Reflections of Computing Experiences in a Steel Factory in the Early 1960s

Pertti Järvinen

Department of Computer Sciences, FIN-33014 University of Tampere, Finland; pj@cs.uta.fi

Abstract: We can best see many things from a historical perspective. What were the first pioneers doing in the information technology departments of Finnish manufacturing companies? In early 1960s, I had a special chance to work in a steel industry that had long traditions to use rather advanced tools and methods to intensify their productivity. The first computer in our company had such novel properties as movable disk packs making a direct access of stored data possible. In this paper, we describe the following issues and innovations in some depth. These include (a) transitioning from the punched card machines to a new computer era, (b) using advanced programming language to intensify production of new computer software, (c) drawing pictures by using a line printer, (d) supporting steel making with mathematical software, (e) storing executable programs to the disk memory and calling and moving them from there to the core memory for running, and (f) building a simple report generator. I will also pay attention to the breakthrough in those innovations and in this way demonstrate how some computing solutions were growing at that time.

Keywords: Report generator, Virtual memory, Path dependency

1. Introduction

Mason et al. [18] said that historical analyses broaden our understanding of those processes by which information technology is introduced into organizations and of the forces that shape its use. They use the expression "dominant design" to describe a new configuration of an organization's technology, strategy, and structure. A dominant design is manifested in several ways: a new organizational infrastructure, new functionality, new products, new services, new production functions, or new cost structures. By changing the basis of competition in the industry, a firm that institutes a dominant design secures an initial competitive edge. According to Mason et al. [18] the Information Systems (IS) research literature contains very few examples of historical analyses.

This paper describes some key issues and a few computing solutions to shed light on a pioneer manufacturing company and its first years to utilize a computer. According to Mason et al. [19] historical research offers perspectives on phenomena that are unavailable by any other methodological means. They reflect the cultural circumstances and ideological assumptions that underlie phenomena and the role played by key decision makers together with long-term economic,

social, and political forces in creating them. Based on my recent efforts at collecting various research methods [13], I can say that a historical method is a rarity in the methodological information systems literature.

The rest of the paper consists of the following topics: introduction to the computer usage, FORTRAN programs for administrative purposes, visualizing some reports, supporting the making of stainless steel by computer, towards a primitive operating system and the computer-aided development of reporting software.

2. Transitioning from the Punched Card Machines to the Computer Era

My description concerns the OVAKO steel factory at Imatra in Finland. In 1963, the company bought its first computer, an IBM 1401 with a punched card reader, line printer, operator console, and four discs units with movable disk packs. The latter were rather new. The IBM marketing men and consultants said that it was then the second newest computer with the same sort in Europe. To relate our hardware with some other installations at the same period, I refer to McKenney et al. [20] who mentioned IBM 1401 in their famous case of Bank of America, where they describe the way they used magnetic tapes for storing bank accounts at that bank. The Bank of America nicely describes both the path dependency [5] and the importance of the selection decision in transitions from the earlier hardware generation to the next generation.

In 1963, I began working with three other IT colleagues. I consider those colleagues as IT experts because they were the only people who could design and execute computer programs. My colleagues, because of their economic education, implemented such administrative applications as payroll, invoicing, order processing, bookkeeping, and budgeting. The company hired me because my scores in the IBM programmer test were acceptably high. My job concerned industrial applications, because as a mathematician, I also had some knowledge of physics and chemistry. My working period started June 1st, about three months before the installation of IBM 1401. I participated in the FORTRAN programming course organized by IBM.

An important observation was that the earlier punched card experts were not able to move easily to the computer time, although our computer used punched cards as input media. The stored program and especially disk memory were quite strange to punched card experts. For example, the chief of the earlier punched card department had designed a new payroll system for a computer, and he based his sketch of the new system upon seventeen sum-cards. The latter meant that the intermediate results in a particular phase of wage calculation process were stored to a new card (sum-card) which was thereafter punched as an intermediate output and later read as an intermediate input for the next phase of that calculation. This example demonstrates that “when novelty increases, the path-dependent nature of knowledge has negative effects because the common knowledge used in the past may not have the capacity to represent novelties now present” [5].

3. FORTRAN Programs for Administrative Purposes

In different places of the factory, there were certain people (more than thirty in continuous three-shift-work) for performing production inspection (PI). Those PI people recorded every event and state-transition considered important. Based on their data, they manually generated different kinds of production and deviation reports.

The new computer was very expensive. The local management wished to produce visible results as soon as possible. For programming, there were two compilers available, one for an assembly language (called Autocoder) and another one for the FORTRAN language, mainly intended for mathematical calculations. The expressions for input and output in FORTRAN were very restricted and simple, but the language itself was quite easy to learn. Although with Autocoder language it was possible to read all kinds of special markings punched on cards, and although in Autocoder there were especially a wide range of expressions for printed output, it demanded a rather long time to become familiar with all the features of Autocoder. Therefore, at the beginning of my job as a programmer I selected FORTRAN, which I used in my programming efforts. My first task was to develop the computer programs that would produce similar reports on production and exceptional events as was earlier done manually. About one year later, I changed those FORTRAN programs to the Autocoder programs with better output quality.

4. Drawing by Using a Line Printer

In a steel making process, they cast molten steel into moulds and after solidifying, they removed the ingots and set down to thermal ingot furnaces for two to four hours before lifting them up for rolling. The number of thermal ingot furnaces was about five. The company described their "used capacity" as a percent share for each hour each day as a figure, that earlier one worker drew manually. The production inspection people recorded all the processing phases of ingots and in this way produced the raw data for the drawing. They produced a figure for the used capacity of all the thermal ingot furnaces once a day.

To produce the same figure with computer was not a trivial task, although there were times by the clock of ingots both when set down and when lifted up. Some ingots were not immediately placed into the ingot heating furnaces but they were allowed to cool completely. Later, they would take them into the ingot heating furnace. Their heating would then require many hours and the heating period could continue from one calendar day to the next. They would have to reconstruct the development of the ingot heating furnace history of the previous day at the beginning of each day. The consideration of clock times required a special care in the program. The local manager, the main user of the figure, gave strong criticism based on bad appearance in the first versions of those figures.

I later saw how the Cascade project [1] built a graph production system. Its purpose was to produce a hardcopy version of information analysis documentation in a proper format. Documentation consisted of tables, matrices, and graphs.

5. Manufacturing Stainless Steel

The main part of steel production from the factory was for different construction steels and for railway building as rails and base plates. Although small, the relative portion of stainless steel was increasing. The main part of stainless steel had type 18/8 or 18/10; it means that percent of chromium (a rather expensive raw material) is 18% while the percent of nickel is 8% or 10%. In addition, the acid sustainable steel contained a small amount of molybdenum about 3%, a very expensive raw material. We now describe the way I utilized computer calculations in the manufacturing of stainless steel.

In the production of steel, the starting point is scrap. Occasionally, they use a small portion of iron ore. They first place the scrap into a furnace and with the use of electricity, the scrap melts. From the melted batch, they do a chemical analysis. In the factory, they built a very efficient arrangement with pneumatic mail for taking this analysis in the chemical laboratory. It took only two minutes. After knowing the content of the initial batch, they add suitable amounts of different additional materials (e.g. Ni, FeCr, FeMo, SiCr and CaSi for slack reduction [15]) to the initial batch. Before adding new materials, and if necessary, the company removes the harmful material.

In the process of making stainless steel, the company could obtain chromium and nickel from the initial batch or from different additional materials that might contain different contents of chromium and nickel. The experts of stainless steel knew that all the chromium and nickel that existed in the additional materials would transfer to the final stainless steel. This fact helped in the calculations because it influenced the amount of additional materials added into the initial batch. We could mathematically describe this problem as a system of seven equations.

After discussion with the technical supervisor of the smelting department, I had developed a computer program to solve the system of seven equations. In practice, after doing the chemical analysis of the initial batch and making the transfer to the furnace, the supervisor made a telephone call to the computer room to report the results of the analysis. The operator then entered those analysis data into my program by using the computer console. It took about thirty seconds to calculate and print the result back to the console. The message was something as, "Please add m kg of material A, n kg of material B, etc." They kept the telephone line open and after the results were ready, the operator told the result to the supervisor.

During the first series of stainless steel making, they produced about 25 smelting charges. The technical boss was at the smelting plant and I was in the computer room. Manufacturing of one batch took about 4 hours to make, so the first series took more than one week to make. Sometimes both the technical boss

and I had to wake up in the middle of night for taking care of this calculation. Nevertheless, I was happy because all the batches made went inside of the very tight limits, i.e. no smelting batch was a scrap.

In steel industry Fabian [8, 9] rather early in the 1950s applied linear programming to all stages of steel making – from coal and ore through finished products. Some sub model is close to the one I had prepared. Fabian's largest process model covers the whole production. One expert in operations research, Bo Nyholm, encouraged me to consider a similar model, but the complexity of product assortment with many production paths on the one hand and the shortage of computer memory and suitable program package on the other hand prevented realization of our attempt.

6. Systems and Applications Programs: From Punched Cards to Disk Storage

The sorting program produced by IBM for our IBM 1401 computer filled two cases, about 4000 punched cards. It would take many minutes to load the cards from the card reader into the core memory. When I followed the reading process, I found that the card reader was reading about half of the cards at the steady rate. After a more careful study, I found that those cards belonged to a sub program intended for sorting data on magnetic tapes, but we did not have any tape unit. I removed those cards, and thereafter the sorting program functioned correctly in our context with four disc units.

The reduced set of the punched cards belonging to our sorting program was still rather large. Its input from the card reader took a long time. Therefore, I continued my studies to shorten the loading time. I had an idea to locate the sorting program to a disc unit. After recording it in the disc unit, the sorting program, I could move the sorting program to the core memory by a short and simple call from the console. The operators were happy, because they saved time in two respects. The loading time was then shorter than before, and the loading always succeeded which was not always true with punched cards. After many repeated usage times they got worse and created a jam in the card reader.

After my first successful trial to utilize disk memory for the sorting program, I applied the same idea to my application programs. I recorded them into the disk memory and I could call them by name from the operators' console. At the same time, I eliminated the so-called IOCS (Input Output Control System) cards from the front of the program cards. Later, I understood that those IOCS cards were the beginning of an operating system, and my arrangement was in fact a simple operating system.

The next step forward was to avoid the upper limit of the core memory of 12K. I compiled my large program as components and located every component to the disk memory. When executing the large program, I read, or my main program read, one component after another from the disk to the core memory. In this way, I could prepare about 100K program and execute it without any problem.

Later, I understood that I had applied an idea of the virtual memory and its static (pre-planned) approach to storage allocation [6].

7. A Simple Report Generator

The first programming tasks were to read a set of punched cards and to write a report. Later, a major part of report requests concerned data in different files stored on disks. The structure of a reporting program was somewhat similar. This created a desire to automate my programming efforts. Hence, I developed a special program for reporting purposes. Later, I recognized that I in fact developed a simple report generator.

It was possible to give the name of a sorted file as a parameter for my report generator. In addition, a user would give the names of data items moved from the file into the report. The order of the data items determined the presentation order of the output form. One could compute a certain output item from stored data items. The way to perform those computations could have a representation as a "mathematical" formula allowing addition, subtraction, multiplication and division operations in addition to brackets. My report program interpreted and evaluated the expression in run time and produced an output to a certain location on the report. We could count the general or total sums and the intermediate sums. After leaving the steel factory 1967, I heard [21] that they used my report generator for many years to do various kinds of tasks; it also functioned as a simple spreadsheet.

The most demanding task in the development of the report generator was an evaluation of the mathematical expression. Later I understood that I in fact solved the problem of the way to transform recursions into iterations ([16], p. 37).

Our report generator differed from ordinary application programs in many ways because it had interpretive flexibility. Doherty et al. [7] define interpretive flexibility as the capacity of a specific technology to sustain divergent opinions. They have also found that

> *"… all technologies offer a range of functions and features that will facilitate some activities, while inhibiting others. Based upon the evidence from the empirical study, it became clear that there were upper and lower limits with respect to the functions that the system supported, and that these boundaries constrained the way in which the technology could be interpreted. More specifically, it was possible to discern, what we have termed, 'enforcing constraints' that make certain elements of the system's functionality mandatory. At the opposite end of the spectrum, it was also possible to identify 'proscribing constraints' that delineate the functions that do not exist, or for whatever reason cannot be used."*

Because our report generator was more flexible than any single report program, its interpretive flexibility was much larger than any report program, or it cannot be included into the domain of the interpretive flexibility concept at all.

My report generator was the first step in the sequence of my trials in computer-aided design of information systems. The next step in early 1980s was a simple file generator that demonstrated how it was easier to support human

memory by computing systems than human data processing [12]. My group's last step in late 1980s was to develop an application generator, Genera [14]. It was similar to an interpreter capable of analyzing and executing Pascal-type specifications. We could quickly generate some twenty to thirty applications with Genera until the commercial application generators made it obsolete.

8. Discussion

In this work, we demonstrated that the transition from punched card machines to a computer made big changes in storing data. Computers can support people's memory with storage media allowing quick storing and retrieval properties. We also showed how the third generation programming language, even such one intended to mathematical calculations, could intensify software production compared with the traditional solution of that time, an assembly language. To eliminate manual work I used the computer to draw some figures. The only device for that purpose was the line printer, not very suitable for such a task.

In addition to those primitive and easy computer applications, we also used a computer for some demanding tasks. Firstly, to solve a set of seven equations is impossible with paper and pencil at the blast furnace with noise and heat. In this task, the computer is superior compared with a human being. We then also demonstrated networking in the germinal form. Secondly, we utilized the disk memory of our computer to improve operators' work by storing our programs to disk and calling them into running from there. Our advances are clearly steps towards modern operating systems. Thirdly, we developed a report generator with spreadsheet facilities. In our construction, we needed knowledge later theorized in connection with compilers.

Gaines and Shaw [10] were a few of the first researchers who performed a historical analysis of hardware/software, state of artificial intelligence and state of human-computer interaction. They structured their analysis into eight years periods based on new generations of IBM big computers. They especially studied consecutive phases of the development of human-computer interaction. They used the model of the six eras as follows:

> *"Each technology ... seems to follow a course in which a breakthrough leads to successive eras: first replications in which the breakthrough results are copied widely; second empiricism in which pragmatic rules for good design are generated from experience; third theory in which the increasing number of pragmatic rules leads to the development of deeper principles that generate them; fourth automation in design based on the theory; finally leading to an era of maturity and mass production based on the automation and resulting in a rapid cost decline."*

By referring to the model of six eras I can say that my innovations or breakthroughs can be found in the computer literature, but were not available at our company. Few people (if any) in Finland then knew those innovations and their design concepts [22]. Knowledge and algorithms concerning construction of compilers [2, 3] and operating systems [4] were already published in the scientific literature in the 1960s and early 1970s. But the March and Smith's seminal article

of design research [17] was published as late as 1995. That article outlines what is design science in information systems, and what are the potential results. March and Smith first wrote that in addition to new design knowledge the new instantiations also can be accepted as research outcomes. Hevner et al. [11] later supported that claim.

I know that this paper has its specific limitation, i.e., it is based on personal memories. But I am happy that I could send the draft to two of my colleagues from that time (Managers Kostamo [15] and Ruotsi [21]) for verification. They both confirmed my text. Another limitation is that my contributions are based on one case only. But to my mind, it is not a very severe shortcoming, because my contributions belong to design research. Instead of providing mathematical or statistical evidence for my tentative contributions, which is normal in mathematics or social and natural sciences, I "proved" my contributions by demonstration. For example, it was possible to satisfy most of the report requests by my report generator.

References

[1] Aanstad P., G. Skylstad G. and A. Sølvberg A., 1971, Cascade – a computer-based documentation system, *Computer-Aided Information Systems Analysis and Design*, Bubenko J., Langefors B. and Sølvberg A. (Studentlitteratur: Lund), pp. 93-118.

[2] Aho A.V. and Ullman J.D., 1972, *The Theory of Parsing, Translation and Compiling*, Vol I: Parsing, (Prentice-Hall: Englewood Cliffs).

[3] Aho A.V. and Ullman J.D., 1973, *The Theory of Parsing, Translation and Compiling*, Vol II: Compiling, (Prentice-Hall: Englewood Cliffs).

[4] Brinch Hansen P., 1973, *Operating System Principles*, (Prentice Hall: Englewood Cliffs).

[5] Cohen W.M. and Levinthal D.A., Absorptive capacity: A new perspective on learning and innovation, *Administrative Science Quarterly*. Volume 35, Number 1, pp. 128-152 (1990).

[6] Denning P., Virtual memory, *Computing Surveys*. Volume 2, Number 3, pp. 153-189 (1970).

[7] Doherty N.F., Coombs C.R. and Loan-Clarke J., A re-conceptualization of the interpretive flexibility of information technologies: Redressing the balance between the social and the technical, *European Journal of Information Systems*. Volume 15, Number 6, pp. 569-582 (2006).

[8] Fabian T., A linear programming model of integrated iron and steel production, *Management Science*. Volume 4, Number 4, pp. 415-449 (1958).

[9] Fabian T. (1963), Process analysis of the U.S. iron and steel industry *Proceedings of a Conference sponsored by the Cowles Foundation for Research in Economics at Yale University* (April 24-26, 1961, Manne A.S. and Markowitz H.M. (Wiley: New York), pp. 237-263. (see http://cowles.econ.yale.edu/P/cm/m18/m18-09.pdf)

[10] Gaines B.R. and Shaw M.L.G., From timesharing to the sixth generation: the development of human-computer interaction. Part I, *International Journal of Man-Machine Studies*. Volume 24, Number 1, pp. 1-24 (1986).

[11] Hevner A.R., March S.T., Park J. and Ram S., Design science in information systems research, *MIS Quarterly*. Volume 28, Number 1, pp. 75-105 (2004).

[12] Järvinen P. 1983, *The ABC System – A Collection of Research Articles*, Report A112, (Department of Mathematical Sciences, University of Tampere: Tampere).

[13] Järvinen P. 2004, *On Research Methods*, (Opinpajan kirja: Tampere, Finland).

[14] Järvinen P., Kiukkonen P., Koskivirta M. and Välimäki H. 1987, *How flexible software could support learning*?, presented in Social implications of home interactive telematics (HIT) conference, June 24-27, 1987, Amsterdam, 16 p.
[15] Kostamo P. Manager of Steel Department 2007, interview 19.2.2007.
[16] Kurki-Suonio R. 1971, *Computability and Formal Languages*, (Studentlitteratur: Lund).
[17] March S.T. and Smith G.F., Design and natural science research on information technology, *Decision Support Systems*. Volume 15, Number 4, pp. 251-266 (1995).
[18] Mason R.O., McKenney J.L. and Copeland D.G., Developing an historical tradition in MIS research, *MIS Quarterly*. Volume 21, Number 3, pp. 257-278 (1997).
[19] Mason R.O., McKenney J.L. and Copeland D.G., An historical method for MIS research: Steps and assumptions, *MIS Quarterly*. Volume 21, Number 3, pp. 307-320 (1997).
[20] McKenney J.L., Mason R.O. and Copeland D.G., Bank of America: The crest and trough of technological leadership, *MIS Quarterly*. Volume 21, Number 3, pp. 321-353 (1997).
[21] Ruotsi E. Manager of Production Inspection Department (2006), interview 13.11.2006.
[22] van Aken J.E., Management research based on the paradigm of the design sciences: The quest for field-tested and grounded technological rules, *Journal of Management Studies*. Volume 41, Number 2, pp. 219-246 (2004).