[CP-02-016] On the Origins of Computing and GIS&T: Part I, A Computer Systems Perspective

Abstract

This paper describes the evolutionary path of hardware systems and the hardware-software interfaces that were used for GIS&T development during its “childhood”, the era from approximately the late 1960s to the mid-1980s. The article is structured using a conceptualization that developments occurred during this period in three overlapping epochs that have distinctive modes of interactivity and user control: mainframes, minicomputers and workstations. The earliest GIS&T applications were developed using expensive mainframe computer systems, usually manufactured by IBM. These mainframes typically had memory measured in kilobytes and operated in batch mode with jobs submitted using punched cards as input. Many such systems used an obscure job control language with a rigid syntax. FORTRAN was the predominant language used for GIS&T software development. Technological developments, and associated cost reductions, led to the diffusion of minicomputers and a shift away from IBM. Further developments led to the widespread adoption of single user workstations that initially used commodity processors and later switched to reduced instruction set chips. Many minicomputers and workstations ran some variant of the UNIX operating system, which substantially improved user interactivity.

Keywords: Job Control Language, Mainframe, Minicomputer, UNIX Workstation

Author & citation

Armstrong, M. P. (2019). On the Origins of Computing and GIST: Part I, A Computer Systems Perspective. The Geographic Information Science & Technology Body of Knowledge (2nd Quarter 2019 Edition), John P. Wilson (ed.). DOI: 10.22224/gistbok/2019.2.14.

Explanation

1. Introduction

A literature on the history of GIST is slowly, but surely, emerging. Early landmarks include an article by Dueker (1979) that catalogs reasons for early system failures, and an accessible book that captures the technological forces that were re-shaping cartography and GIST in the mid-1980s (Monmonier, 1985). Coppock and Rhind (1991) add a comprehensive look with a British “flavour” and divide GIS history into four main periods to structure their discussion. Major works by Foresman (1998) and Chrisman (2006) largely, but not exclusively, focus on the development of GIST in different institutional contexts or in different thematic domains. For example, Dobson and Durfee (1998) describe advances at Oak Ridge National Laboratory, and Chrisman (2006) describes developments at

Harvard, while Estes and Jensen (1998) provide a perspective on interactions between remote sensing and raster GIS. Ben and Sue Niemann conducted a number of interviews with GIS innovators that were published in Geo Info Systems in the late 1990s (e.g., Niemann & Niemann, 1999). More recent pieces adopt a more traditional chronological basis in providing a condensed version of GIST evolutionary processes (Waters, 2017) and a re-envisioning, sensu lato Robert Frost, of what might have been, had GIS pursued a more spherical rather than a projectional road of development (Goodchild, 2018).

This paper adopts a different perspective and focusses on the hardware and the hardware-software interfaces that existed during the “childhood” of GIS, defined approximately as the late-1960s to the mid-1980s, a span that roughly approximates parts of Coppock and Rhind’s first and second periods. While software and data are clearly the most important part of modern systems (hardware has receded into the background), that was not the case in the early days of GIST because the technology extant at those times placed a limit on what could feasibly be accomplished. Those limitations, in turn, caused programmers to innovate in order to wring more capability out of the available technology. One need only to examine the proceedings of a key GIST conference to see significant coverage of hardware issues at Auto-Carto III (January, 1978) and the effective absence of such discussions at Auto-Carto 7 (March, 1985).

While this computer systems focus may strike some readers as overly deterministic, or Whigish, there are plenty of setbacks to report. GIS had a long childhood, it developed slowly, in many places, with many fits and starts. Accordingly, the emphasis placed on limitations (such as capacity and latency) that were imposed as a consequence of available technology is appropriate. These early limitations also affected how GIST software continued to evolve into the current era. The term GIST is also used in a broad sense, ranging from comprehensive software products that were developed to capture, organize and manage, analyze and display geospatial information, to other more focused software intended mainly to produce maps and/or analyze data sets using a specific method. Finally, I want to emphasize that while this paper is grounded in widely accessible publications, much of the narrative is also based on personal experiences and having access to a “gray” or fugitive literature.

2. Computer Epochs

The history of GIST and computing is decidedly non-linear and as a consequence it is difficult to characterize accurately in a sequential narrative. For convenience, I will treat developments by placing them into three epochs (mainframes, minicomputers and workstations), with each epoch having defining computer system characteristics related to interactivity, latency and user control. Consequently, even though there is overlap among the epochs, each had a more or less distinctive “feel” to the user.

2.1 The Mainframe

Mainframe computers were expensive, large-scale systems with near state-of-the-art technology that could process large data sets and handle multiple jobs. Large, of course, means large for that time; in the current era of “big data” we routinely encounter data collections that were nearly unimaginable in the late 1970s.

IBM was long a dominant force in the deployment of business mainframe systems. Other companies competed for leftovers, though there were differences between the business world where conventional wisdom was that “nobody ever got fired for buying IBM” and university computing where the competition was more heated and a more diverse collection of hardware was often encountered as a consequence of the application of criteria other than the business-case bottom line (e.g., numerical precision).

While IBM had success with several earlier models, most notably the relatively small system 1401 and the larger 7090 series, the introduction of the System/360 series was a game changer as its single architecture spanned a very wide range of capabilities and markets. IBM even built migration paths with backward compatibility between the 1401 and low-end 360/30 models and between the scientific computing oriented 7090 series and 360/65, 360/75 and particularly 360/91 models. IBM also introduced new lines of peripheral devices with the unveiling of the System/360 series in 1964 (see Ceruzzi, 2003, Chapter 5).

In 1980, the U.S. Geological Survey received a three-volume report (“Computer Software for Spatial Data Handling”) amounting to more than 1,000 pages, that was produced under contract and published by the International Geographical Union Commission on Geographical Data Sensing and Processing (Marble, 1980). Duane Marble served as the overall editor of the three reports, though each volume had different coordinators: Volume 1 (Full Geographic Information Systems) was coordinated by Hugh Calkins, Volume 2 (Data Manipulation Programs) by Donna Peuquet, and Volume 3 (Cartography and Graphics) by Kurt Brassel and Micheal Wasilenko. The abstract concisely describes the contents of these volumes, indicating that they contain

“standardized descriptions of over seven hundred computer programs and systems for spatial data handling. Each program module is described, in so far as possible, in terms of function, machine requirements, and programming language.”

The 1980 report was an updated version of a draft report that was compiled in 1976 (Marble, 1978). Substantial growth can be observed by examining a few of the headings that are held in common between the reports. Assuming that there are similar protocols in place for data collection, the number of full geographical information systems, for example, increased from 22 to 85. It is also possible to get a glimpse at the software and hardware ecosystems that predominated at the time of the 1980 report. Table 1 enumerates a raw count of the instances of particular types of computer manufacturers, though many entries employed more than one computer (e.g., a mainframe for geospatial analyses and a minicomputer to handle interactive digitizing tasks). It should also be noted that a “count” for a single-purpose program (e.g., to computer the area of a polygon) is treated identically to a full state-wide GIS.

Table 1. Frequency count of computer manufacturers in the appendix to Marble (1980).

Manufacturer	Predominant System	Count
IBM	System/360	219
IBM	System/370	131
IBM Total	360 + 370	350

Manufacturer	Predominant System	Count
CDC	Cyber and 6000 series	195
Univac	1100 series	150
DEC	PDP and 10 series	84
Data General	Eclipse and Nova	18
Burroughs	6000 series	14
Honeywell		13

It is clear that IBM dominated geospatial applications; System/370 was an upgrade to the System/360 series and maintained substantial compatibility with it. The table also captures a point in time when GIST was transitioning from mainframes to minicomputers (e.g., DEC PDP and Data General), described in greater detail below.

While IBM systems grew to dominate both the general and geospatial markets, one main operating system (OS/MVT, with MVT meaning multiprogramming with a variable number of tasks) vexed many novices given the arcane and seemingly bizarre vocabulary and syntax of its OS/JCL (job control language). Though documentation was provided by IBM, it was not user-friendly in any meaningful way. Indeed many beginners complained about needing to join the “International Brotherhood of Magicians”, and simply copied JCL instructions with no comprehension of what each instruction actually accomplished.

As a consequence of this complexity, university computer centers often provided access to locally written System/360 documentation and most created customized JCL handouts that pertained specifically to the setup used at that institution. Other attempts were made to provide a more generalized overview of JCL (see, e.g., Rattenbury, 1973). Here is a simple example of a short FORTRAN program with accompanying JCL that reads a single value from a card-image record and then punches that value on a new card:

```
//PUNCHIT JOB
/*ID PS=1046
/*ID CODE=IOWA
// EXEC FORTLDGO,PARM.FORT='DECK'
//FORT.SYSIN DD *
          READ(5,10)PI
10        FORMAT(1X,F7.5)
          WRITE(6,10)PI
          END
/*
//GO.SYSIN DD *
3.14159
/*
```

In this JCL example, the first line is a user-specified job name (PUNCHIT). The two lines that begin with /*ID provide accounting information. The subsequent line (// EXEC) indicates that the job should be compiled, loaded, and executed using a FORTRAN compiler. The PARM parameter has nothing to do with cheese, but, rather, lets the machine know that a card deck will be produced. The subsequent line beginning with //FORT informs the machine that FORTRAN source statements will follow until the /* is encountered. If the program compiles successfully, it is loaded and executed with the data found after the //GO.SYSIN DD * statement (SYSIN is system input and DD is data definition). Reading the input is terminated with the final /* statement. It should be noted that there must be a space between the // and EXEC in line 4 (this is a space for an optional step name) and there cannot be a space between // and FORT on the subsequent line. These types of syntactical minutiae made people crazy but also provided a basis of employment for many who mastered the JCL dark arts. JCL got particularly dicey when data and programs were stored on magnetic tapes and parameters describing variable record lengths and blocking factors (how records are grouped together to enhance access efficiency) needed to be specified. OS/MVT also provided a time sharing option for interactive editing and batch job submission and output.

2.2 Batch processing and punched cards

Batch processing was a common mode of interaction with mainframes in the 1970s. Batch processing effectively means placing a job into a queue and waiting for output, where the wait duration was a function of special requests (e.g., tape mounting) and the number of jobs already in the queue, as well as the amount of memory and CPU time requested via a JCL statement. For example, if your job required considerable amounts of memory or CPU time, it meant other jobs could not run and your job would be given a lower priority. Though batch jobs could be created from disk or tape, punched cards (aka Hollerith and IBM cards) were the most common mode of program and data input in university computer centers (Figure 1). Cards were important because they were free, while storing programs and data on disk required an expenditure of funds. Alternatively, tapes containing programs and data could be stored in a centralized facility and requested on a JCL statement, but that required an operator to hold the job, locate the tape (usually on a particular rack) and mount it. Though cards provided cost-effective analog storage, the downside was that they were bulky, and order dependent. As a consequence, card decks often had dark diagonal stripes applied by a “magic marker” to permit the restoration of its sequencing. Apocryphal stories abounded about some poor schlimazel who spent considerable time keypunching cards only to have the deck spill haphazardly.

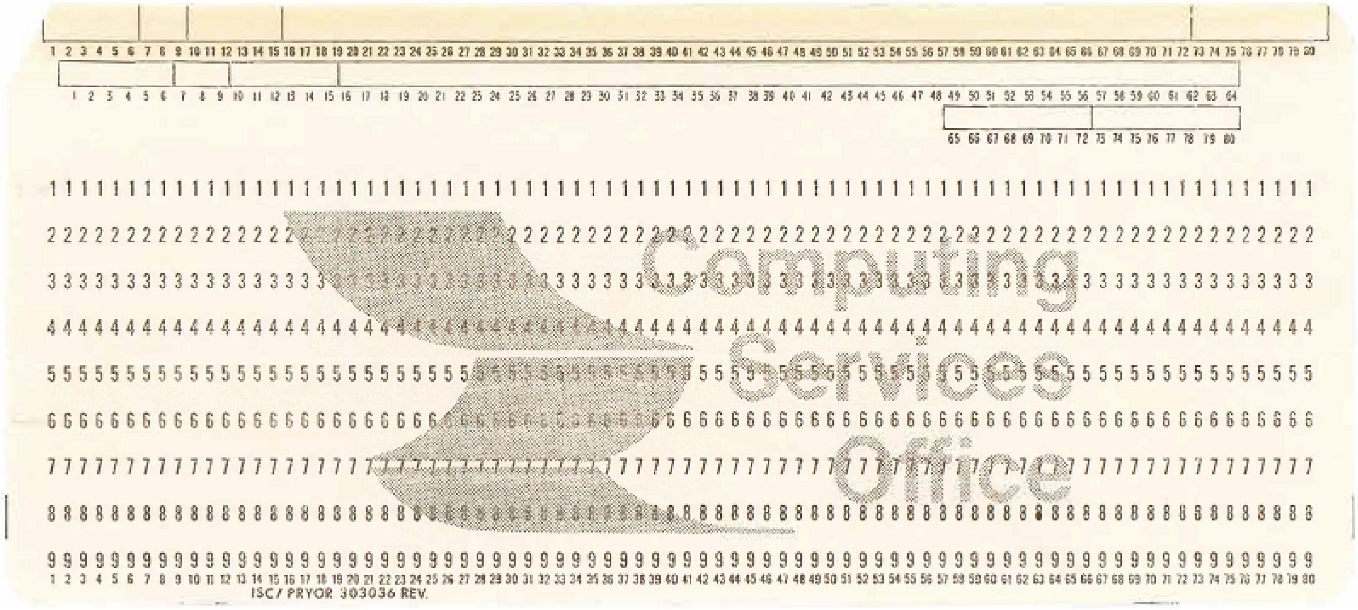

Figure 1. Image of a non-punched (no holes) punched card from the University of Illinois at Urbana-Champaign circa 1980 (Source: author).

2.3 Dominance of FORTRAN

As shown in Table 2, FORTRAN dominated all other languages in GIST software implementations. While there are many dialects of the language (see Chrisman, 1978) most entries are simply labeled FORTRAN IV. Nothing else comes close. It should also be noted that many of the entries have double counts in that it was common to encode some standardized operations in efficient assembly instructions while other operations were written in a higher level language (usually FORTRAN). FORTRAN's dominance would soon be significantly diminished with the rapid diffusion of UNIX workstations and programmers who worked in C and later, C++.

Table 2. Count of the number of program language implementations (Source: Marble, 1980).

Language	Count
FORTRAN	555
Assembly	32
COBOL	26
PL/I	21
ALGOL	20
BASIC	13
C	1

As shown in Table 1, the manufacturer with the second highest count of system instances is

Control Data Corporation (CDC), and this count includes older 6000 series as well as subsequent CYBER systems. For example, the MLMIS system implemented for the state of Minnesota used CDC machines. CYBER systems were found in many universities because they used 60 bit words, a feature that was particularly useful when iterative calculations were performed in either single or, especially, double precision; rounding errors accumulate. For a discussion about the representational implications of finite precision to geospatial applications, see Chrisman (1984).

At the University of Illinois in the late 70s and early 80s, a CDC Cyber 175 machine served as a time sharing computer and also as a means through which batch jobs could be created and submitted to an IBM 360/75 system. Lines 1 and 2 in the following FORTRAN code fragment show the instructions that are used to set up input and output files in a CDC time-sharing environment running NOS (Network Operating System). The “TAPE” convention is a carry-over from another era; these are all disk files. 5 and 6 are the conventional units in FORTRAN for reading and writing. TAPE 7 (DIGREP) holds a tabular report of digitized areas, while TAPE 8 (DIGMAP) contains coordinates that can be plotted using interactive commands.

```
 1        PROGRAM AREA1(INPUT,OUTPUT,DIGMAP,DIGREP,
 2       +TAPE5=INPUT,TAPE6=OUTPUT,TAPE7=DIGMAP,TAPE8=DIGREP)
 3        COMMON X(1000),Y(1000),IPNTS,ACCUM(6),SCALE,NUNIT
 4       +,AREA,PERIM,WIDTH
 5        DIMENSION ISUB(6),UNIT(8)
 6        DATA UNIT/"ACRES","SQ. FT","SQ. YDS","SQ. METERS","SQ. MILES",
 7       +"SQ. KM","SQ. INCHES","SQ. CM"/
 8        REWIND 7
 9        WIDTH=0
10        REWIND 8
11        PRINT *,"THIS IS A PLANIMETRY PROGRAM."
12        PRINT *,"PLEASE SET MODE SWITCH ON DIGITIZER TO "3"."
```

Figure 2.

In the NOS environment these interactive plotting commands took the following form:

GRAB, GCSTEKT
GET, AREA2/ID=3CHOU7Z
AREA2

The GRAB command made a graphics software library available for devices made by a particular display hardware manufacturer (Tektronix). AREA2 is a FORTRAN plotting program that uses the Tektronix device to plot the coordinates stored in the DIGMAP file. The compiled program is stored in a user library (identified as 3CHOU7Z) and is executed by entering the program name (AREA2) with the expectation that there is a file named DIGMAP already created and available for use during the time-sharing session. The syntax is notably friendlier than the JCL example. It should also be noted that the “device dependence” illustrated in this example was a notable problem when writing applications software in this era.

2.4 Minicomputers

Minicomputers have been in existence for many decades, and while several manufacturers had early involvement (e.g. CDC) the Digital Electronics Corporation (DEC) played an outsized role with the development of the PDP-8. A major distinction of this first generation of minicomputers, in comparison with mainframes of the era, was that they used a smaller word size (e.g. 12 bit vs. 36 bit) that had the knock on effects of limiting memory address space as well as numerical precision in calculations. Nevertheless, the substantial reduction in costs associated with minicomputers allowed their acquisition by smaller organizational units, such as academic departments and government GIST offices, and their early move to interactivity, often via teletype terminals, invited the development of a tightly-knit user base who liked to be “close to the machine”, an affordance that eventually gave rise to the personal computer era.

Minicomputers also played a foundational role in the development of the ARPANET, a precursor to the Internet, by solving an incompatibility problems among manufacturers. In 1971 ARPANET had fifteen nodes: nine PDP-10s, five System/360s and the Illiac-IV (Ceruzzi, 2003, p. 194). Compatibility was achieved by placing interface message processors are each node; these were standardized as 16 bit machines and the idea of 16 bit minicomputers proliferated, moving to second generation minis from Data General (Nova) and then the DEC PDP-11 series.

Minicomputers were originally delivered to users with operating systems that were native to the particular manufacturer. DEC provided RSX-11 and later VMS, PR1ME used PR1MOS, and Data General had its AOS, for example. DEC PDPs had a large minicomputer market share and were used to develop the UNIX operating system. One particular version, BSD UNIX developed at the University of California-Berkeley, was widely used on college campuses and it later served as a basis for manufacturer-distributed operating systems running on workstations. The evolution of minicomputers continued with the development of the VAX 32 bit architecture and then the Data General Eclipse systems (see Kidder, 1981).

At GIS conferences in the early 1980s, hardware and software vendors would often rent space in exhibit halls. It was common to observe moving vans emblazoned with “Computer and Electronics Moving” (or some variant) at conference center loading docks since minicomputers were not designed with portability in mind and required specialized handling. While they typically did not require the controlled environmental conditions needed by mainframe systems, they were often dishwasher in size, and sometimes were contained in multiple cabinets, particularly when they were configured with peripheral disk and/or tape drives.

One well-known GIS software product from Esri gained wide use as installed on minicomputers from manufacturers such as PR1ME and DEC: ARC/INFO. As described by Morehouse (1985) at the Auto-Carto 7 symposium, ARC/INFO married the topological model (to represent absolute and relative locations; Peucker & Chrisman, 1975) and the relational model (to represent attributes). The INFO part of the system was originally developed by Henco, Inc. (now Expressway Technologies, Inc). It should be noted that Dueker (1985) described a set of closely-related geo-relational constructs at the same conference. Since its original introduction, this software has gone through considerable evolution. Though now widely deployed in Windows environments, early versions mimicked operating systems of that era and used a “command line” interface that required some effort to master.

2.5 UNIX workstations

In the 1980s the ongoing process of electronic miniaturization and consolidation continued and eventually led to the development of compact, single-user workstations. Sun, which featured the tagline “the network is the computer”, offered products featuring the commodity 68000 microprocessor from Motorola and a variant of BSD UNIX. These systems were relatively powerful and were priced between minicomputers and the low-powered PCs that were just beginning to make inroads at the time. Because GIS had such high demands for handling large data sets and high resolution map displays, workstations were an effective alternative for many university researchers and firms. Software written for UNIX minicomputer implementations could also be easily ported to workstations.

Though Sun started its workstation line with the 68000 processor, work at Berkeley and Stanford ushered in new ways of thinking about how performance could be inexpensively increased. Architectures like those employed by the System/360 and VAX 11/780 employed a large set of complex instructions, thus minimizing what was then relatively slow access to memory. In fact, these were referred to as a complex instruction set computers (CISC). But with substantially increased memory access speeds, a smaller set of instructions could be combined to quickly and efficiently accomplish computational tasks. Moreover, in practice a small percentage of instructions dominates the execution of normal programs (Hennessy & Patterson, 2019, p. A16). The concept of reduced instructions set computers (RISC) had its roots in Berkeley (Patterson) and Stanford (Hennessy) and led to the Sun SPARC workstation that used RISC principles. Other vendors followed suit, most notably the IBM RS/6000. These were impressive UNIX workstations in the mid-to-late 1980s, but technology continued to progress and they soon would be overwhelmed from below as the rise of Intel and Microsoft had begun. Newly capacious and speedy PCs would soon reign supreme.

3. Discussion

There can be no doubt that much early GIST software was developed using some type of System/360 or 370 machine. For example, Tomlinson (1998) reports that IBM Canada developed CGIS hardware and software, and describes a transition of the CGIS from an IBM 1401 to a much larger IBM 360/65 with 512K of memory in the 1960s. Meyers, Wilson and Durfee (1976) also describe the development of Oak Ridge National Laboratory ORRMIS GIS on an IBM 360/91. Other early statewide systems that were implemented in IBM System/360 and 370 environments include LUNR (New York) and MAGI (Maryland).

Despite the strong majority position of IBM, much work was required to convert data and software developed using its 32 bit architecture in predominantly batch mode to other types of systems that were either standalone or linked to a mainframe. This was not normally a straightforward process. Tomlinson, Calkins, and Marble (1976, p. 129) in writing about a data reduction process used by ORRMIS state that the “seven-track tape produced by the scanner/digitizer contains words that are 18 bits long; these must be transformed into 32-bit words in order to be intelligible to the IBM/360 computer”. This amounts to converting 18 bit unsigned integers to 32 bit unsigned integers using a subroutine running on the 360 system (Meyers et al., 1976, p. 26).

Further insights into the difficulties associated with using 1970 era hardware can be gleaned by a description of CGIS retrievals in which a “user queries the system through the services of a programmer who utilizes the CGIS retrieval subsystem" which consisted of programs written in the PL/I language (Tomlinson et al., 1976, p. 65); the user interface was a human. Latency was also an important limiting factor in the use of early systems. CGIS users, for example, were said to misunderstand system capabilities because "they want answers computed the same day that questions are asked, rather than waiting a day or two …" (Tomlinson et al., 1976, p. 71). It is also notable that one of the CGIS system libraries was "JCLIB" that contained job control language statements "necessary to execute the commands in the system" though the system certainly continued to develop into a more interactive system and eventually used Tektronix storage tube technology (Tomlinson et al., 1976, p. 68, 189).

Interactivity continued to improve as minicomputers became more prevalent in GIST implementations, though again there were barriers to entry. UNIX, while far more intuitive than JCL, is still a bit odd on first encounter. And while it is extremely powerful, for example, with an ability to concatenate commands in a pipe, some UNIX commands, including all their options, require some dedication to learning. The UNIX "grep" command is a case in point. GIS software of that era tended to mimic the basic command line structure of the prevailing operating systems (e.g., PRIMOS and UNIX). WIMP (windows, icon, mouse, pointer) interfaces would be developed in workstation environments, but would only come into fruition and widespread adoption in Windows implementations of GIS software in the 1990s.

References

Ceruzzi, P. E. (2003). A History of Modern Computing. 2nd Edition. Cambridge MA: MIT Press.

Chrisman, N. R. (1978). Programming for transportability: A guide to machine independent FORTRAN. Cambridge, MA: Harvard Laboratory for Computer Graphics and Spatial Analysis.

Chrisman, N. R. (1984). On storage of coordinates in geographic information systems. Geo-Processing, 2, 259-270.

Chrisman, N. R. (1998). Academic origins of GIS. In T. Foresman (Ed). The History of Geographic Information Systems: Perspectives from the Pioneers. Upper Saddle River, NJ: Prentice Hall. pp. 33-45.

Chrisman, N. R. (2006). Charting the Unknown: How Computer Mapping at Harvard became GIS. Redlands, California: ESRI Press.

Coppock, J. T. & Rhind, D. W. (1991). The History of GIS. In Geographical Information Systems: Principles and Applications, edited by D. J. Maguire, M. F. Goodchild, and D. W. Rhind, 21–43. London: Longmans Publishers.

Dobson, J. E. & Durfee, R. C. (1998). A quarter century of GIS at Oak Ridge National Laboratory. In T. Foresman (Ed), The History of Geographic Information Systems: Perspectives from the Pioneers. Upper Saddle River, NJ: Prentice Hall. pp. 231-264.

Dueker, K. J. (1979). Land resource information systems: a review of fifteen years experience. Geo-Processing, 1(2), 105-128.

Dueker, K. J. (1985). Geographic information systems: Toward a geo-relational structure. Proceedings of Auto-Carto, 7. Falls Church, VA: American Congress on Surveying and Mapping, pp. 172-177.

Estes, J. E., & Jensen, J. R. (1998). Development of remote sensing digital image processing and raster GIS. In T. Foresman (Ed). The History of Geographic Information Systems: Perspectives from the Pioneers. Upper Saddle River, NJ: Prentice Hall. pp. 163-180.

Foresman, T. W. (Ed.). (1998). The History of Geographic Information Systems: Perspectives from the Pioneers. Upper Saddle River, NJ: Prentice Hall.

Goodchild, M. F. (2018). Reimagining the history of GIS. Annals of GIS, 24(1), 1-8.

Hennessy, J. L., & Patterson, D. A. (2019). Computer Architecture: A Quantitative Approach (6th edition). Cambridge, MA: Morgan Kaufman.

Kidder, T. (1981). The soul of a new machine. Boston, MA: Little, Brown.

Marble, D. (1978). Computer software for spatial data handling. Area, 10(2), 115.

Marble, D. (Ed.). (1980). Computer Software for Spatial Data Handling. (Three volumes). Ottawa, ON: IGU Commission on Geographical Data Sensing and Processing.

Marble, D. F., Calkins, H. W., & Peuquet, D. J. (1984). Basic readings in geographic information systems. SPAD Systems Ltd., Williamsville, NY.

Meyers, C. R., Wilson, D. L., & Durfee, R. C. (1976). An application of the ORRMIS geographical digitizing and information system using data from the CARETS project. ORNL/RUS-12. Springfield, VA: National Technical Information Service.

Monmonier, M. (1985). Technological Transition in Cartography. Madison, WI: The University of Wisconsin Press.

Morehouse, S. (1985). ARC/INFO: A Geo-Relational Model for Spatial Information. Proceedings of AutoCarto, 7. Falls Church, VA: American Congress on Surveying and Mapping, pp. 388-397.

Morton, G. M. (1966). A Computer Oriented Geodetic Data Base; and a New Technique in File Sequencing. Ottawa, CA: International Business Machines Co. Ltd.

Niemann, B., & Niemann, S. (1999). Roger F. Tomlinson, the Father of GIS. GeoInfo Systems, 9(3), 40-44.

Peucker T. K., & Chrisman, N. (1975). Cartographic Data Structures. The American Cartographer, 2(1), 55-69.

Rattenbury, J. (1974). Introduction to the IBM 360 Computer and OS/JCL (Job Control

Language), Revised edition. Ann Arbor, MI: University of Michigan Institute for Social Research.

Tomlinson, R. F. (Ed.). (1970). Environment Information Systems. In Proceedings of the UNESCO/IGU First Symposium on Geographical Information Systems. IGU Commission on Geographical Data Sensing and Processing.

Tomlinson, R. F., Calkins, H. W., & Marble, D. F. (1976). Computer handling of geographical data. Paris, France: The UNESCO Press.

Waters, N. (2017). GIS: History. In Richardson, D., Castree, N., Goochild, M. F., Kopbayashi, A., Liu, W. & Marston, R. A. (Eds.). The International Encyclopedia of Geography. New York, NY: John Wiley.