programming" are those important "skills" that we need to teach our children and "algorithms" allegedly rule our world. But these words have a history and are not constant in meaning.[2] The history of words has a bearing on how history is or can be written and not taking into account their historical dimensions may lead to anachronisms, wrong interpretations or Whig history. Indeed, as Karine Chemla has noted before [3]:

> Historians often worked under the assumption that the main components of scientific texts problems, algorithms and so on are essentially ahistorical objects, which can be approached as some present-day counterparts.

This insight also applies to everyday usage. For instance, the current usage of "algorithm" in popular *and* scientific discourse often no longer refers to some

1 Both authors are supported by the ANR PROGRAMme project ANR-17-CE38-0003-01. This paper is a draft of a paper to appear in: J. Abbate and S. Dick, (eds.), *Abstractions and Embodiments: New Histories of Computing and Society,* Johns Hopkins University press, forthcoming.

2 Koselleck, R., Begriffsgeschichte, Suhrkamp, 2006

3 Chemla, K., 'On mathematical problems as historically determined artifacts. Reflections inspired by sources from ancient China', *Historia Mathematica*, vol. 36, nr. 3, 2009, pp. 213–246, here p.213.

mathematical object but rather to what one used to call software or programs, one may think, e.g., of the so-called Facebook algorithm. Still profiting from the mathematical origin of the word, it may suggest that the world is governed *not* by human-made and so potentially erronous, biased and complex technologies but by mathematics, making us blind for human bias in programs (so-called "algorithmic" bias) and the commercial interests behind it.

It is the aim of this paper to study the historical origins, transformations and interrelations of the triptych Code, Program and Algorithm (and Software) with the explicit aim to render these notions more transparent historically.[4] We will first focus on their origins and then show how the technologies of the 20th century reclaimed these words to reshape their meanings. From the initial small and technical contexts in which these words are used, they, later, will become part of general discourse.

2 Origins and the first technological appropriations

Today it seems natural to situate the words "code", "program" and "algorithm" in a computational context. But the words were already part of the English vocabulary long before the first computers. As with so many other words derived from Greek or Latin roots, they become proper members of the English reservoir of words in the 17th century during the Renaissance, though variants already appear in Middle English.

Algorithm

The word "algorithm" probably has the most intricate history. It was originally a latinized version of Al-Khwarizmi, surname of the Arab mathematician and astronomer Abu Ja'far Mohammed Ben Musa (c. 780 - c.850), known for his treatise on what is now called algebra and for his treatise that describes the Hindu way of reckoning. This treatise is now lost in its original Arabic version, but has survived in a number of medieval Latin translations dating back to the 12th century. As typical for the medieval, hand-copied, manuscript, they began by "dixit"(said), "inquit" (said) or "scripsit" (wrote) followed by the name of the original author. In this case "dixit algorismus", Al-Khwarizmi said. As the first (relevant) word of the codex, the name of the author became the way to refer to this manuscript, its title in a sense. With the many translations into popular languages, the word algorismus lost its original meaning and with time, the word came to refer to the content of Al-Khwarizmi's treatise, the Hindu system of writing and calculating numbers (viz., our decimal positional system and its rules of computation).

Once this meaning had stabilized, it was increasingly used to refer to other systems of calculating, either extending the decimal positional form of computation

4 Some work has already been done on this, specifically on the origins of "program". See: Grier, D. A., "The ENIAC, the Verb 'to program' and the emergence of digital computers", *IEEE Annals for the history of computing*, vol. 18, nr. 1, 1996, pp. 51–55; Grier, D.A., "Programming and planning", *IEEE Annals for the history of computing*, vol. 33, nr. 1, 2011, pp. 85â€“87; Haigh, T. and Priestley, M., "Where code comes from: Architectures of Automatic Control from Babbage to Algol", *Communications of the ACM*, vol. 59, nr. 1, 2016, pp. 39–44. However, we have found new sources which locate the origin of "program" more exactly.

or creating new forms analogous to it. These include the algorithm of fractions or proportions, for doing calculations with fractions , or, later, also the “algorithm of infinitesimal differentials” on the Continent or “algorithm of fluxions” in Newtonian England. In the famous priority dispute between Newton and Leibniz over the invention of the calculus, the algorithm of the calculus even became a point of contention in itself, some claiming that while Newton invented the method, Leibniz invented the algorithm.[5] This goes to show that a subtle differenciation develops, between the conceptual solution of a problem (the method) and the material or notational implementation of that solution (the algorithm). The closeness of notation and algorithm only loosens up during the 19th century, when mathematicians will increasingly use algorithm with the meaning of a schematic computational process or of a stepwise procedure. Together with some other words, such as method, procedure, rule or calculus, algorithm becomes one of the words mathematicians use to denote (semi-)formalized (numerical) solution to a (mathematical) problem.[6]

Code

The word “code” derives from the Latin word “codex”, that is, a collection of texts, in particular law texts. In its transferred meaning, it also refers to a system or collection of rules to be followed, such as a code of honour or a receipt or prescription for preparing a certain medicine. Thus we see that “code” in this original meaning had more in common with “algorithm” in its usual understanding of today. Only in the 19th century the term “code” starts being used in its more modern sense where it refers to the use of a system of signs to “encode” a certain information. This usage of “code” dates back to the invention of the telegraph system. It had been common practice in the 18th century to encrypt diplomatic or important commercial messages by one long word. These words were collected in a book or in a codex. This practice became more widespread with the telegraph where these code words were both cost-efficient and secure and could refer to short messages, stock, companies, etc. E.g., “shamefaced” stood for “sell at current market price”. These codes were then “translated” into a so-called “telegraphic alphabet” and “telegraphic language”, e.g. combinations of long-short signals. It was only later that “code” shifted and referred also to this telegraphic alphabet and language, that is, Morse code or Baudot-Murray code.[7] When in the early 20th century punched cards were introduced for Hollerith machines and other meccanographic devices, the word “coding” was introduced here too to denote the process of translating questionnaires or data sheets into the “language of punched cards”, holes and notches.[8]

Program

Finally, the word “program” was a 17th century neologism derived from the concatenation of the two Greek words προ (before) and γραφειν (writing), a pre-

5 Lazare Carnot, *Betrachtungen über die Theorie der Infinitesimalrechnung*, translated and augmented by J.C. Hauff, Frankfurt, 1800, p. 84.

6 See e.g. Church, A., “An unsolvable problem of elementary number theory,” *American Journal of mathematics*, vol. 58, nr. 2, pp. 345-363.

7 Friedman, W.F., The history of the use of codes and code language, Washington, 1928.

8 Herman Hollerith, The electronic tabulating machine, Journal of the Royal Statistical Society, Vol. 57, No. 4 (Dec., 1894), pp. 678-689, here p. 684.

scription[9], a written notice of things to come. Slightly later, it gained its more modern meaning as an advance notice to describe the ordering of an activity like, for instance, the program of a concert or, more generally, a general plan or scheme of something to be done, like an itinerary, a training schedule, a production plan etc.

As "code" met up with telegraphy, so did "program" meet up with another technological advance in the late 19th century: "program(me) clocks" developed to "*furnish a convenient and practical clock, that may be set to strike according to any required programme*".[10] These kind of devices could be applied to automate time schedules or production plans, ringing at preset times on a factory work floor, at railway stations or in a school. With time, these devices grew more complicated and more generally deployable "program devices" or "program machines" were invented to automate the operations of machines such as a paper cutting machines, a washing machine etc. In that context, "program" came to stand for the automatic carrying out of a sequence of operations or as an automated scheduler.

In parallel with this development, the word "program" was also picked up in the context of radio engineering. At first, a program referred to the *physical* program as transmitted within a broadcasting network. With the rapidly expanding broadcasting industry and increased network complexity, the problem of scheduling "programs" in different networks became important. Any program had to be connected to the right station at the right moment at so-called "switching points" Originally, this was done by an operator who had to "*listen for cues indicating the end of a program, and then operate the proper keys or change connections*" [11], but an increase in the number of switches made this impossible. Therefore, the manual switching had to be (partially) automated through (relay) switching equipment. In this context, "program" steadily transposed from radio programs to the technology itself, with terms like "program circuits", "program trunks", "program switching", "program line", "program loop", etc.[12]

Though all three words (algorithm, code and program) were items present in the language's lexicon since the 17th century, they were recycled in new contexts that were both specialist and technical already in the late 19th and early 20th century. 'Algorithm' lost its footing in notation to become a more general mathematical term as 'stepwise procedure', shifting from what Kenneth O. May has called "mathematical technology" to "mathematical science". 'Code' then, was taken up by the professional telegraph users to denote any form of compression of messages, be it through code words or (binary) code symbols. From there on, 'coding' became the verb to refer to the, often repetitious and boring, activity of translating language into code words or symbols for telegraphy, telephony or also card punching. Finally, 'program' entered the engineering discourse in the mid 19th century to talk about the

---

9 Thanks to Martin Carlé who provided an analysis of "pro-gram" in terms of its Greek origins and so, also, its connotation of pre-inscription, cfr. his talk *Literate Programming, containerisation and the future of Digital Humanities* at the Autumn meeting of the PROGRAMme project in Bertinoro. Slides are availabale here.

10 Estell, S.F., *Improvement in programme-clocks*, U.S. patent nr. 98678, patented January 11, 1870.

11 Murphey, P.B., *Broadcast switching system*, U.S. patent nr. 2238070, filed May 10 1940, granted April 15 1941.

12 The origins of "program" will be discussed in more detail in our paper *Roots of "program" revisited*, revised version under review for the *Communications of the ACM*.

automation of time schedules, production plans, or, later only, the switching of radio programs in a network and the automatic sequencing of operations. Thus, even before our three words became part and parcel of computing vocabulary, they had already migrated from their general meaning to usages in specialist and technological communities. It is from these communities that they will eventually spread to computing.

3 The second appropriation under the sign of digital computing

The roads leading to the modern digital general-purpose computer are many, but a classic and important passageway remains the large electromechanic and electronic calculators built in the 1940s, mostly to help in the computation of ballistic tables. Here the traditions of business computing, scientific computing and military command and control meet with the reliable relay and the pioneering electronics technologies of automation. This encounter will shift the semantics of ‘code’, ‘program’ and ‘algorithm’ into computing.

The mathematical field now called ‘numerical analysis’ had started to grow since the end of World War I and the many groups and bureaus for (manual or machine-aided) computation had slowly developed their own techniques and workflows.[13] The mathematicians involved sometimes used the word ‘algorithm’ to talk about the steps of the computational procedure to be followed, sometimes they also used ‘rule’, ‘formula’ or ‘procedure’, but the most common word used was without a doubt ‘method’.

While each computation group had its own ways of organizing computation, most of them had some kind of three-tiered process. First came the mathematical analysis of the problem (including finding the right ‘method’ in the literature), then came the preparation of the problem for the human computers using a “computing plan” and finally the computation itself and the taking down of the results using “computing sheets”. When this manual (or machine-aided) process was ported to large mechanical calculators such as the ASCC/Harvard Mark I or the Bell machines, there was continuity with these practices but replacing the human by a machine changed some things. Usually, mathematical analysis of the problem still came first. After that, the method had to be prepared in a form that allowed an easy set-up on the machine and finally the machine computation itself. In order to prepare a problem for set-up on the machine, one had to ‘code’ the computation in a form that was machine-understandable, on the other hand, one had to operate the machine during execution. This whole process from mathematical analysis (planning) to setting up the problem on the machine (coding) would eventually evolve in *one* of the definitions of programming in the 1950s. The widespread use of the word ‘program’, however, did not originate in this context.

Instead, the germ of what our modern word ‘program’ would become, lies in the application of automatic control to calculating machines. Already in the 1930s engineers had started to build ‘program devices’ (See Sec. 2) to control the sequence of operations in calculating machines. With the new complexities of time scheduling and radio broadcasting, also the automation of transfer of control or even conditional

13 Grier, D. A., *When computers were human* Princeton: Princeton University Press, 2005.

transfer of control had been achieved. As IBM engineer James W. Bryce describes in a 1937 patent: “Such programming means will enable the operator to program the sequence of transfers and to selectively route the transfers from any selected accumulator to any other selected accumulator” [14]. The IBM team that would develop the ASCC/Harvard Mark I with Howard Aiken applied this technology to the machine. As a consequence, they speak in the patent of a “program tape” (instead of the later “control tape”) that “schedules the operations of the machine, selecting the functions and the sequences of their performance”.[15] Further mention is made of a “print program”, or a “multiplying program”, but in the later ASCC/Mark I manual[16] these have disappeared and are now referred to as e.g. “multiplying sequences”. Grace Hopper and other members of the ASCC/Mark I team would later speak of “coding routines” or sequences. The routines frequently needed would be stored in a “tape library” containing “control tapes [that] are of general application”.[17] Thus, whereas the engineers, who were thinking in terms of the automation of control via program devices spoke of ‘programs’, Aiken and his team, who were coming from the mathematical and human side of the problem, spoke of “coding routines” or, more frequently, “sequences”.

From a certain viewpoint, coding the ASCC/Mark I can be seen as the automation of a machine set-up. That development can be found in other contemporary machines too. Samuel Caldwell and Vannevar Bush devised a control mechanism to automate the set-up of their improved differential analyzer in the 1940s.[18] Before, the machine had to be set-up manually connecting all the right parts of the machine – a process that took several hours. Now, the connections, empirical data and initial conditions were all coded on tape which then controlled the automatic set-up of the analogue analyzer. The Bell machine relay calculators model III to V all had a battery of Baudot-coded tapes with both data and instructions to automatically set up the right calculation to be executed. All these machines, however, could only follow a sequence of coded instructions. Coded iterations and full automatic conditional transfers were, at least initially, not possible.[19]

---

14 Bryce, A.E., *Cross-adding accounting machines and programming means therefor*, U.S. patent nr. 2,244,241, filed Oct. 1 1937, granted June 3, 1941.

15 Lake, Claire D., ‘Zero eliminating means’, patent US 2,240,563, filed Aug. 31, 1938, granted May 6, 1941.

16 Hopper, Grace et al, “A manual of operation for the automatic sequence controlled calculator”, *The annals of the computation laboratory of Harvard University*, vol. 1, London, Harvard University Press, 1946.

17 It is quite well-known that the coders of ASCC/Mark I developed a systematic practice whereby they stored pieces of code that were checked out and known to be correct in Notebooks. These books could then be used later to just copy down existing code and which was thus a maual process of subroutining. As Hopper explained later, it was this manual practice that very much affected her later work on one of the first compilers, cf. Computer Oral History Collection, Grace Murray Hopper (1906-1992).

18 Bush, Vannevar and Caldwell, Samuel H., “A new type of differential analyzer,” *Journal of the Franklin Institute*, vol. 240, no. 4, 1945, pp. 255–326.

19 Coded iterations and automatic conditional tranfer of control were later introduced on the ASCC/Harvard Mark I. This was achieved through a special device built by Bloch and which was called the “Subsidiary sequence unit”. It did have a conditional stop which allowed the moderator to move the tape to another position. They also had a practice of gluing tapes together in order to have loops. For the Bell machines,

These control structures were, however, present on ENIAC once it was ready for service (1946). That machine differed fundamentally from its mechanical and analog contemporaries because it was electronic and thus achieved a much higher speed of computation. However, in its original form, it had to be manually set-up, not unlike the original differential analyzer. It is here where the most momentous transfer of meaning to 'program' would happen. Mauchly's original short 1943 proposal for an "electronic computer" already referred to a "program device", which later became a 'program control unit" and evolved into the ENIAC's "master programmer" which centrally controlled the local programming circuits for sequencing loops and conditionals. It is from there that the term in ENIAC developed, playing over different semantic extensions, referring both to individual (control) units (as in "program switches"); smaller pieces of an entire program (as in "program sequences"); or the complete schedule that organizes program sequences (as in a "*complete program [for which] it is necessary to put [the] elements together and to assign equipment in detail*").[20] "Program" in this context always refers to how automatic control, locally or globally, is organized.

The practice of putting computations on the ENIAC, however, rapidly made the term 'program' drift further away from hardware to be transferred to the organization of a computation. In an appendix to a 1945 report entitled *Remarks on programming the ENIAC*, Eckert, Mauchly and others wrote about the entire "computational program" and how to " *to link the elementary programming sequences into a complex whole*" using an "*elaborate hierarchy of program sequences*" that could be built up with the master programmer and which relied on, what they call, "sub-routines" [21]. In another report written by Haskell B. Curry and Willa Wyatt planning for ENIAC to do ballistic calculations, the problem "*is studied with reference to the programming on the Eniac as a problem in its own right*" [22]. They develop a method where each computation is "*broken into pieces, called stages*" which are defined as "*a program sequence with an input and one or more outputs.*" Linking together the inputs and outputs, smaller program sequences could be combined into more complex programs and a general "schedule" of programs could be translated into wiring diagrams that indicate how ENIAC should be set-up. The semantics of the "program device" discourse is still at play here, but generalizes from the sequencing of operations to include also the scheduling of sequences of operations.

The intricacy and time-consuming process of manually setting up the ENIAC for computation and the sheer speed of the machine, led to ENIAC being rewired, resulting in an automation of the set-up process. In this new configuration, a sequence of coded instructions could be introduced on ENIAC, through the setting of switches or the reading of punched cards.[23] Now the logic of linking elements together in a

---

iteration was present in Model III, conditional jumps only on Model V.

20 Curry, H.B. and Wyatt, W., *A study of inverse interpolation on the Eniac*, Aberdeen Proving Ground, Maryland, Report nr. 615, 19 August 1946.

21 Eckert, Presper J., Mauchly, John W., Goldstine, Hermann H, Brainerd, J.G., *Description of the ENIAC and comments on electronic digital cgoldomputing machines*, Contract W 670 ORD 4926, Moore School of electrical engineering, University of Pennsylvania, 30 November 1945., p. 3-7.

22 Curry, H.B. and Wyatt, W., idem, p. 6.

23 See Haigh, T.; Priestley, M. and Rope, C., 2016. *ENIAC in action. Making and Remaking the modern computer*, MIT Press, 2016. Of course, the introduction of coded instructions was inspired by the relay machines.

program that automatically controlled the machine could be done through symbolic encoding. As a result, the word ‘program’ slowly transferred even further from hardware towards what is now called software.

It is commonly known that the word “program” in its current meaning comes from ENIAC.[24] But it is the encounter of automating sequences and scheduling with calculation that put “program” there in the first place. Actual practice then made “program” shift further, from actual hardware to the configuration of that hardware. It is in that sense that Douglas Hartree defines “program” as “*the process of drawing up a schedule of the sequence of individual operations required to carry out the calculation*”.[25] Compared to the Harvard Mark I team, Hartree is less focusing on planning and coding sequences (viz. the work of the human computer and its translation to the machine), but rather on how to translate the plan into a configuration of the program device that will start the required operations, either in the form of wiring diagrams or in the form of symbolic coded instructions.

4 Shifting frontiers in the pioneering 1950s

The 1950s can be characterized as a period of pioneering projects to build a reliable digital electronic computer. The work towards reliability, mass producibility and standardization is also reflected in the attempts to define basic terms in glossaries and shape common practices. Looking at usages and definitions of ‘program’ and ‘code’ in different professional and local contexts brings out how fluid their meanings still are in the 1950s. Especially the development of automatic coding or programming in the mid 1950s impacted on the semantic envelope of these words.

Looking at the influential work by Goldstine and von Neumann[26], they do not use the word ‘programming’, but instead differentiate between planning and coding of a problem. After the mathematical preparation of a problem. A flowdiagram is introduced to plan the computation and on its

, viz. the mathematical analysis of the problem, and coding, viz. both flowcharting and the actual coding of the flowchart as computer instructions. This distinction was often picked up, but with the important shift that ‘programming’ was now used too, referring to the whole process of planning, flowcharting and coding, whereas ‘coding’ would now be reduced in meaning to the machine coding part only.

This is the distinction one finds the 1954 ACM Glossary, viz.,

> Code (verb): to prepare problems in computer code or in pseudocode for a specific computer.
>
> Program (verb): to plan a computation or process from the asking of a question to the delivery of the results, including the integration of the

24 Grier, D. A., ‘The ENIAC, the Verb “to program” and the emergence of digital computers’, *IEEE Annals for the history of computing*, vol. 18, nr. 1, 1996, pp. 51–55. and Haigh et al. 2016.

25 Hartree, Douglas R., *Calculating instruments and Machines* , Urbana, University of Illinois Press, 1949, pp. 111-112.

26 Goldstine, H.H., and von Neumann, J., *Planning and coding of problems for an electronic computing instrument*, Volume 2 of *Report on the mathematical and logical aspects of an electronic computing instrument*, part I,II and III, 1947-48, Report prepared for U. S. Army Ord. Dept. under Contract W-36-034-ORD-7481.

> operation into an existing system. Thus programming consists of planning and coding[27]

Some definitions restrict 'programming' further and contrast it with 'coding, e.g. the Bureau of Standards in 1948 define see 'program' as a "*general verbal description of the method of solving a particular problem on a computer*". This is even more pronounced in the IBM glossaries of the 1950s, for the IBM 650, one author even writes: "Programming and flow charting are synonymous – the remainder is mere coding."[28] This shows that 'programming' has shifted once more in the 1950s. From structuring automatic control, in some contexts it can now generalize to the whole process of planning and coding or, in some cases, even be restricted to one aspect of planning, viz., flowcharting.

While a strict differentiation was made between flowcharting and coding, usually, coding was subsumed under programming. Or, put differently, coding was just another task of the programmer besides flowcharting. This becomes explicit in a 1951 discussion:

> L.A. Ohlinger (Northrop Aircraft Compay): I would like to ask how many programmers and coders are employed in order to keep UNIVAC busy full time?
> J.L.McPherson: We do not distinguish between programmers and coders. We have operators and programmers.[29]

Indeed, while both activities of flowcharting and coding can be distinguished in theory, they cannot in practice, the same person has to do both.

However, classic computing history has it that a distinction between the jobs of 'programmer' and 'coder' existed, the first being occupied with problem analysis and flowcharting, the latter with porting the flowchart to the machine.[30] In practice, however, no such distinction appears and even a report from the United States Department of Labor describing the *Occupations in electronic data-processing systems*[31], identifying no less then 13 different occupations in electronic data processing, only uses "coder" in reference to "coder clerks". These are people who "convert items of information obtained from reports and records to codes for processing by automatic machines", viz. the people who already coded information in the beginning of the 20th century. The coding of instructions, however, is considered to be part of the job of the programmer. What did exist was a distinction between the chief programmer or systems analyst and junior programmers. Evidently, it is reasonable to assume that flowcharting was more in the hands of the former and coding more in the latter category, even though a neat separation in practice was impossible.

---

27 First ACM Glossary prepared by a committee, chairwoman Grace Hopper, 1954.
28 R.V. Andree, Programming the IBM 650, 1958, p. 81.
29 J. Presper Eckert, James R Weiner, H Frazer Welsh, Herbert F Mitchell, The UNIVAC system, AIEE-IRE '51: Papers and discussions presented at the Dec. 10-12, 1951, joint AIEE-IRE computer conference, 1951, pp. 6–16
30 The claim, with implications for gender distribution, is prominent in Ensmenger, N., *The computer boys take over. Computers, Programmers, and the Politics of Technical Expertise*, Cambrdige, Massachusetts, MIT Press, 2010.
31 United States Department of Labor, Occupational Analysis Branch of United States Employment Service, *Occupations in electronic data-processing systems*, 1959.

It thus seems a fair question to ask where the ‘myth’ of the coder comes from? The answer lies in Grace Hopper’s influential talks on automatic coding or programming.[32] Hopper remarks that “the analyst, programmer, coder, operator and maintenance man were separated”, though, admittedly, the “distinction between a programmer and a coder has never been clearly made”. Her distinction is, a programmer “prepares a plan for the solution of a problem” (viz., a flow chart), while a coder has “to reduce this flow chart to coding, to a list in computer code.” The motivation for this, self-admitted, artificial separation, becomes clear in the rest of the paper: “It is this function, that of the coder, [...] that is the first human operation to be replaced by the computer itself.” Thus it is the introduction of “automatic coding” (sometimes als unfortunately called “automatic programming”) that accounts for a retrospective, artificial distinction.

This observation helps to explain the different ways ‘coding’ and ‘programming’ are used. In UNIVAC and IBM circles usage was mostly according to the definitions quoted above, ‘coding’ was translation into machine code, ‘programming’ was either everything from planning to coding, or the part before coding, viz. planning and flow charting. This is in part a reflection of the hierarchy on the working floor of a commercial computer installation. In other places, especially at universities, both words are used interchangeably or coding is subsumed under programming. The ambition there was often to make the computer accessible to every user, in particular scientific users not necessarily versed in engineering and machine details. This contrasts with commercial computer installations where the user had to pass through the programmers and operators to get his work done on the machine.

A symbolic ‘readable’ way of programming the machine directly after problem analysis, was first championed by Hartree or Wilkes in the U.K.[33] or Zuse and Rutishauser in Germany and Switzerland. On the EDSAC a symbolic assembler-like code was developed so “the machine may be said to understand the same language as a computor”.[34] Or, quoting Aleck Glennie on his Autocode system for the Manchester Mark I, “we must make coding comprehensible, [t]his may be done only by improving the notation of programming.”[35] Similarly, at M.I.T.’s Whirlwind it was decided early on to make the computer available to the “casual user” through “automatic standard subroutines” that “can be used almost as easily as an equivalent built-in order, with resultant saving in the programmer’s time.” Eventually this resulted in one of the first “automatic coding systems”, “a comprehensive system of service routines [...] to simplify the process of coding.”[36]

With “automatic coding systems” of the 1950s ‘coding’ becomes less of an issue and ‘programming’ increasingly is the focus of human effort. This is tangible in the definitions of ‘program’. In November 1952 a ‘program’ on the Whirlwind is defined as a “ program is a sequence of actions by which a computer handles a problem”, a

32 Grace Hopper, Automatic Programming â€" Definitions, Symposium on Automatic Programming for Digital Computers, Office of Naval Research, Department of the Navy, Washington, D.C., 13-14 May 1954, p. 1–5.

33 Wilkes, M.V., Wheeler, D.J. and Gill, S., *The preparation of programs for an electronic digital computer*, Addison-Wesley, 1st edition 1951, 2nd edition 1957.

34 Wilkes, M.V., ‘Programme Design for a High-Speed Automatic Calculating Machine,’ *Journal of scientific instruments*, vol. 26, nr. 6, 1949, pp. 217–220.

35 A. Glennie, The Automatic Coding of an Electronic Computer, lecture notes 1952.

36 Project Whirlwind Summary Report no. 22 first quarter 1950 p. 24 and no.31 third quarter 1952, p. 12

definition still close to Hartree or the EDSAC team, and a ‘coded program’ is a “set of instructions that will enable a computer to execute a program.”[37] In December 1952 then, now with automatic coding, it becomes a “ program is an ordered sequence of words, written with the intention of having it typed on paper tape in the (new) Flexocode and inserted in [Whirlwind I] by the intermediary of the Comprehensive Conversion Program”.[38] This shift suggests the one that will happen later, end of the 1950s, when so-called ‘programming languages’ would become used and ‘program’ will become a ‘text’ in those languages.

This is also true for the commercial computer firms who also invested in automation of the programming process. Looking at how this happens at IBM or UNIVAC one finds the following. The IBM FORTRAN system is “a IBM 704 program which accepts a source program in a language [...] resembling the ordinary language of mathematics, and which produces an object program in 704 machine language, ready to be run.” Thus, “A FORTRAN source program consists of a sequence of source statements, of which there are 32 different types.” [39] Equally, Univac’s FLOW-MATIC is described as shifting “the programming effort from detailed coding to problem definition and system analysis”, and MATH-MATIC “describes the problem from the user’s standpoint, rather than the program required by the hardware of the computer”: As a consequence writing FLOW-MATIC or MATH-MATIC ‘programs’ amounts to writing ‘sentences’ and the “conversion of the problem, expressed in pseudo-code, into the necessary program, in machine code, is performed entirely automatically and internally”.[40] This more ‘linguistic’ or even ‘syntactic’ definition of ‘program’[41] would later be confirmed by ALGOL’s definition of ‘program’: “sequences of statements and declarations, when appropriately combined, are called programs”[42], and, one years later, “A program is a self-contained compound statement, i.e, a compound statement which is not contained within another compound statement and which makes no use of other compound statements not contained within it.”[43]

5 Business and science, or, software and algorithm?

The years around 1960 mark a double evolution, the surge of computer service industry and the slow establishment of computing as an academic discipline, both can

37 MIT Computation Laboratory, Memorandum M-1624-1, p. 1.
38 MIT Computation Laboratory, Engineering Note E-516, p. 3.
39 J.W Backus et al., *THe FORTRAN Automatic Coding System for the IBM 704*, IBM: Poughkeepsie 1956, p. 7
40 Ash, R. et al, *Preliminary Manual for MATH-MATIC and ARITH-MATIC Systems for ALGEBRAIC TRANSLATION and COMPILATION for UNIVAC I and II*, Automatic Programming Development, Remington Rand UNIVAC, 19 April 1957.
41 Compare also with Nofre, D., Priestley, M. and Alberts, G., “When technology became language: the origins of linguistic conception of computer programming, 1950-1960,” *Technology and Culture*, vol. 55, nr. 1, pp. 40–75.
42 Perlis, A. J., and Samelson, K. Preliminary report–international algebraic language.Comm. ACM 1, No. 12 (1958), 8-22, and Numer. Math. 1 (1959)
43 Naur, P. (ed.). Report on the algorithmic language ALGOL 60. Comm. ACM 8 (1960), 299-314, and Numer. Math. 2 (1960), 106-136.

be tracked down in the usage of two words, viz., 'algorithm' and 'software'. As the statistics show, the rare word 'algorithm' starts to spread from 1958 onwards, the neologism 'software' from 1961 onwards. Their appearance and fast dissemination is indicative of the self-consciousness of new professional groups. As the prevalence of the term 'software' in the trade magazine *Datamation* shows, it is mainly used by the computer service industry (later: software industry). The term 'algorithm', on the contrary, is most present in the publications of the ACM, a professional society of computer scientists. Both words were 'launched' into professional and public discourse purposefully.

'Algorithm' had been used by numerical analysts and computing professionals occasionally before 1958, but the choice to name the new international scientific programming language ALGOL, acronym of 'ALGOrithmic Language' made the word a household term. As is well-known, the origins of the ALGOL-language date back to meeting of German and Swiss mathematicians and it was allegedly Heinz Rutishauser who repeatedly used the word "algorithmic notation" since 1955, but Herman Bottenbruch who coined the phrase "algorithmic language".[44] The German mathematicians had been starting to use the word "algorithmischer Programm" because it allowed to introduce a new distinction. The "algorithmic language" introduced something in between a mathematical solution to a problem and the program in machine code: "Such algorithmic notations, as we shall call them, have the appearance of classical mathematical notation but include certain dynamic elements which remind one of ordinary programming."[45]

While the first description of ALGOL in 1958 was still couched in mathematical terms and spoke of a 'International Algebraic Language',[46] the 1959 description moved to 'algorithmic'. With the fuzz created around ALGOL in some circles, say the ACM and universities, the term 'algorithm' gained currency, while words as 'method' or 'rule' faded away. In particular, the specially created 'Algorithm section' in the *Communications of the ACM*, first edited by J.H. Wegstein, to publish "algorithms consisting of "procedures' and programs in the ALGOL language" (CACM February 1960. 3 (2)) helped to establish it. It also shows that 'algorithm' is to be situated at the boundary between 'method' and 'program' and may refer to both. So ALGOL became the "internationally accepted method of describing numerical calculations in most journals devoted to computation"[47]

When Donald E. Knuth, from 1962 onwards, started work on his *Art of Computer Programming* series, and chose to talk about "analysis of algorithms" rather than "non-numerical analysis" as the topic of series,[48] the word 'algorithmic' became even more cemented as a key word, certainly for those who could call themselves computer scientists. This also shows up in the statistics. While 'algorithm' surges

---

44 Durnova, H and Alberts, G., Was Algol 60 the first algorithmic language?, IEEE Annals of the History of Computing, IEEE Computer Society, 2014, 36 (4), p. 104-106.

45 H.C. Schwarz, An introduction to ALGOL, Comm. ACM 5 (Feb 1962), 82-95.

46 Perlis, A. J., and Samelson, K. Preliminary report–international algebraic language.Comm. ACM 1, No. 12 (1958), 8-22

47 R.E. Grench and H.C. Thatcher (ed.), Collected Algorithms 1960-1963 from the Communications of the Association for Computing Machinery, p. iii, Argonne National Laboratory, report ANL-7054).

48 D.E. Knuth, *The Art of Computer Programming: Fundamental algorithms*, Addison-Wesley, 1967, p. vi-vii.

around 1960 (the ALGOL effect) in all three publications, it remains a rare word in the trade journal *Datamation*, whereas it features prominently in the *Communications of the ACM* since 1960, though it is subject to some waves of fashion.

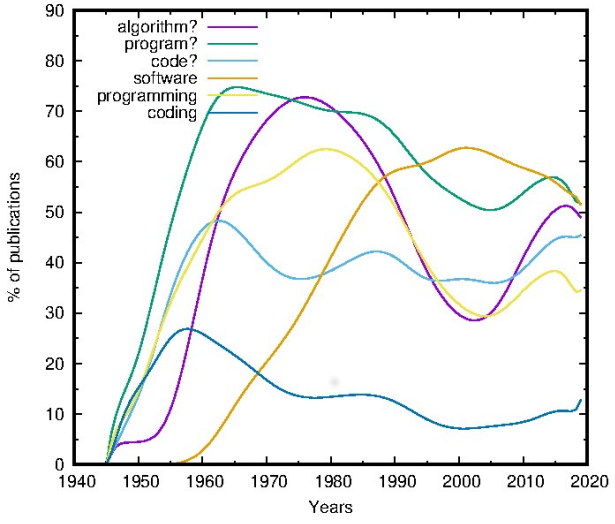

Illustration 1: Data from the Communications of the ACM 1954-today

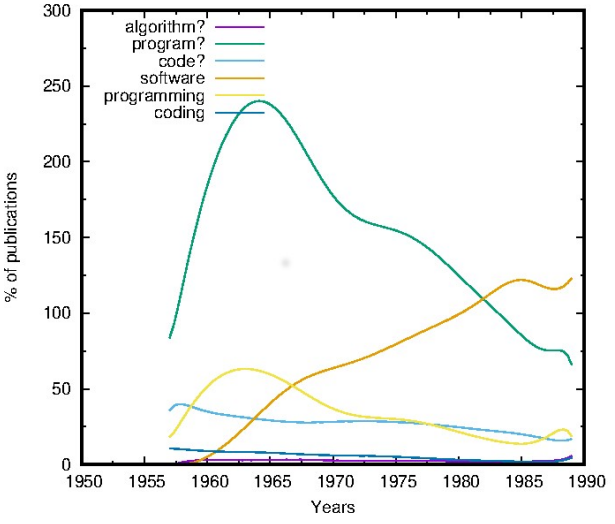

Illustration 2: Data from Datamation (1957-1989)

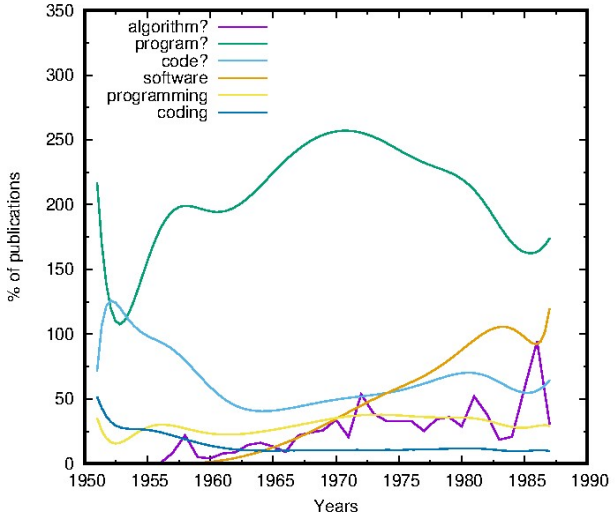

Illustration 3: Data from the AFIPS Proceedings (1951-1987)

Apparently, the new word ‘software’ started off as a joke in the 1950s, the other side of the more common ‘hardware’ of the military or the computer industry.[49] The word, however, only was taken up from 1960 onwards, as the statistical graphs convincingly show. ‘Software’ is first used in 1960-1961 and then goes on to become a common term. Again, the graphs also show it is a term particular for a specific community, viz. those involved in the computer business and service industry as represented by *Datamation*. The term also appears in publications of the ACM or the IEEE, but it never reaches the same prevalence among computer scientists and engineers.

In fact, the word ‘software’ starts its course in advertising. The first occurrences in journals on the West Coast are all in job advertisements, at first those of Ramo-Woolridge, later of other companies too.[50] On the East Coast, the word is used at first be the computer service firm C-E-I-R.[51] In both cases, the word may have been chosen for accounting reasons. Ramo-Woolridge, a primary consultant for the U.S.’s space program, was under a “hardware ban” (viz. forbidden to sell hardware to the military, to avoid monopoly). Calling their services ‘software’, even if that meant microprogramming computers, might have been a decoy tactics. In a similar vein, C-E-I-R had had problems capitalizing the costs encurred over training programmers or developing programs[52] and might thus have come up with a new term to make this investment more tangible.

The term certainly stuck quickly. It surfaces in the development of COBOL and, at first, is mainly used to refer to programs that help to program, such as compilers, assemblers, utility or monitor programs.[53] But soon ‘software’ becomes the generic term used by the programming services industry for talking about programs as a commodity, as a commercial product. The prominent role of C-E-I-R’s president H.W. Robinson at ADAPSO may have played an important role in establishing the word. IBM, who had had a quarrel with C-E-I-R around a rental of their STRETCH computer, at first dismissively defines ‘software’ as a ‘slang term for programming system’ but will adapt the term eventually too in their programming service bureaus. As the graphs of *Datamation* show, with the spread of ‘software’ one observes a slow but steady decline of the word ‘program’, substituting the ‘neutral’ program for the ‘commercial’ software, turning the ‘programming service industry’ into ‘software industry’.

---

49 Many claims to first uses exist, e.g. Paul Niquette, Grace Hopper, or in the RAND Corporation, but they all have in common that it was originally used as a joke-like designation of things not hardware.

50 E.g., “Senior programmers are urgently needed to help develop a large ‘software’ package for commercial and military applications for R-W stored logic computers”, Datamation 1961, 1.

51 E.g, “The art of programming – it can scarcely be called a science yet – has grown concurrently in the past 9 or 10 years with the hardware. The resulting large systems of programs have now reached such a degree of complexity and power as to rival the machines. This ‘software,’ as it is currently called”, Datamation, 1960, 9.

52 Interview H.W. Robinson by Bruemmer, 13 July 1988, Oral History, Charles Babbage Institute.

53 See Haigh, Software in the 1960s as Concept, Service, and Product, IEEE Annals of the History of Computing 24 (1), 5-13, 2002.

6 Discussion and Outlook

After the preceding pages it is rather euphemistic to say that the words ‘program’, ‘code’, ‘algorithm’ (and to a lesser extent ‘software’) have been continuously subject to semantic change. As a matter of fact, the semantic ‘tectonics’ still goes on. We left ‘program’ and ‘code’ around 1960 but their further evolution may be gleaned from the statistical graphs. As ‘automatic coding’ and ‘programming languages’ gain currency the frequency of “coding” goes down, while “programming” goes up. “Coding” in most cases now mainly refers to coding data or encoding or decoding practices. In the 1980s and, more recently in the 2010s, ‘coding’ and ‘coder’ became a bit more popular again. ‘Coding’ now has become a colloquial term for ‘programming’ emphasizing the recreative side of programming, closer to ‘hacking’, contrasting it with the professional business of programming applications.[54] One can also observe from the graphs that the popularity of ‘programming’ has fallen since the economic crisis of about 1973, while at the same time ‘software’ continued its upwards trend. If the ACM statistics (the only ones going beyond the 1980s) are representative, it would seem this evolution was returned around 2000 when ‘software’ lost ground to ‘program(ming)’ and to ‘code(ing)’ again. But of course, more research is needed to interpret these data.

The word with the most differentiated behavior in our graphs is ‘algorithm’. All graphs shows a slight surge of popularity around 1960 when ALGOL was all the rage, but afterwards it is barely mentioned in the trade magazine *Datamation*, whereas in the AFIPS proceedings it is slowly but steadily used, but it is in the ACM publications that it features most prominently, even if subject some ‘seasonal’ variations. It peaks around 1975 with a downwards trend afterwards until 2000 when the trend goes up again. The heydays of structured programming might play a role here, but there is also a hint from the fact that the Algortihms section of the ACM becomes a journal in its own right, *Transcactions on Mathmatical Software*, changing the focus from algorithms to programs or even software again.[55] But again more research would be needed to correctly interpret these data.

It is important, finally, to note that nowadays the word ‘algorithm’ has begun to be used in a broader sense still. Examples are Google’s algorithm, Facebook’s algorithm, algorithmic trading or even algorithmocracy, a state form where political decisions are influenced or even formed by algorithms. This signification of ‘algorithm’ extends

---

54 According to Haigh and Ceruzzi’s soon to be published new version of *A history of modern computing* it might have been the inflation of job titles such as ‘analyst’, ‘software engineer’, ‘program architect’ etc. that made the less pretentious ‘coder’ seem to be more appealing, cf. SIGCIS discussion list, May 5 2020.

55 “The publication of algorithms in TOMS replaces the algorithms department of Communications of the ACM […] From the top-down view of science, TOMS is partially on the top, abstract level and partially on the second, more concrete level. The top level, for example, is represented by fundamental research papers on the analysis and critical evaluation of computer programs; the second level is represented by practically oriented, concrete research and development in traditional areas like linear algebra, polynomial manipulation, and nonlinear programming. [...] This means that the listing must be replaced by something more suitable for reading (e.g. English text, very high level languages, flowcharts of the second or third level of a top-development of the programs).” John R. Ryce, Purpose and Scope, ACM Transactions on Mathematical Software, vol. 1, nr. 1, 1975, pp. 1-3.

from the stepwise recipe-like numerical or non-numerical procedure to the complex or system of programs and parameters that underly parts of technological infrastructure. This recent evolution of the word would certainly merit detailed historical and political scrutiny, especially given its boost in popularity with the latest wave of artificial intelligence. A superficial browsing of our data seems to suggest that its roots are to be found in the late 1960s or early 1970s when engineers began to speak of 'scheduling algorithms' in operating systems.[56] It further gained traction in the 1980s when some communities started using the word in a more general way, such as researchers in artificial intelligence (e.g. David Rummelhart or David Marr) or economists devising programs for automatic trading on the stock market.[57]

"What's in a word?" Surely the small differences, shifts and interrelationships of words reflect how people invest their vocabulary with new distinctions, ambitions, ideas and horizons. They may reflect practices or encounters, or voice professional or disciplinary ambitions. Once taken up by a community, or even adopted as a part of public speech, the subtle dynamics of and between words often disappears and gets more cemented. It becomes invested with economic or political interests and becomes part of the often subconscious set of values and distinctions that characterize everyday vocabulary. Looking beyond the screen of words is a necessity and history helps to identify and critically engage with the forces that drive words today.

56 'Scheduling algorithm' and the like account for a large part of the increase in using the word 'algorithm' in the AFIPS proceedings.

57 The economists knew the word through 'linear programming', a mathematical theory developed in the late 1940s.