Telling the long and beautiful (hi)story of automation!

d'Udekem-Gevers, Marie

Published in:
Making the History of Computing Relevant

Publication date:
2013

Document Version
Publisher's PDF, also known as Version of record

Link to publication

Citation for pulished version (HARVARD):
d'Udekem-Gevers, M 2013, Telling the long and beautiful (hi)story of automation! in *Making the History of Computing Relevant: 17-18 June 2013 London (IFIP & Science Museum).* Springer, Dordrecht, pp. 173-195.

Telling the long and beautiful (hi)story of automation!

Marie d'Udekem-Gevers

University of Namur, Faculté d'informatique
Rue Grandgagnage 21, B-5000 Namur, Belgique
marie.gevers@unamur.be

Abstract. The purpose of this paper is to suggest and justify a framework for the history of computing that interests a wide public. The aim is to set the history of computing in the much broader context of automation, while also addressing the evolution of ideas. It suggests first a new detailed classification of programs (in the broad sense). Then it tries in particular to sketch out a "phylogenesis" of automation from the 12th to 19th centuries in Europe. It discusses various automatic devices: particularly, clocks and their annexes, but also organs, games, looms and early computers. Finally, it addresses the stored-program computer and high-level languages.

Keywords. Automation, Blaise Pascal, Charles Babbage, clocks and their annexes, conditional branching, languages, programming, regulation, relevance to the general public, sequence, stored-program computer.

1 Introduction

According to French historian Fernand Braudel [5],

> "History occurs at different levels [...]. On the surface, the history of events works itself out in the short term: it is a sort of microhistory. Halfway down, a history of conjunctures follows a broader, slower rhythm. [...] And over and above this 'recitatif' of conjuncture, structural history, or the history of the longue durée, inquires into whole centuries at a time. It functions along the border between the moving and the immobile, and because of the long-standing stability of its values, it appears unchanging compared with all the histories which flow and work themselves out more swiftly, and which in the final analysis gravitate around it. [...] In any case, it is in relation to these expanses of slow-moving history that the whole of history is to be rethought, as if on the basis of an infrastructure. All the stages, all the thousands of stages, all the thousand explosions of historical time can be understood on the basis of these depths".

To make the history of computing attractive, I would like, following Braudel, to suggest that we examine, in a general manner, the long-term, multidimensional history. In particular, I propose to focus on the history of *automation* from

Antiquity to the 'stored program computer', while also addressing the related *mental evolution* and highlighting the *dematerialization* that occurs on the technical level at the same time as the *increased autonomy* of the machine.

For the past 10 years, I have had the experience each year of giving lectures (lasting 30 minutes to 30 hours) to students at the University of Namur (Belgium) or on a more sporadic basis during speeches given to a wider public. In my experience, this approach generates a great deal of interest on the part of students. It could, I believe, also be used in a museum setting. In this presentation, I therefore propose to explain and justify the main thread of my approach.

Long histories, tracing the full evolution of a specific object from its origins, are rare in the field of science and technology. Those relating to automation are particularly infrequent, since they involve a wide range of applications and call on specialists in very diverse disciplines. We can cite two such studies, both published after 1960: the paper by Jean Sablière [49] which at one time was a primary reference in the field, and the more recent article by Brian Randell [44]. The latter is a major contribution to this subject and is essential reading. Randell analyses the origins of computer programming, focusing on the concept of sequence control. To do this, he reviews early automatic devices (pointing out the "pegged cylinder" as a programming medium), Vaucanson's and Jacquard's contributions (underlining that punched media are "clearly separate"), Babbage's contributions, the contributions of some of his direct successors and, finally, the stored-program concept (taking into account, moreover, the ability of the machine to calculate the addresses of the variables, but not envisaging high-level languages).

The current paper revisits Randell's analysis and generalizes it, while supplementing it with additional information. It proposes a broader conception of the notion of a program[1] and provides a detailed typology of this concept. Furthermore, it integrates that concept into the much broader one of automation. The frame of reference that I propose is both generalized and inclusive: it is aimed not only at providing a rigorous definition of any automaton, but also coherently describing the evolution of automation, leading to the development of stored-program computers equipped with software enabling programming in high-level languages. Two comments need to be made at this point. First, it should be stressed that this framework was designed in the spirit of texts written by specialists in computer organization and design, in particular J. P. Meinadier [39] as well as D. A. Patterson and J. L. Hennessy [41]. Second, it should be noted that application of this frame of reference to the history of automation should, in the end, be a tool to help non-computer scientists understand what a computer that enables programming in high-level languages is.

Furthermore, in his introduction, Randell [44] states the following about his own choice of historic milestones:

> "It is important to realize that many of these particular developments have been selected more because I personally find them interesting than because of any contemporary importance (…)"

[1] The definition proposed here is also broader than that of A. G. Bromley [9] who in

Like Randell, I have selected[2] stages in history that are interesting in order to illustrate the analytical perspective that I have adopted. However, our perspectives do not completely overlap, and the milestones I have chosen differ slightly from Randell's. In contrast with Randell, for example, I have examined in greater depth the history of mechanical clocks and their annexes, highlighting the passage from the wheel (not discussed by Randell) to the cylinder as well as the transition, which from my point of view is fundamental, from stationary pegs to moving ones (which Randell mentions only in passing).

Now that this brief, annotated review of the literature on the long-term history of automation has been completed, we should address the history of different fields that over the centuries were affected by automation. It should be noted that, in general, these fields do not intersect, are unfamiliar with each other and use different vocabularies. Furthermore, these specialized literatures often omit to give pertinent details for those who are interested in automation, since they generally have other preoccupations. For example, a specialist in the history of carillons is interested in the complexity of the melodies played via automation, but not in the programming of those melodies. Overall, the literature pertaining to automata reveal, when compared side by side, many points of confusion, gaps and inconsistencies. It is therefore particularly important to start by providing some definitions. Examples and illustrations will come later. First, let us recall that the word "automaton" comes from the Greek "*automatos*" and thus means etymologically "that which moves by itself." According to Devaux [15], "automation" denotes a mechanism by which the more or less complex sequence of operations takes place spontaneously each time it is triggered. I suggest that an automaton[3] be defined as a machine incorporating automation. Following Sablière [49], Jacomy [29] and Gille [21], let us consider that two principles allow automation to function: regulation (which implies simultaneity in relation to the process being regulated) and programming (which implies precedence in relation to the process being programmed). Let us also note that the two principles are not incompatible. Regulation is synonymous with *feedback* in a broad sense. As defined by Wikipedia**, "**feedback describes the situation when output from a phenomenon in the past will influence an occurrence or occurrences of the same phenomenon (or the continuation / development of the original phenomenon) in the present or future". As to programming (in a broad sense[4]), it can be defined as determining in advance the sequence of operations (or actions or motions) that have to be performed by a machine and then recording this sequence on a medium that serves as a memory.[5] It is useful to emphasize that all programming therefore implies, by definition even, the existence of a memory.[6] A program (in a broad sense) is an ordered set of operations (or actions or motions) performed by an

[2] Steps not addressed here may, in particular, be found in an article by Randell [43].

[3] This concept is not taken into account by J. Lafitte [31].

[4] As concerns the semantics of the word 'program", my attitude is therefore opposed to that of A. Brennecke [7] which considerably limits the meaning of this word.

[5] Definition adapted from J. Marguin [38].

[6] A so-called "*stored program"* used by the stored-program computer is thus only a special case.

automatic system. I would like to explicitly mention an elucidating distinction between the program as it is conceived by humans and the program as it is 'understood' (i.e. read) by the machine.

Starting from the understanding of Meinadier [39] that there are three major categories of programs which have succeeded each other over time, and adopting his vocabulary, I would also like to propose a new (more detailed) classification of programs (as they are understood by the machine). According to Meinadier's typology [39] (see Table 1), a program is said to be internal (to the machine) if it is fixed. Furthermore, a program is termed "external"[7] by Meinadier if, like Jacquard's loom, it is not unchangeable but rather is (manually) interchangeable. Personally, I will explain this concept later but, as B. Randell [45] suggested to me, I propose to consider the category of external programs as containing two sub-categories, which are above all relevant for comparing calculating machines[8]: the sub-category of exchangeable programs (taken into account by Meinadier) but also the more primitive sub-category of programs that are manually modifiable in situ[9]. In any case, such programs are accessible to the user. Finally, a program is defined as a "stored program" in the last possible case. Meinadier does not note that this last type of program is fully 'manageable' automatically, (as we will describe at length below) i.e. by the machine itself. However, as we are going to see, this point is fundamental.

Table 1, Suggested classification (and evolution) of programs (as understood by the machine).

Programs (as understood by the machine)	
Classes	Feature
1. 'Internal'	Fixed
2. 'External'	
2.1. In situ	Modifiable *manually*
2.2. Logically separate from the programmed device	Replaceable *manually*
3. 'Stored program'	Fully manageable *automatically*

[7] D. A. Grier [23] stresses that the expression "external programming" was employed for the first time in 1951. This was in a "paper by IBM researchers which attacked the concept [of stored-program computer]. This paper described the IBM card programmed calculator, which was an accounting machine connected to an electronic arithmetic unit" [23].

[8] This distinction is not however very useful for analyzing the history of pegged cylinders used in musical automata: these cylinders either have mobile pegs (e.g. in carillons) or exchangeable ones (e.g., in organs), essentially as a function of their size.

[9] I would, however, like to stress that there are intermediate cases.

We still have to state a basic detail that is rarely addressed in the literature: a machine can have several levels of programming (possibly of the same class). This configuration is quite frequent and applies to most of the automata that will be discussed in the text.

2 Automata prior to stored program computers

We should begin by stressing the importance of the Mechanicians' School of Alexandria [18] (3rd century BCE - 1st century CE) as pioneers in the history of automation. These early mechanicians, using, on the one hand, pegged[10] wheels and, on the other hand, levers, ropes, cylinders,[11] pulleys, etc., were able to build automata, notably in the fields, closely related at the time, of religion and theater. It is certain that they used the principle of programming. However, in contrast with what certain authors have stated, it seems that they did not use any regulation [25] [26].

We can now move on to the successors of the Alexandrians as concerns automata: the Byzantines[12] and the Arabs. The latter collected the manuscripts of Alexandria and translated them into Arabic. They also refined some of the techniques: we can see that they used the principle of regulation in addition to programming in order to build automata.[13] Furthermore, it has been proven that the oldest description of a "pegged cylinder" used to automate the playing of a musical instrument is the work of the Musa brothers, in 9th century Bagdad [32] [34]. The "flute player" automaton that they described was very sophisticated and based on water power. To produce the desired music automatically, a wooden programming cylinder had to be made, with pegs of the proper dimensions and in the right places, and this cylinder had to be rotated at the proper speed. The Musa brothers provide a great volume of technical details: for example, they state that several melodies can be memorized on a single cylinder. They also explain that if one wanted to have several melodies for the mechanism, it was better to build a wide cylinder (rather than one with a large diameter) [32]. However, it seems that they did not raise the possibility of modifying a cylinder or replacing it, once the automated musical

[10] According to their actual shape, the pegs (or teeth or cams) offer, as a general rule, the possibility of producing different movements (from all or nothing [binary effect], all the way to very progressive movement ["continuous" effect]). At the time of the ancient Greeks, it seems that each peg was an equilateral triangle [17].

[11] As underlined by B. Randell [44], Heron of Alexandria did not use 'pegged cylinders' as such but rather their forerunners implying ropes. "The rope contained eyelets which fitted over pegs protruding from the cylinder, and was fixed to the cylinder with wax. Thus as it was slowly pulled off the cylinder, the cylinder rotated first one way, then another, at various speeds [44]". I would like to add that this technique is primitive to the extent that it does not involve any long-term memorization of the sequence of the desired movements: when the rope is unwound, it must be rewound properly around the cylinder and refastened before the process is started again.

[12] See, for example, the water-clock from Gaza (Palestine) circa 530 (see G. Dohrn-van Rossum [16]).

[13] Via conical valves (cf. Hill [25] [26]).

instrument had been made. The program can therefore be described as internal, according to the classification proposed above.

The knowledge of the Arab world was transmitted to the West, in particular by means of translation into Latin, starting in the 12th century, in Muslim-occupied Spain. Let's now turn to Europe.

2.1. Programmed automata not used for calculation in Europe

According to Dohrn-van Rossum [16], from the 12th century, technical innovations were made in Europe to wake the monks: the direct coupling of one or several bells and a water-clock as well as the automation of the repeated striking of one bell (ringing) or of several bells (tune). This automation is based on an internal program.

One of the very rare accounts of this type of programming can be found in the rather rough representation on the reverse of the last folio of a manuscript dating from the reign of James II of Catalonia (1291-1327)[14]: it shows a mechanism consisting of a water-clock (on the bottom), and a set of bells (on top) that the clock would ring at a given interval. According to Farré-Olivé [20]: “This could be used as an alarm, or as a diversion for a party of assembled guests, or at some gathering of people.” On the drawing, a set of bells can be seen (only five are shown), above which is a pegged wheel. This pegged wheel can definitely be considered a program: we can imagine that the pegged wheel would rotate and strike the bells, creating a sequence of sounds represented on a musical score, drawn in the top left part of the original document. This program can be characterized as internal. It should be noted that it is more rudimentary than the mechanism proposed by the Musa brothers: a wheel can be seen as a cylinder with a single track... It should also be stressed that the drawing does not give any indication “of how the pins actuate the hammers placed by the bells below” [20]. As Farré-Olivé [20] mentions: “we are presented with a structure conveying the main ideas, without practical details.”

The next stage in the history of automation is a major event of the Middle Ages, even though it seemed to have passed virtually unnoticed at the time: the invention of mechanical clocks. This probably occurred in the 13th century simultaneously at several European monasteries. It apparently predates the representation of the Catalan clepsydra, but as Farré-Olivé [20] explains:

> “The old water-clocks and the new weight clocks could well have existed side-by-side for a period of time… because the water-clock time measurer would be far less costly to construct than any type of early weight clock.”

[14] The drawing may be a later addition. According to the analysis by Farré-Olivé (p. 374), it was undoubtedly made between 1291 and 1430.

For purely technical reasons, the invention of mechanical clocks led to the passage from unequal hours[15] (as measured by water-clocks) to equal hours[16]. In addition, as Dohrn-van Rossum [16] notes:

> "Mechanical clocks offered the possibility of animating large, heavy automata and of keeping them running more reliably than could be done until then with hydraulic movements."

It therefore became possible to build mechanisms known as "monumental astronomical clocks", which were used for entertainment purposes. This involved hoisting the clock and its annexes into a tower (initially a church tower) in order to make them more visible and more audible. According to the classification proposed by Lehr [32], the annexes of mechanical clocks implemented from the 14th century onwards in Europe can be divided into three groups: astronomical and time measurement instruments (astrolabes, dials with representation of astral movements, calendars, etc.), musical instruments (automated mechanical organs, carillons, etc.) and devices for the theater (automata representing, for example, angelic musicians or the Magi, etc.).

The hypothesis explaining the invention of mechanical clocks, as formulated by J.D. Robertson and reiterated by Dohrn-van Rossum [16], is as follows: technology progressed from a bell-ringing mechanism to a clock escapement mechanism controlled by a foliot (see Figure 1). We should identify the basic components of these two mechanisms: a (vertical) crown wheel (which can be considered as an internal program), a verge with two blades (engaging alternatively the wheel teeth) and a weight. The difference lies in the results obtained. In the case of the bell-ringing mechanism, it is the repeated beating of a clapper on a bell. In contrast, the clock escapement mechanism, linked to the weight, will drive a back-and-forth movement of the foliot and its weights: the effect is to divide the flow of time into a succession of units. From this point on, time measurement no longer had to be based on a continuous flow of liquid, as was the case with the clepsydra, but rather on a counting of units [6]. We should also note that the weighted foliot is a regulator of escapement: it sets the duration[17] of the periods of crown wheel movement, separated by periods of rest [40]. A mechanical clock can therefore be described, in the vocabulary adopted here, as a programmed and regulated automaton.

Hourly bell-ringing involves a much more complex mechanism than the movement of the mechanical clock and, according to Dohrn-van Rossum [16], it was invented only in the early 14th century, probably in the city-states of Italy. Bell-ringing relies on the production of a toothed wheel, or "count wheel": the breadth of the bump determines the number of times the bell rings. This is manifestly a case of automation based once again on a centralized, internal program. And this time the invention, though it may appear relatively commonplace to us, was considered

---

[15] By convention, the period of daylight was always 12 hours in winter and in summer: it was therefore the duration of an hour that varied by the seasons and by the latitudes.

[16] Equal hours imply that the duration of one hour is set as 1/24th of one day.

[17] This length of time may be modified by the displacement of two weights.

very significant at the time. It was only by being combined with hourly bells that mechanical clocks would become widespread, first in the cities and later in the countryside. This led to a secularization of time (public clocks) [16].

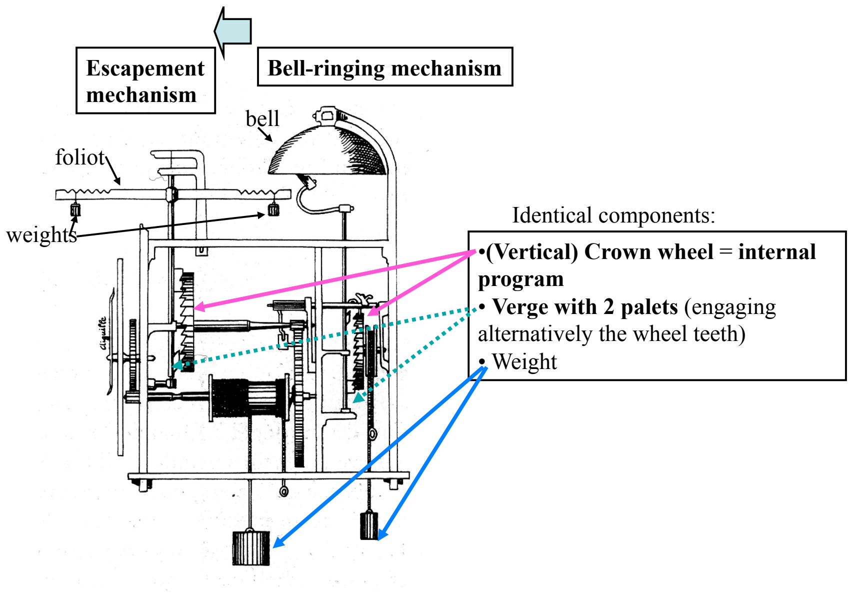

Escapement mechanism
Bell-ringing mechanism
bell
foliot
weights
Identical components:
•(Vertical) Crown wheel = internal program
• Verge with 2 palets (engaging alternatively the wheel teeth)
• Weight

Fig. 1. Diagram of a mechanical clock with both escapement and bell-ringing mechanisms.

A manuscript[18] dating from the mid-14th century shows an improvement in a variant[19] of the programming technique for carillons or organs: the melody to be played was still short and simple, but now it could be modified thanks to the possibility of moving the programming pegs[20]. Of course, there is no proof that the described innovation was ever actually implemented. And while this new feature may seem of little importance from a musical point of view, it is fundamental from the perspective of the present article. Could this not be seen as the first mention of an external program (modifiable in situ) in the history of European, and most likely world, technology?

[18] Kraków, Universytet Jagielloński, Biblioteka Jagiellońska, ms. 551, fol. 47 rev. – 49 rev. (cited by A. Lehr [32]).
[19] The pegs are not located around the circumference of the wheel, but rather on several radii of the programming wheel.
[20] On this subject, see A. Lehr [32] and A. Lehr *et al.* [34].

The next step in perfecting automated carillons led, in the late 15th century, to a greater number of bells and to increased length and complexity of the melodies. For programming, this involved the use of several toothed wheels, arranged in parallel, thus forming a sort of cylinder with fixed pegs [34] (see Figure 2), equivalent to the cylinder drawn by the Musa brothers in the 9th century. Then, around 1530 according to Lehr [33][21], automated carillons were built with moveable cylinder pegs (see Figure 3): this made it possible to change the tunes, now potentially quite sophisticated, that the carillon could play [34].[22] It is therefore certain that by the early 16th century in Europe, mechanicians had built automata with external programs.

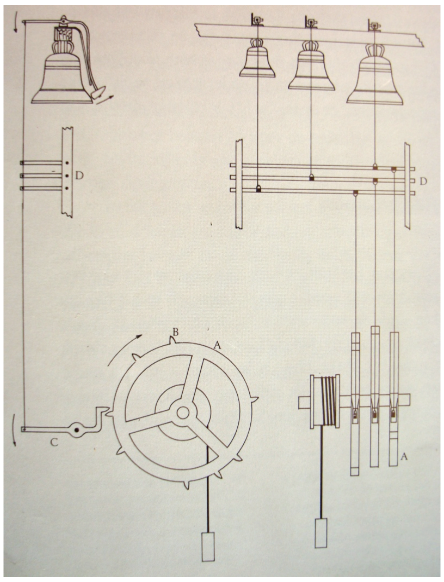


Fig. 2. Diagram of the oldest programming cylinder (with several wheels) for a carillon (late 15th century).

[21] This innovation is sometimes attributed to a certain Barthélémy Coecke or de Koecke, also known as the "Buffoon of Aalst." However, such an attribution seems to be pure fancy, in light of the results of in-depth studies on this subject by A. Lehr. According to Lehr ([33] and [34]), the "*Zot van Aalst*" only became known as "Barthélémy Coecke" in the 19th century, and his role in the improvement, in the 15th century, did not involve the programming cylinder, but rather the technique enabling manual playing of the carillon. Furthermore, the text written by a monk from Saint Michael's Abbey in Old Flemish, mentioning "eenen sot van aelst" and giving rise to far-fetched interpretations, has been commented by L. Rombouts [46].

[22] "A regular grid of holes on the cylinder made it possible to move the cleats and thus modify the piece of music, particularly as a function of the religious calendar. However, these changes were rarely made more than four times each year […]" (Buchner and Rouillé [11]). See also: http://whc.unesco.org/en/list/943/video.

This brings us to the automata developed after the Middle Ages, independently of clocks. We should note first of all that it was difficult to “notate” a programming cylinder for a musical automaton, i.e. to place the pegs in the proper locations to obtain the desired melody. In the 17th century, the idea was developed of using paper tapes to simplify the notation of the cylinders for mechanical organs. First, the punching was performed flat on sheets of paper, and then these punched sheets were glued on the cylinder [14].

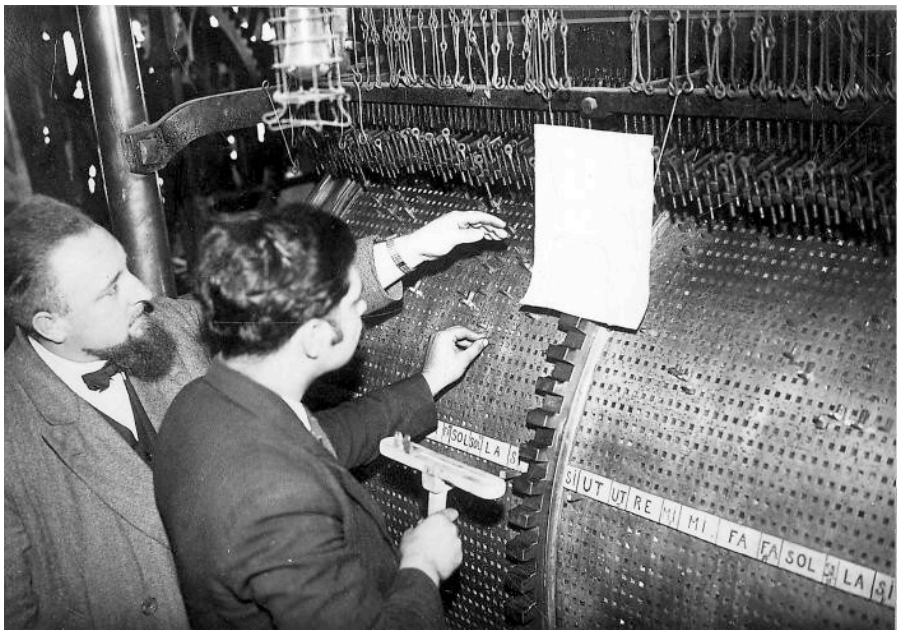


Fig. 3. Staf Nees (1901-1965), Director of the Royal Carillon School "Jef Denyn" (Mechelen, Belgium), programming a cylinder with moveable pegs for a carillon.

We now move on to the 18th century, known as the golden age of automata. Two categories must be distinguished: entertaining or educational automata, and looms, or more precisely, automated machines for the weaving of designs. To illustrate the first category, let’s consider the “*Writer”* [57] made in 1770 by the clockmaker and mechanician Pierre Jaquet-Droz. This automaton is capable of writing *any text with a maximum of 40 letters*. This is a very interesting example, since its programming contains two levels. It is, in fact, equipped with an internal program:

> “A stack of rotating cams, causing three levers linked to the writer’s hand to move. One lever controls right-left movements, another back-and-forth movements, and the third vertical movements. There are three cams for the formation of each letter” (Devaux [15]).

It also contains an external program: a wheel with 40 interchangeable pegs. This wheel “raises all of the cams, sliding on its [vertical] axis, in order to successively bring the cams for the different letters to the levers” [15]. Other famous entertaining automata were built in the 18th century, notably by Jacques Vaucanson. The latter moved from making entertaining or educational automata to producing automated looms.

This leads us quite naturally to address this type of non-entertaining, utilitarian automata. Looms that reproduce designs were the focus of a remarkable innovation. In 1725, Basile Bouchon abandoned raised media for the memorizing of a program and used a medium with punch holes[23], i.e. a punched *paper strip,* that can easily be replaced. Along with Daumas [14], we can formulate the hypothesis that Bouchon was inspired by the process used to mark pegged cylinders since the 17th century. We could say that he simply eliminated one step in the manufacturing process for an external program. As Rouillé [47] notes:

> “The concepts of zero and vacuum are not natural ones. The use of zero was not generalized in the West until the Middle Ages, and it was only in the 17th century, with Toricelli’s experiments on atmospheric pressure that we really became aware of what the total absence of elements meant – a vacuum. Therefore, it is not surprising that we had to wait until the early 18th century for people to envisage control of the repetitive actions of a machine not with pegs on a rotating shaft (…) but rather, as in the case of Bouchon’s loom, with punch holes on a paper strip, a medium that took up less space and whose length was no longer limited.”

After Bouchon, Jean-Philippe Falcon and then Jacques Vaucanson developed automated machines for the weaving of designs, using external programs on a punched medium. Then in the early 19th century, Joseph-Marie Jacquard further perfected the automated loom and made its use in an industrial setting. As we will see, these looms would also have an influence on calculating machines.

2.2. Calculating machines

Calculating machines are automata that are quite different from those envisaged up till now in this contribution: in effect, they have the specificity of manipulating symbols. As such, it is relevant to make the distinction between the two varieties of memorized information: first, the data (which can be apprehended at different levels) and, second, the operations (more or less complex and whose sequencing constitutes a program) to be performed on those data.

How do we define a basic calculating machine? Marguin [38] provides the following definition: such a machine “includes an automatic mechanism to carry tens”. Let’s adopt this definition, while still noting that it is focused on the automation of the rules used for mental calculation, involving position numbering

[23] “(…) the arrangement of the punch holes was established as a function of the design that one wanted to reproduce on the cloth" (P. Rouillé [47]).

(regardless of the base used). We propose here that each basic[24] mechanism for adding numbers should be considered a form of programming (in the broad sense), since it incorporates calculation rules and is based on a given algorithm to process the various digits that make up the numbers to be added, this algorithm being more or less complex depending on the machine in question. The mechanism for carrying tens is particularly crucial in a mechanical machine that uses toothed gears to represent the digits. As Babbage [3] explains:

> "The process of addition requires for its completion that an apparatus should be attached to the wheels on which the sum is placed for the purpose of carrying any tens that may occur to the next highest digit. This mechanism may be constructed upon three different principles. It may be:
> - Successive in its operation
> - It may postpone its operations
> - It may anticipate operations".

One of the first and most famous calculating machines was invented in 1642 by Blaise Pascal. As Marguin [37] states:

> "It is surprising that automation of the carry operation was invented so late, at a time when machines in general and clock mechanisms in particular had reached a remarkable degree of perfection."

But we have to understand the mindset of the time:

> "Daring to invent a machine with the ability to automatically perform an operation that until then was strictly intellectual was akin to denying the divine nature of man" [38].

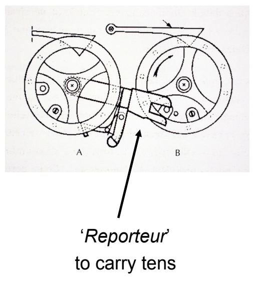


Fig. 4. Diagram of the carry mechanism, the basis for the Pascaline.

[24] This is not the case for the other basic mechanisms (as listed by Bromley [9]) of non-rudimentary calculating machines: the basic apparatus for storing and transferring numbers.

The "Pascaline" [19] is a simple adding machine. We consider here that it is based on an internal program that automates the carrying of digits, thanks to a series of carry mechanisms, each located between two toothed wheels. The central part in Figure 4 is the carry mechanism. It "is gradually lifted by the B wheel for the units. When it passes 10, it is suddenly released and falls (under its own weight), pushing the A wheel for the tens forward one position." In the Pascaline, tens are carried successively. Furthermore, it should be noted that in this machine, the storing and computing functions are not separate.

The next important steps in the history of calculating machines occurred in the 19th century and were the work of Charles Babbage. His "Difference Engine[25]," aimed at generating accurate and precise numerical tables, was "designed to calculate using repeated addition as in the method of finite difference."[26] There was not yet a separation of the memory and calculation functions. However, it did have two levels of programming: in addition to the internal program (analogous to the Pascaline program, but refined) for carrying the tens[27], it also included a higher level internal program (in the form of a "cam stack" [51]), which dictated the execution of one given sequence of additions, which always remains the same.

In 1834 [42], Babbage began to work on plans for another machine that he called the "Analytical Engine" (see Figure 5). As L. F. Menabrea stresses in his famed "memoir" of 1842, later translated into English and annotated by Ada Lovelace [36]:

> "Mr. Babbage has devoted some years to the realization of a gigantic idea. He proposed to himself nothing less than the construction of a machine capable of executing not merely arithmetical calculations, but even all those of analysis, if their laws are known."

The aim of this engine was thus to perform automatically any sequencing of operations (set out in writing in a kind of table, examples of which are provided in this same text by Menabrea). According to the classification proposed in the present contribution, the "Analytical Engine" had three levels of programming. Its automation was based on an external program[28] (in the form of a sequence of

---

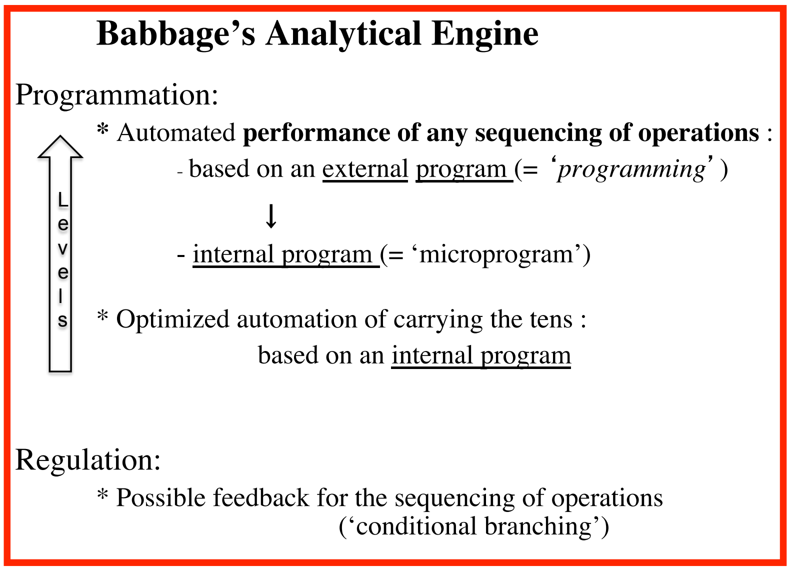
[25] Babbage worked on his "Difference Engine No. 1, from 1822 until 1833" (Bromley [9]). In 1846, he "worked on another machine which he called Difference Engine No. 2" (Bromley [9]).

[26] R. Horton [28] explains: "When calculating a polynomial using the method of finite differences and after the initial starting values have been calculated by hand, it is possible by using the method of differences to generate the rest of the required table using repeated addition."

[27] "In Difference Engine No. 1, that carry propagation was a sequential process" (Bromley [9]).

[28] "The highest level is a description of the user-level programs that exist for the machine. What Babbage thought could be done – exploiting the basic micro-programmed operations in order to carry out user-oriented tasks such as solving sets of simultaneous equations, or working out terms in recurrence relations" (Bromley [9]).

punched cards[29] inspired by those that Jacquard used in his automated looms), which itself was implemented by an internal program (a sort of micro-program, as it is now understood, in the form of a set of pegged cylinders[30]), which in turn controlled notably the sequencing of carry operations, performed by another internal program. We should note that Babbage equipped his Analytical Engine with sophisticated "anticipating"[31] carry mechanisms, located on three "carriage axes" ("with their peculiar apparatus"[32]).



Babbage's Analytical Engine

Programmation:

* Automated **performance of any sequencing of operations** :

- based on an external program (= '*programming*')

↓

- internal program (= 'microprogram')

* Optimized automation of carrying the tens :

based on an internal program

Regulation:

* Possible feedback for the sequencing of operations ('conditional branching')

Fig. 5. Definition of the automation of Babbage's Analytical Engine.

Moreover, the Analytical Engine had to be capable of making a decision on the basis of a result that it had obtained. In other words, it had to enable what is currently called "conditional branching"[33], which is another way of saying it had to

[29] "All the movements of the cards are relative to the present position, and there is no absolute addressing" (Bromley [9]).

[30] Babbage calls such a pegged cylinder a "barrel" with "studs". As noted by Bromley [8], "The barrel may be thought of as a microprogram store and a single vertical row of studs as a word of that store. […] In general, the barrel orders its own advance via several of the control levers. […] The barrel can […] order a transfer to another vertical up to seven positions either forward or backward relative to present one."

[31] According to the terms used by Babbage himself [4].

[32] According to the terms used by Babbage himself [3].

[33] "The idea of conditional control, which had been a convenient feature in Difference Engine No. 1, starts to look like an essential part of the design of the Analytical Engine because of division" (A. G. Bromley [9]).

be “regulated” (as defined above)[34]. We must also note another significant detail. Another innovation was needed in the Analytical Engine: the splitting of the functions into two organs, the “Mill” (or central processing unit) and the “Store” (with storage registers or cylinders)[35]. However, as A. G. Bromley [8] notes, “Babbage’s store has no structure accessible to the programmer”. Furthermore, this first “memory” was designed by Babbage for data only: the user’s program **always** remains on punched cards.

Babbage’s ingenious inventions were largely forgotten, and they are generally (see [50] and [10]) considered as having had practically no impact on the subsequent history of technology.

Let’s now move on to the next steps in the history of calculating machines, by looking at those machines that use electrical circuits. In this article, I will not examine the “Tabulating Machines,"[36] patented by Hermann Hollerith in 1884 and which also involve punched cards (but only to memorize data)[37]. These machines only performed a simple mechanization of counting[38].

We will therefore move on to the large numerical calculating machines of the 1940s. They contained at least two levels of programs. Above the internal program (responsible for carry operations and using electro-mechanical or electronic switches) was another program that dictated the sequencing of arithmetic operations. This latter program was either internal (and definitely fixed) (e.g. G. Stibitz’s BTL *Model 1*) or external. As explained by W. Aspray [1], on the basis of an unpublished text written by J. H. H. Goldstine and J. von Neumann [22], there were then two possible ways of making an [external[39]] program:

> “Either all of the connections are made prior to the computation, as in the ENIAC, where one first sets **switches** and **plug cables**, which effectively hard wires the instructions into the machine prior to the computation, or connections are established at the moment in the computation, when they are needed, as in the Harvard Mark I and the Bell Labs relay calculator, in which instructions are fed in as needed from an external **paper tape**. The advantage of the first approach is that, once entered, all of the instructions can be ***executed at electronic speed***; the second approach permits

[34] We should note that as far back as 1914 Torres y Quevedo [52] used the term ‘regulation’ to describe the action that today is called ‘conditional branching’.

[35] This separation was a consequence of the increase in the size of the machinery, which is itself related to Babbage’s determined and successful research in order to minimize the time needed for carry operations (see D. Swade [50] and A. G. Bromley [10]).

[36] On the subject of these machines and their inventor: see, for example, R. Ligonnière [35].

[37] These cards are not the medium for a program, but they do contain qualitative data.

[38] The carry operation was limited here to its simplest expression and was done using base 100. Hollerith himself [27] made the following description: “A number of mechanical counters are arranged in a suitable frame […] The face of each counter is […] provided with a dial divided into 100 parts and two hands, one counting units the other hundreds. […] A suitable carrying device is arranged so that at each complete revolution of the unit hand the hundred hand registers one …”

[39] This term is not used by W. Aspray [1].

> indefinitely ***long strings of instruction***s, does ***not require as much time in problem setup***, and can be implemented with less hardware"[40].

The first approach is, according to the vocabulary used in the present contribution, that of a manually modifiable program in situ, whereas the second is that of a program logically separate from the programmed device and which therefore can be changed manually.

3 Stored-program computers

At the end of World War II, the idea for a new design[41] emerged: the computer "that incorporated the best features of each approach, based on storing instructions as numbers in the computer's internal electronic memory [1]."[42] This idea was present in the draft written by von Neumann on June 30, 1945 [56] and it is unanimously considered fundamental. It implies that written numerical codes (also later called "machine language"[43]) are now needed to communicate with the calculating machine[44]. The practical consequence of this is that it enables the machine to modify its own programs.

However, in the opinion of many commentators, this idea is insufficient to fully define the concept currently identified by the term "stored program computer". To do this, according to B. Randell [45], two other crucial characteristics also have to be taken into account: "the ability of a program-controlled device to identify some information as a program and to switch to executing that program[45]" and "the fact that it is possible to calculate addresses, and so make dynamic decisions as to which data to use, or which instruction to obey". These two characteristics were absent from the above-mentioned text by von Neumann [56], but the seeds were already present in a slightly earlier text by Turing [54]. As B. E. Carpenter and R. W. Doran [13] explain:

> "The difference in attitude between von Neumann and Turing is evident in how their central processors deal with instructions. Early calculators had

[40] The italics and bold characters have been added by me.

[41] I am deliberately not entering here into the debate concerning the precise paternity and dating of this idea. See W. Aspray [2].

[42] The first computers still used punched strips (or cards), but the program was in the central memory during its execution.

[43] Such a language is defined by D. E. Knuth [30] as follows: "a language which directly governs a computer's actions, as it is interpreted by a computer's circuitry".

[44] M. Campbell-Kelly [12] underlines that « A small-scale experimental computer known as the miniature or baby machine first operated successfully on June 21 1948, and was the first EVAC-type electronic stored-program computer to be completed. [...] Binary programs were put, bit by bit, into the store using manual keys and a 'typewrite' of 32 pushbuttons, each button corresponding to one bit of the store line. »

[45] B. Randell [44] notes that "the connotation of the computer being able to [...] execute its own programs" is "surpassing the notion that Babbage had arrived at over a century earlier".

> programs on punched tapes through which they stepped, executing each instruction as it turned up. Von Neumann retained this attitude in his 1945 report, for he thought of the processor as receiving a stream of orders from consecutive memory locations. He never explicitly mentions an instruction address register – this was unnecessary, for the next instruction comes from the current position in the memory tape. Turing did have an instruction address register explicitly containing the instruction number IN, i.e. the position of the next instruction. [...] Turing's, concept of memory was much closer than von Neumann's to a random access addressable[46] device".

B. Randell [44] pursues this on the subject of these two drafts from 1945:

> "In both cases, however, what now seem very awkward techniques of program self-modification were needed for to make the machine calculate the addresses of variable – since neither the idea of index registers (B-lines as they were to be called at Manchester, where invented [and operational by April 1949] [47]) nor of indirection had yet arisen. However, once all these aspects of the stored-program concept had been provided, [...] machine-level programming essentially as we know it now had arrived."

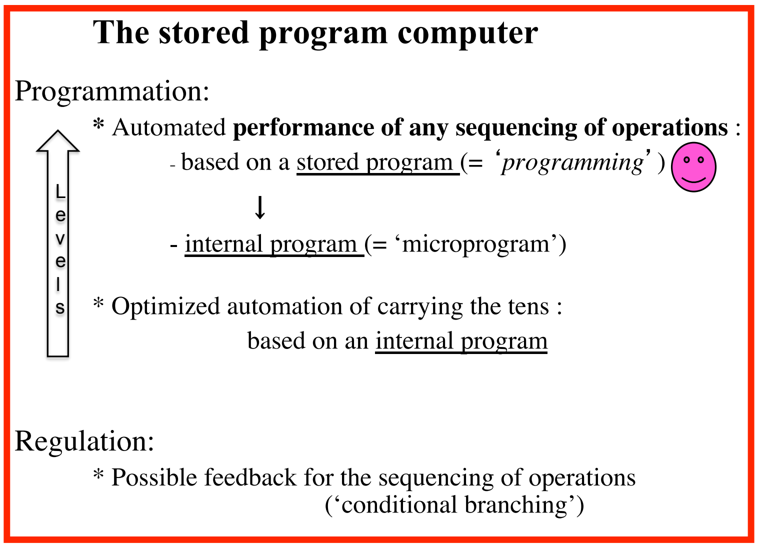
Furthermore, one of the applications of the possibility of modifying the program as described above, was not exploited before the 1950s. It is "the ability of one program to process another, treating it as data" [13]. However this "aspect" of modification is extremely important from a practical point of view: thanks to this, a programmed automaton can help people not only calculate, but also write its own programs.[48] Then, it is possible to invent and use higher-level languages (first, assembly languages and then, operational algorithmic or programming languages) and eventually to know nothing of the specific equipment characteristics of the computer being used when writing a program[49].

[46] A sufficiently large random addressable access store can therefore be considered a practical concretization of the Turing machine's infinite tape (see Turing [53]).

[47] See M. Campbell-Kelly [12].

[48] "*In the early day of automatic computing, programming was considered as some kind of art… With the advent of faster computers, however, the need for writing a program very soon became a nightmare and left no room for artistic feelings. The situation required immediate action in order to reduce the terrible burden. The relief came through the computers themselves: if computers were able to carry such a heavy load of computing, which before had taken years on a desk calculator, they certainly could also assist in writing programs. Indeed they could; it turned out that it was possible to write programs in a notation somewhere 'between' machine code and standard mathematical notation, which was then translated into correct machine code by the computer itself with the aid of a special translation program*" (H. Rutishauser [48]).

[49] The strict definition of a program for a computer scientist is therefore "*an expression of a computational method [i.e. algorithm] in a computer language*" (D. E. Knuth [30]).


The stored program computer

Programmation:

* Automated **performance of any sequencing of operations** :

- based on a stored program (= '*programming*')

↓

- internal program (= 'microprogram')

* Optimized automation of carrying the tens :

based on an internal program

Regulation:

* Possible feedback for the sequencing of operations ('conditional branching')

Fig. 6. Definition of the automation of a stored-program computer.

The concept of "software" emerged and was clearly differentiated from that of "hardware". As a consequence of storing the program in central memory, this fundamental advance for the *programmer* is combined with increased ease for the *builder*: "Treating instructions in the same way as data greatly simplifies both the memory hardware and the software of computer systems" [41]. Obviously, this is also beneficial for the computer *user,* who can go from one program to another instantaneously [41]. "The commercial implication is that computers can inherit ready-made software provided they are compatible with an existing instruction set" [41]. In the end, we can conclude, along with Patterson and Hennessy [41], that the invention of the stored-program concept "let the computing genie out of its bottle."

And to close out this long history of *automation*, I find it enlightening to stress that the stored program computer (see Figure 6) still, from a logical[50] point of view, takes into account both programming and regulation, identical to Babbage's Analytical Engine (see Figure 5), except for the fact that the highest-level program is, during its execution, stored in the central memory, instead of being external to the machine (on punched cards).

[50] From a physical point of view, we should note that the carry operation is performed in binary mode, which is technically quite easy.

4 Comments

Since Antiquity, humans have realized their ancestral dreams[51] of manufacturing automata. An overview of the long history of automation up to the stored program computer leads us to a few general thoughts that are essentially related not only to programming, but also to regulation.

We have seen that the programs as designed by humans involve a very precise sequencing that they have in mind. It should be stressed that this sequencing may consist either in a simple repetition or in a potentially more complex sequence. We have already mentioned the cases of simple repetition, for example, of a sound (alarm bell) or a unit of time (mechanical clock). We have also talked about various complex programmed sequences: for sounds (e.g. carillon melodies), movements (e.g. automata for astronomical clocks, such as bell strikers), letters (texts by Jaquet-Droz's Writer automaton), designs (weaving patterns) and, finally (when people dared at last to put intelligence into a machine), calculations (*Formula* of Babbage's Analytical Machine or, more generally, algorithms).

Furthermore, we have illustrated the fact that, to ensure that the machine executes their planned design, people first made fixed programs, then manually modifiable or replaceable ones and finally those that are 'fully' manageable (i.e. which can not only be replaced but also be written and then executed) by the machine itself (see 2nd column of Table 2). These three milestones (respectively called 'internal', 'external' and 'stored program computer' by Meinadier [39] (see 1st column of Table 2) have been shown to be fundamental, and their importance must be stressed. It should be added that, in parallel with this evolution towards greater flexibility and increased autonomy of the machine, we can observe, over time, an increasing dematerialization of the program support (possibly of the highest level), in other words, of the interface enabling humans to communicate with the machine. The major steps in this history involve the passage from raised supports to punched ones (which would seemingly go hand in hand with a mental evolution), then the passage to electrical circuits and finally the use of machine language (see 3rd column of Table 2). It is evident that the fundamental change, from the point of view of the history of the program's support, is the one that accompanies the passage to the so-called 'stored program': since we move from a support enabling a read-only machine with sequential access (or more generally, access relative to the current position[52]) to a support enabling writing and direct access to dynamically calculated[53] addresses (see 4th and 5th columns of Table 2). In the history of books, we can see an analogous evolution from the point of view of access: from the ancient roll of papyrus (which must be read as it is unrolled), we moved to books with pages (invented in the 1st century of the current era). It is

[51] For example, Book V 749 of the Iliad contains the following fragment "The gates of Olympus which *open self-bidden [αυτομαται]…* »

[52] A. G. Bromley ([8] and [9]) uses the term 'relative addressing' in this instance.

[53] It is only with addressable memory that the notion of an address for an instruction takes on its full meaning. The predecessors of stored program computers which were controlled by, for example punched tape, "could largely get by without addresses because of the fact that they could control the moving of the tape (q. v. Turing machines)" (B. Randell [45]).

striking to note that Turing [24][54] explicitly compared the central (magnetic) memory of Mark II to a book with pages:

> “It is as if information in the magnetic store were written in a book. In order to find any required piece of information it is necessary to open the book at the required page.”

Table 2. Suggested classification (and evolution) of programs (such as they are understood by the machine) and corresponding medium type.

Programs (as understood by the machine)		Corresponding Medium type		
Classes	Practical feature	Human/machine communication	Practical feature from the machine’s point of view	Machine access characteristics
1. ‘Internal’	Fixed	Using pegs / electrical circuits	Read-only	Sequential access
2. ‘External’				
2.1. In situ	Modifiable *manually*	Using pegs / electrical circuits	Read-only	Sequential access
2.2. Logically separate from the programmed device	Replaceable *manually*	Using pegs / <u>holes</u>	Read-only	Sequential access
3. ‘Stored program’	Fully manageable *automatically*	Using symbols (a ‘machine language’)	Writable	Random access (and calculated addresses)

The concept of regulation has been addressed only briefly in this contribution. However, let’s note that, when assimilated with conditional branching, it helps to increase the autonomy of a calculating machine, by enabling it to make decisions when the program is being run.

The brief and recent history of computing has now been placed within the longer history of automation. This has revealed deep-seated trends in the evolution of technology. I believe that this is likely to be of interest to a broad public.

[54] Cited by M. Campbell-Kelly [12].

Acknowledgements. I would like to thank Professors Brian Randell and Pascal Verroust (members of IFIP WG 9.7) and Luc de Brabandère (mathematician and philosopher) for reading the original manuscript and for helpful criticism and suggestions. I am also indebted to Professor Rita de Caluwe (*Universiteit Gent*) as well as to Philippe Slégers (member of the Association Campanaire Wallonne) and Serge Joris (administrator of the World Carillon Federation) for providing information about carillons. I would also like to acknowledge Richard Horton (Conservator Engineering & Metals, Babbage Project Engineer, Science Museum, London) and three professors at the University of Namur: Dr. Vincent Englebert, Dr. Ir. Laurent Schumacher and Dr. Wim Vanhoof for their help. Finally, I would like to thank Virtual Words Translations, which translated the original French text into English.

A preliminary, less detailed version of the present article was the subject of two presentations in 2011: one in French at Collège Belgique (slides are available on the website) and the other in English at the international conference "History and Philosphy of Computing" in Ghent (Belgium).

Figure 1: Basic drawing by G. Oestmann (with permission of the author) in G. Dohrn-van Rossum, p. 109, annotated M. d'Udekem-Gevers.
Figure 2: From A. Lehr *et al.* 1991, p. 89 (with the permission of M. Lehr).
Figure 3: From the Royal Carillon School "Jef Denyn" Mechelen (with the permission of K. Cosaert, director of the Royal Carillon School).
Figure 4: Drawing by Serge Picard, from Musée des arts et métiers (Paris), "Blaise Pascal", *Les carnets*, with the permission of Anne Chanteux (Head of the Multimedia Centre and of the network of technical museums and collections); annotation by M. d'Udekem-Gevers.

References

1. Aspray, W.: John von Neumann and the origins of modern computing. The MIT Press, Cambridge, London (1990)
2. Aspray, W.: The stored program concept. IEEE Spectrum, p. 51 (September 1990)
3. Babbage, C.: On the Mathematical Powers of the Calculating Engine. Unpublished Manuscript (1837), reprinted in: Randell B. (ed.): The Origins of Digital Computers: Selected Papers (3rd ed.), pp. 19--54. Springer, Berlin, Heidelberg, New York (1982)
4. Babbage, C.: Passages From the Life of a Philosopher. Longman, Green, Longman & Roberts, London (1864) (Chapter VIII, http://www.fourmilab.ch/babbage/lpae.html)
5. Braudel, F.: On History. University of Chicago Press, Chicago (1982)
6. Braunstein, Ph.: Préface. In Dohrn-van Rossum, G.: L'histoire de l'heure – L'horlogerie et l'organisation moderne du temps. pp. IX--XVII. Editions de la Maison des sciences de l'homme, Paris (1997)
7. Brennecke A.: A Classification scheme for Program Controlled Calculators, in: Rojas, R., Hashagen, U. (eds) The First Computers: History and Architecture, pp. 53--68. MIT Press, Cambridge, London (2000)
8. Bromley, A.G.: Charles Babbage's Analytical Engine, 1838. Annals of the History of Computing, 4(3), 196 --217 (1982)

9. Bromley, A.G.: The Evolution of Babbage's Calculating Engines. Annals of the History of Computing 9(2), 113--136 (1987)
10. Bromley, A.G.: Difference and Analytical Engines. In: Aspray, W. (ed.), Computing before Computers. Pp. 59—98. Iowa State University Press, Ames (1990)
11. Buchner, A., Rouillé, P.: Les instruments de musique mécanique. Gründ, Paris (1992)
12. Campbell-Kelly, M.: Programming the Mark I: Early Programming Activity of the University of Manchester. Annals of the History of Computing 2(2,) 130—168 (1980)
13. Carpenter, B.E., Doran, R.W.: The other Turing machine. Comp. J. 20(3), 269--279 (1977)
14. Daumas, M.: La petite mécanique et les origines de l'automatisme. In Daumas, M. (dir.): Histoire générale des techniques, Vol. 3, pp. 172--195. Presses universitaires de France, Paris (1969)
15. Devaux, P.: Automates et automatismes. Que sais-je? Presses universitaires de France, Paris (1942)
16. Dohrn-van Rossum, G.: L'histoire de l'heure – L'horlogerie et l'organisation moderne du temps. Editions de la Maison des sciences de l'homme, Paris (1997)
17. Drachmann, A.G.: The Mechanical Technology of Greek and Roman Antiquity. Munksgaard, Copenhagen (1963)
18. Drye, E. : Alexandrie, la naissance de la mécanique. La revue 20, 28--34, Musée des arts et métiers, Paris (1997)
19. Ellenberger, M., Collin, M.M.: La machine à calculer de Blaise Pascal. Nathan, Paris (1993)
20. Farré-Olivé, E.: A Medieval Catalan Clepsydra and Carillon. Antiquarian Horology 4(18), 371—380 (1989)
21. Gille, B. : Les mécaniciens grecs : la naissance de la technologie. Seuil, Paris, Collection Science ouverte (1980)
22. Goldstine, H.H., von Neumann, J.: On the principles of large scale computing machines. Unpublished text, reprinted in: Taub, A.H. (ed.) John von Neumann Collected Works, Volume V, pp. 1—32. Pergamon Press, New York (1961)
23. Grier, D.A.: The ENIAC, the Verb 'to Program' and the Emergence of Digital Computers. IEEE Annals of the History of Computing, 18(1), 51--55 (1996)
24. HB1, Programmers' Handbook for Manchester Electronic Computer Mark II, (Turing, A.M. ed.) March (?) (1951); Errata March 13 (1951)
25. Hill, D.R.: Arabic Water-Clocks. Sources & Studies in the History of Arabic-Islamic Science, History of Technology Series 4, University of Aleppo, Aleppo (1981)
26. Hill, D.R.: Technologie. In Rashed, R. (ed.) Histoire des sciences arabes, Le Seuil, Paris (1997)
27. Hollerith, H.: An Electric Tabulating System, reprinted in: Randell, B. (ed.) The Origins of Digital Computers, Selected Papers (3rd ed.), pp. 133—143, Springer, Berlin, Heidelberg, New York (1982)
28. Horton, R., e-mail to the author, January 20 (2012)
29. Jacomy, B.: Une histoire des techniques. Seuil, Paris (1990)
30. Knuth, D.E.: The Art of Computer Programming, Vol. 1, Fundamental Algorithms. Addison-Wesley, Boston, 2nd ed. (1975)
31. Lafitte J. : Réflexion sur la science des machines. Cahiers de la nouvelle journée 21, Librairie Bloud & Gay, Paris (1937)
32. Lehr, A.: De geschiedenis van het astronomisch kunstuurwerk. Zijn techniek en muziek. Nijhoff, Den Haag (1981)
33. Lehr, A.: Van Paardebel tot Speelklok (2e herziene druk). Europese Bibliotheek, Zaltbommel (1981)
34. Lehr, A., Truyen, W., Huybens, B.: Beiaardkunst in de Lage Landen. Drukkerij Lannoo, Tielt, (1991)

35. Ligonnière, R.: Préhistoire et histoire des ordinateurs. Robert Laffont, Paris (1987)
36. Lovelace, A.A.: Sketch of the Analytical Engine invented by Charles Babbage, by L.F. Menabrea, from the Bibliothèque Universelle de Genève, October, 1842, No. 82, with notes upon the Memoir by the Translator. (1843) http://www.fourmilab.ch/babbage/sketch.html#NoteB
37. Marguin, J. : Le reporteur et la naissance du calcul mécanique. La revue 2, pp. 26--32, Musée des arts et métiers, Paris (1993)
38. Marguin, J. : Histoire des instruments et machines à calculer. Hermann, Paris (1994)
39. Meinadier, J.P. : Structure et fonctionnement des ordinateurs. Larousse, Paris (1971)
40. Mesnage, P.: La construction horlogère. In : Daumas M. (dir.) Histoire générale des techniques. Vol. 2, pp. 289 –- 310. Presses universitaires de France, Paris (1965)
41. Patterson, D.A., Hennessy, J.L.: Computer Organization and Design (3rd ed.). Elsevier (2005)
42. Randell, B.: The Origins of Digital Computers: Selected Papers (3rd ed.). Springer, Berlin, Heidelberg, New York (1982)
43. Randell, B.: From Analytic Engine to Electronic Digital Computer: The Contributions of Ludgate, Torres, and Bush. Annals of the History of Computing, 4(4), 327--341 (1982)
44. Randell, B.: The Origins of Computer Programming. IEEE Annals of the History of Computing, 16(4), 6--14 (1994)
45. Randell, B., Comments to the author, June 12 (2012)
46. Rombouts, L.: Zingend brons – 500 jaar beiaardmuziek in de lage landen en de Niewe wereld. Davidsfonds, Leuven (2010)
47. Rouillé, P. : Trous de mémoire. La revue 2, 34--41, Musée des arts et métiers, Paris (1993)
48. Rutishauser, R.: Handbook for Automatic Computation. Vol. 1, Part a, Description of ALGOL 60, Bauer, F.L. et al. (eds.), Springer, Berlin, Heidelberg (1967)
49. Sablière, J.: De l'automate à l'automatisation. Gauthier Villars, Paris (1966)
50. Swade, D.D.: The Difference Engine: Charles Babbage and the Quest to Build the First Computer. Viking, New York (2001)
51. Swade, D.D.: Automatic Computation: Charles Babbage and Computational Method. The Rutherford Journal 3, http://www.rutherfordjournal.org/article030106.html (2010)
52. Torres y Quevedo, L.: Essais sur l'Automatique. Sa définition. Étendue théorique de ses applications", Revue de l'Académie des sciences de Madrid (1914), reprinted in: Randell, B. (ed.) The Origins of Digital Computers, Selected Papers (3rd ed.), pp. 89—107, Springer, Berlin, Heidelberg, New York (1982)
53. Turing A.M.: On computable numbers, with an application to the Entscheidungsproblem. Proc. Lond. Math. Soc. 42(2), 230--265 (1936) http://www.cs.virginia.edu/~robins/Turing_Paper_1936.pdf
54. Turing, A.M.: Proposals for Development in the Mathematics Division of an Automatic Computing Engine (ACE), Report E882, Executive Committee, NPL (1945), reprinted with foreword by Davies D.W. as NPL report, Com. Sci. 57, April 1972
55. Unesco, http://whc.unesco.org/en/list/943/video
56. von Neumann, J.: First Draft of a Report on the EDVAC. Contract no. w-670-ord-4926. Techn. Rep., Moore School of Electrical Engineering, University of Pennsylvania, Philadelphia, PA (1945) www.virtualtravelog.net/ entries/2003-08-TheFirstDraft.pdf
57. Youtube, http://www.youtube.com/watch?v=Pd_21_pfSRo&feature =related