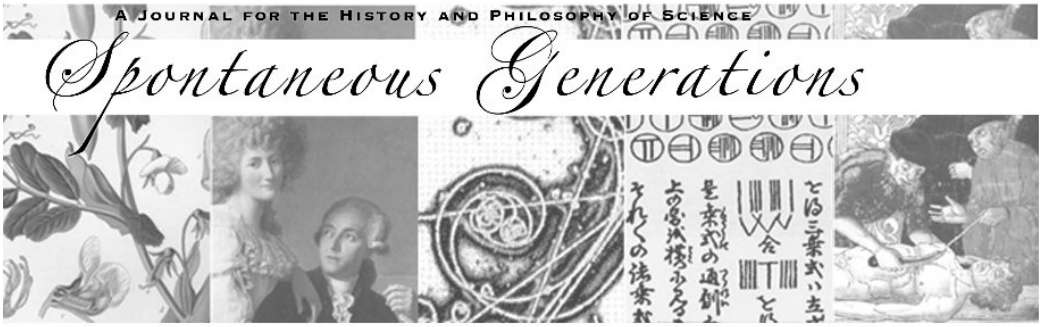

Crossing the Newton-Maxwell Gap: Convergences and Contingencies

Author(s): Matti Tedre & Erkki Sutinen

Source: *Spontaneous Generations: A Journal for the History and Philosophy of Science*, Vol. 3, No. 1 (2009) 195-212.

Published by: The University of Toronto
DOI: 10.4245/sponge.v3i1.3388

E D I T O R I A L O F F I C E S

Institute for the History and Philosophy of Science and Technology
Room 316 Victoria College, 91 Charles Street West
Toronto, Ontario, Canada M5S 1K7
hapsat.society@utoronto.ca

Published online at jps.library.utoronto.ca/index.php/SpontaneousGenerations
ISSN 1913 0465

PEER-REVIEWED

Crossing the Newton-Maxwell Gap
Convergences and Contingencies*

Matti Tedre†
Erkki Sutinen‡

The shift from electromechanical computing to fully electronic, digital, Turing-complete computing was one of the most influential technological developments of the twentieth century. The social, economic, political, interdisciplinary, and cultural aspects behind that shift were significant, but are often ignored. When the contingencies and controversies behind the birth of modern computing are forgotten, the history of computing is often misrepresented as one of uncomplicated linear progress. In this article some of the sociocultural aspects of the birth of modern computing are reviewed. The significance of interdisciplinary work is discussed. The concept of the stored-program paradigm is introduced, and some sociocultural factors behind its birth are discussed. Finally, it is argued that some traits of research that are often considered to be negative, such as opportunism, eclecticism, and stubbornness, have played a positive role in the birth of modern computing technology.

I. INTRODUCTION

The birth of modern (digital, electronic, Turing-complete) computers in the 1940s was one of the most influential technological shifts of the twentieth century. The core ideas of modern computers–ideas such as the Church-Turing Thesis, the Universal Turing Machine, and the von Neumann architecture–have since their inception remained largely unchallenged. Perhaps because of the centrality and power of those core ideas in modern computing, one often forgets the contingencies behind their birth. These core ideas, which constitute what might be called *the*

*Received April 2008. Revised paper accepted March 2009.

†Associate Professor Matti Tedre works as the Head of B.Sc. Program in IT at Tumaini University - Iringa University College, Tanzania. His research interests are the philosophy of computer science, social studies of computer science, and computer science education.

‡Professor Erkki Sutinen works as the Head of Department at Department of Computer Science and Statistics, University of Joensuu, Finland. He works primarily in the field of educational technology, and his primary research interests are ICT for development and computer-assisted education in challenging learning settings.

stored-program paradigm, arose in the 1940s because of a very special economic, social, and political situation.

The stored-program paradigm is not a well-defined concept, but in a somewhat technical sense one can use the term *stored-program paradigm* to refer to the constellation of innovations that surround the stored-program computer architecture. Those innovations include a formalization of computable functions (the Church-Turing Thesis); the idea that instructions can be encoded as strings (the Universal Turing Machine); the idea that instructions and data reside in the same memory storage; random-access memory; and the separation between memory, the processing unit(s), control unit, and input/output unit(s) (von Neumann architecture). Today the stored-program idea is taken as a largely unquestioned basis for automatic computing. The constellation of ideas around the stored-program concept guides forms of inference; promotes one kind of logic of justification; supports some practices of research; and offers some conventions for settling scientific disputes. The stored-program view also limits proper objects of study and proper scientific questions, and it offers a theoretical background for interpreting results.

This article presents some of the contingencies and convergences that gave birth to modern computing. The first part of this article is historical by nature, and deals with the shift from electromechanical and analog computing to electronic, digital, Turing-complete computing. It begins with a portrayal of the open-minded interdisciplinarity and techno-enthusiastic culture of the U.S. in and before the 1940s, continues with an overview of the contingencies and controversies that underlaid the birth of the stored-program paradigm, and ends with a discussion of the unexpected freeing of the stored-program concept from military classification. The second part of this article discusses the birth of the stored-program paradigm in terms of social studies of computer science. After that the article discusses the status of the stored-program paradigm, some sociocultural aspects of work that led to the formation of the stored-program paradigm, and the role of interdisciplinarity in technological work in the field of computing. It is proposed that a number of traits of research that are often characterized as negative in scientists' work–such as opportunism, eclecticism, and stubbornness–sometimes play a positive role in technological work.

II. The Birth of Modern Computing

Automatic computing is an old idea, dating back more than two thousand years (e.g., de Solla Price 1959). However, in the early 20th century automatic computing took giant steps in terms of flexibility and

efficiency. The key words in the birth of modern computing were *fully electronic operation*, *Turing-completeness*, and *digital operation*. Already before the mid-1940s there had been computers that met some of those criteria. For instance, the ABC computer and Colossus were digital and fully electronic, but they were not Turing-complete (Williams 1985, 270–271, 294–296); Konrad Zuse's Z3 was digital and Turing-complete, but it was not fully electronic because it used relays (220–221); and Harvard Mark I was digital but it was electromechanical, not fully electronic (243–244). The Electronic Numerical Integrator and Computer (ENIAC) was arguably the first computer that combined all those ideas. But taking those three ideas together was not an uncomplicated or well-planned process. This section presents some of the contingencies and controversies concerning the birth of the ENIAC.

Pre-Modern Computing

In the field of applied mathematics, one of the common problems is that of calculating the integral, i.e., the area under a given curve. If the curve can be presented as a well-behaved function, it can often be integrated analytically, but if the curve is either not known in an analytic form or is not well-behaved, calculating the area precisely is almost impossible (e.g. Williams 1985, 206–212; Bowles 1996; Polachek 1997). In the history of computing technology, beginning from the early 1820s, there have been numerous attempts to build devices–differential analyzers–for measuring the integral (Campbell-Kelly and Aspray 2004, 45).

In the early 1900s accurate differential analyzers were increasingly needed for the calculation of ballistic firing tables, used by the armed services of the U.S. and Britain. The first working differential analyzers were constructed by Vannevar Bush at Massachusetts Institute of Technology (MIT) around the turn of the 1930s (Williams 1985; Polachek 1997), and replicated and redesigned in Britain in 1935 (Croarken 1992). The final version of Bush's machine, "Rockefeller Differential Analyzer # 2," was used extensively during the Second World War by the armed services of both America and Britain (Williams 1985, 206–212).

The differential analyzers were results of interdisciplinary work, yet the role of interdisciplinarity in the development of the computing technology has been given too little attention (Puchta 1996). A number of sectors needed Vannevar Bush's computing machinery and contributed to its development: for instance, electrical engineering, mathematics, the sciences (especially physics), radio technology, and warfare (Puchta 1996; Bowles 1996; Williams 1985, 209). Puchta (1996) argued that breaking the disciplinary boundaries was necessary in order to achieve the goals that the device was built for, and that the interdisciplinary development work

enabled more efficiency and creativeness within the traditional disciplinary boundaries, too.

The fields that Vannevar Bush's work with differential analyzers brought together had not had much interplay before. For instance, Bush's work combined knowledge from logic, pure or formal mathematics, mechanical engineering, and innovations from seafaring (Williams 1985, 209; Puchta 1996; Bowles 1996). Fields that had an impact on Bush's work and subsequent development of early computers include physics, mathematics, logic, engineering, astronomy, navigation, and even meteorology (Naur 1992, 596–597; Campbell-Kelly and Aspray 2004, 52–59). Also after the turn to digital computing, the pioneers of computing came from various fields: Alan Turing was a logician and mathematician, Leslie Comrie was from the field of astronomy, John Atanasoff was an electrical engineer and mathematician, John von Neumann worked on a number of mathematical fields, Wilkes and Mauchly were physicists, Herman Goldstine was a mathematician, and John Presper Eckert was an electrical engineer.

Vannevar Bush's machinery was born in a time and place of extraordinary public faith in the power of technology to benefit humankind. Analyzing the context in which Vannevar Bush's machinery was developed provides information about the intellectual, interdisciplinary, and social dimensions, as well as sources, of modern computing (Puchta 1996; Bowles 1996). Bowles (1996) argued that the techno-utopistic, technologically enthusiastic, and practical atmosphere of the U.S. enabled American scientists to obtain the funding needed for the development of computing machinery. Campbell-Kelly and Aspray (2004,19) called the American enthusiasm "America's love affair with office machinery." Flamm (1988, 136) even claimed that tradition-bound culture was the crucial factor that made British scientists unable to compete with their American colleagues in building the differential analyzer. Culture has been argued to affect the valuation of theory and practice in a field, the amount of funding a field gets, the foci of research, public support of a field, and the popularity of technological disciplines (Campbell-Kelly and Aspray 2004, 19; Flamm 1988, 136; Bowles 1996; Aspray 2000).

The techno-utopistic atmosphere brought about a number of phenomena that supported the success of the U.S. in the field of automatic computing. Bowles (1996) named four contextual factors that were the main sources of success for the development of computing machinery in the U.S. (see also Croarken 1992; Croarken 1993). First, Bowles (1996) argued that the American popular and professional culture reinforced the practical, professional values of the American society. Second, the American universities supported computing fields, and many of them

had their technical schools, such as the University of Pennsylvania's Moore School of Electrical Engineering and MIT's laboratories (Bowles 1996; Aspray 2000). Third, the engineers in the U.S. held a high public status–the American engineer was the hero of the new century–and this affected the financial support from government and industry (Bowles 1996; Aspray 2000). Fourth, the American press reinforced an anthropomorphic, laudatory image of computational instruments. For instance, the American press used terms such as "the super-brain," "man-made mental giant" (Martin 1993), and "the robot Einstein" (Bowles 1996). This anthropomorphic terminology perhaps derived from the language of the pioneers of early computing (Grier 1996).

Shift to Digital, Electronic, Turing-Complete Computing

The differential analyzers of the early 1900s were slow, inaccurate, and expensive. Although differential analyzers could produce results three times as quickly as a human computer (Comrie 1944), they could not satisfy the growing calculation needs of the U.S. Army. There was also a trade off between accuracy and cost of the analyzer: Comrie (1944) stated that the accuracy of differential analyzers was roughly $1/x$, where x is the amount of money one was prepared to spend. In May of 1943, the desperate need for high-speed calculation led the Ballistic Research Laboratory of the U.S. Army to agree to fund a new high-speed computer. This agreement in part led to the beginning of the construction of the ENIAC at the Moore School of Electrical Engineering at the University of Pennsylvania (Polachek 1997; Winegrad 1996). The change that the birth of fully electronic computers entailed is sometimes called *crossing the Newton-Maxwell gap* because it marks a shift from mechanical and electromechanical computing machinery (governed by Newton's laws of motion) to electronic devices (governed by Maxwell's laws of electromagnetism) (Ceruzzi 1997).

The U.S. Army's need for calculating speed was the main force driving the development of computing in the early 1940s. The existing computing technology could not keep up with the U.S. Army's need for computing power for developing new weapons and calculating trajectory tables (Flamm 1988, 48; Campbell-Kelly and Aspray 2004, 73; Williams 1985, 272–273). Historians of computing usually agree that the best people in the field of computing became associated with the University of Pennsylvania's Moore School only because of the U.S. Army's increased need for automatic computing during the Second World War (Marcus and Akera 1996; Flamm 1988; Campbell-Kelly and Aspray 2004; Williams 1985). The U.S. Army took a gamble on untested computing technology, bypassed the scientific establishment (Flamm 1988, 252), and decided

to fund a new computing technology project proposed by John Presper Eckert and John Mauchly at the Moore School of Electrical Engineering. This gave Mauchly and Eckert the chance to realize their ideas for a fully electronic computing machine.

Even though the Army gave a green light to Eckert and Mauchly, there was strong resistance towards the ENIAC project. Headed by the highly recognized computing pioneer Vannevar Bush, the established scientific community fiercely opposed the ENIAC project as well as Eckert and Mauchly's choice of electronic circuit elements instead of electromechanical ones (Flamm 1988, 48). Compared to electromechanical elements, electronic circuits were new and untested technology. Also, many of the pioneers of computing had built their careers on electromechanical technology. Despite that opposition, the Army was willing to gamble on Eckert and Mauchly's radically new approach. That gamble, together with Eckert and Mauchly's stubborn defiance of the scientific establishment, led to the birth of the fully electronic, digital, Turing-complete computer. Interestingly, economic aspects did not play a decisive role in the development of modern computing–it was cheap to hire staff to do calculations manually, so electronic computers were developed despite the costs, rather than as cost-saving devices (Campbell-Kelly and Aspray 2004, 141). The traditional cost-sensitive customers, such as private companies and civilian government institutions, would not have funded such experimental projects, but for national security purposes cost was not an important factor (Flamm 1988, 2).

A crucial part of the success of the ENIAC project was moving the right people to the right place. In the wake of the Second World War, the U.S. Army moved a number of top scientists to the Moore School of Electrical Engineering at the University of Pennsylvania, and the university accelerated academic programs by eliminating vacations (Campbell-Kelly and Aspray 2004, 71). A lot of credit for the ENIAC project has been given to John Mauchly, who was in the midst of a very strong research and training center for electrical engineering at the Moore School. Mauchly had mentors above him and students beneath him, and he could enlist both into the ENIAC project (Marcus and Akera 1996). But Mauchly did not initiate everything–Marcus and Akera (1996) argued that Herman Goldstine's initiative was crucial to launching the ENIAC project. In addition, Marcus and Akera (1996) argued that the birth of electronic computing owed as much to various trends in the historical context as to the ability of Mauchly to draw together various pieces of the puzzle himself.

The ENIAC team had a grand vision for the new computing machine: Grier (1996) wrote that the designers and constructors of the ENIAC believed that the ENIAC would revolutionize science by establishing a new

scientific method based on electronic computing. To make this change clear, the researchers attempted to delineate a clear boundary between the new world of electronic computers and the older world of human computers by using words that could not be connected with human computing (e.g., they talked about "programs" instead of "calculation"). Another reason for the new terminology was the incommensurability of concepts concerning electronic computers with previous knowledge (Tedre 2006, 208).

Convergences and Contingencies

When the ENIAC was completed it was the first large-scale electronic, digital, Turing-complete computer. The success of the ENIAC convinced many institutions and people from science, military, and industry to commit to the development of electronic high-speed computing (Marcus and Akera 1996). The ENIAC was also the only electronic computer in the world for three years (Mauchly 1979). Marcus and Akera (1996) argued that there were multiple origins of high-speed electronic computing, all of which fundamentally came together because of Word War II. Firstly, there was significant development in the components needed for electronic computing (although those components had been developed for measurement equipment, not for computers). Secondly, there were advances in theoretical and experimental physics (such as increased understanding of the behavior of electrons). Thirdly, the situation of the Moore School of Electrical Engineering in the midst of firms (such as Philco, RCA, and Atwater Kent) as well as major research facilities (such as Bell Laboratories) provided a fertile soil for the ideas of Mauchly and others.

Although World War II had, in addition to the positive effects, also some negative effects on the development of computing in the U.S.–such as ending a wide variety of computing research projects (Aspray 2000)–the war also shifted emphasis from demand for cost-effectiveness to demand for function, performance, and availability at any cost (Pugh and Aspray 1996). Due to World War II both government and academic circles were able to recognize the importance of automating and centralizing computation (Croarken 1992). It has been argued that the theoretical and practical developments at the University of Pennsylvania would at any other time, in all likelihood, have been dismissed as interesting, but impractical and expensive (Winegrad 1996).

Grier (1996) noted that, unfortunately, the ENIAC became a burden before it was even completed. The ENIAC became operational just as the development team at the Moore School began to understand and appreciate the power of stored programs and serial arithmetic. The

concept of the stored-program computer was conceived around January 1944 (Mauchly 1979), and the developers of the ENIAC considered that step so important and so powerful that they wanted to abandon their previous work with the ENIAC, and to focus on the new technology. However, the development team quickly discovered that it was not possible for them to abandon their previous work because the sponsors wanted to get their money's worth, and because of the large intellectual investments in the machine (cf. Grier 1996). Nevertheless, John von Neumann (1945) published Eckert and Mauchly's ideas in June 1945 in his landmark text *First Draft of a Report on the EDVAC*. Thus, in a twist of fate, the architecture conceived largely by Eckert and Mauchly–although clarified by von Neumann (Ceruzzi 2003, 21–22)– came to be known as *von Neumann architecture.*

Usually technological breakthroughs funded by the Army, especially those as powerful as the new stored-program concept, are strictly withheld from the public. For instance, the British Colossus computers were classified for an extended period of time (Williams 1985, 287–296). The development of the ENIAC was indeed done secretly. Publication of papers was strictly forbidden, and discussion was limited (Winegrad 1996). However, due to a quirk of history, the ideas generated by the ENIAC team were, after the unveiling of the machine, freed from their military security classifications.

There were several reasons for that lapse of security. First, although von Neumann was criticized for using anthropomorphic language with computers (Grier 1996; Stibitz 1985), using neurological terms instead of engineering terms enabled von Neumann to circumvent military security (Pugh and Aspray 1996). Second, the ENIAC was completed too late in order to demonstrate its military value during the war (Pugh and Aspray 1996). Third, the Army officials conceived the ENIAC as a general-purpose calculator instead of a specialized military machine. Fourth, the Army as well as the University of Pennsylvania were eager to publicize their accomplishments in electronics (Pugh and Aspray 1996).

The publication of the stored-program plans created considerable interest both nationally and internationally. The Moore School even offered a famous series of lectures about everything that was needed for constructing a stored-program computer (Campbell-Kelly and Williams 1985[1946]). In May 1946, Leslie John Comrie returned to England from the U.S. with a copy of von Neumann's draft, and after Maurice Wilkes from Cambridge had read the document, Wilkes' Mathematical Laboratory decided to build their own stored-program computer (Croarken 1992). The stored-program concept and principles spread quickly to academics around the world.

After World War II, those American universities that had entered the computing field in the 1940s were able to significantly influence how computing developed in the following decades. In effect, the early entrants to the nascent field of computing were able to define the models and exemplars for modern automatic computing (Aspray 2000). In addition, those universities were able to also set research agendas in the emerging field of computing in ways that were to those universities' own advantage (Aspray 2000). Early entrants to computing were also able to set the research problems so that the starting points were most favorable for them, the outcomes were anticipated, and only the tools and methodology necessary to achieve the results were unknown. Merely being first is enough to shape the discipline.

In 1949, the world's first stored-program computer became operational. However, two computers have been argued to have been the first: EDSAC in the Cambridge Mathematical Laboratory performed its first automatic calculation on May 6, 1949 (Worsley 1975); and the first of the two computers that became the BINAC had run nonstop for 44 hours in April 1949 (Mauchly 1979). Some sources claim that the BINAC ran its first program in the late summer of 1949 (Williams 1985, 361–362). The topic of 'firsts' in electronic computing is controversial, but nevertheless, before the 1950s, the first fully electronic, digital, stored-program computer was operational: the Newton-Maxwell gap had been crossed and the stored program paradigm had been born.

III. Discussion

The birth of the first fully electronic, digital, Turing-complete computer happened during the Second World War under very special social, political, and economic circumstances. Those developments were brought to a head with a number of computing pioneers taking a route opposed by the scientific establishment. The development of modern electronic computing owes much to a number of non-technological factors. The factors that steered the development of computing include *organizations*, such as universities, profit-making companies, institutions, and government branches; *personalities* with different characters, backgrounds, contacts, reputations, and motivations; *contingencies*, such as security lapses, convergences of common interests, quarrels, visionaries, opinion leaders, and misunderstandings; *interdisciplinarity* between fields such as weapons research, theoretical and experimental physics, electrical engineering, mathematics, and logic; and *cultural influences*, such as techno-enthusiasm, emphasis on practicality, techno-utopianism, risk-taking, regulations, policies, and élitist attitudes (cf. Tedre 2006, 204–207).

The history of the birth of modern computing supports the contingency thesis, which holds that the current state of affairs in a specific science could have developed through a different route (Hacking 1999). The contingency thesis stands in opposition to the inevitabilist thinking, which holds that there is a natural path of development, and no matter what historical contingencies may affect local steps of development, laws of nature and logic eventually direct development to the best, inevitable, direction. But even though the contingency thesis is supported by the sheer number of lucky coincidences and planned convergences in the history of modern computing, one inevitabilist aspect is certain. Although a great number of non-technological factors can be attributed to the development of modern computing, technological and theoretical aspects of computing are at the core of most texts on the history of computing. That is, the laws of nature and logic dictate some limits of automatic computing.

Incommensurable Concepts

As noted earlier, much of the credit for the rapid spread of the stored-program concept to universities worldwide can be attributed to von Neumann's choice of neurobiological terminology instead of engineering terminology. Generally speaking, one of the main motives for anthropomorphizing computing terms is the incommensurability of new technological knowledge with earlier knowledge. To make the new computing knowledge easier to understand, von Neumann attempted a metaphor transfer from neuropsychology to computing technology. By explicitly referring to an article published in the Bulletin of Mathematical Biophysics in 1943, von Neumann (1945) made parallels from neuropsychology (neurons, synapses, axons) to the designs of the stored-program computer. Furthermore, von Neumann (1945) called input and output devices of the machine *organs*: "*The device must have organs to transfer* [numerical information] *from R into its specific parts C and M.*" In contrast to von Neumann, Mauchly and Eckert wanted to disassociate from earlier science by introducing new terminology, such as *a program*. Grier (1996) argued that "*the researchers attempted to delineate a clear boundary between the new world of electronic computers and the older world of human computers.*"

Although the use of anthropomorphic terminology when dealing with computer systems has been labeled a "symptom of professional immaturity" (Dijkstra 1982, 129–131), anthropomorphic language is not shunned today. Everyday computing terminology today includes anthropomorphic terms such as neural networks, master and slave, sleeping, killing, dying and being alive (processes), artificial intelligence and memory, agents, viruses, parents, siblings and relatives (trees), and

so forth (Tedre 2006, 213). Usually anthropomorphic terms in computing are attempts to create a parallel between a radically new concept and previous common knowledge.

The stored-program paradigm truly epitomized a radical intellectual and technological change. The stored-program paradigm introduced a number of concepts without an analogy to the computing standards of the time. For instance, the concepts of program counter and instruction register did not have counterparts in science before the stored-program paradigm (program counter keeps track of the instruction sequence and instruction register stores the instruction that is being executed). As the terminology in post-stored-program computing was incommensurable with the terminology in pre-stored-program computing, one could not demand a continuation between old and new concepts (cf. Feyerabend 1970; Kuhn 1970). Such a demand for continuation, explanation, or reduction of concepts simply could not have been realized.

Resistances and Accommodations

Investigating the social aspects of the birth of the stored program paradigm provides information about the mechanisms of the growth of knowledge and technological shifts in computing. But for that discussion, one must take an epistemological stand: there are both *brute facts* and *institutional facts* in computing (see Searle 1996). The physical and chemical properties of the substances that computers are made of do not depend on people, but almost everything else in the computing domain is socially constructed. The arrangement of logic gates, the radix of numeral systems, the design choices of execution units, abstraction layers, and all hierarchies in computing are determined by choices that computer designers make. Designers in computing fields make their choices based on a diversity of issues such as economics, functionality, architectural choices, power consumption, heat dissipation, standards, usability, abstraction choices, personal design preferences, and so forth. However, although hardware and software designs are completely human-made, one cannot tell for certain if the hierarchical structures of computer systems reflect an inherent hierarchy of the world, or if they are a tool for understanding an inherently unorganized world (Tedre 2006, 437).

The numerous contingent elements in the ENIAC project make it difficult to situate it in an inevitabilist framework, which explains technological development as following an inevitable route defined by natural laws (e.g., Hacking 1999, 78–79). Rather, the development of modern computing was greatly influenced by the ideas and strategies of a number of people who came together due to sociopolitical circumstances. As the scientific community, headed by Vannevar Bush, opposed the

ENIAC project and Eckert and Mauchly's choice of electronic circuit elements (e.g., Flamm 1988, 48, 252), the history of the ENIAC project is also an example of the fallibility and occasional dogmatism of the scientific community. Individual academics also had a great influence on the success of early computing–for instance, Vannevar Bush's role as a mediator between the subcultures of science, mathematics, and engineering had an influence on the MIT science policy (Puchta 1996).

The standard explanatory frameworks of scientists' work (e.g. Hempel 1965, 331–496; Popper 1959[1935]; Kuhn 1996[1962]) fail to account for scientists' and engineers' work with the ENIAC project. Instead, Pickering's (1995) "mangle" framework offers a fruitful perspective from which to view the ENIAC project. In the mangle framework there are abstract scientific theories about phenomena, there are down-to-earth models of how scientific instruments work and what can be done with them, and there are the instruments themselves. The development of new science or a new instrument is often a long-lasting struggle between problems (resistance) and solutions to them (accommodation). The accommodation processes can include, for instance, revising a scientific theory, re-thinking beliefs about how an apparatus works, or modifying the apparatus itself (Pickering 1995; Hacking 1999, 71). The aim of the scientist is to find a robust fit between science, model, and apparatus. To create that fit scientists employ different material, conceptual, and social accommodation strategies that, naturally, face different kinds of new resistances.

In the mangle of early computing, the Turing Machine (Turing 1936) was the abstract theoretical construction, the blueprints for the fully electronic programmable computer were the (high-level) model of an instrument, and the ENIAC and its peripherals were the instruments. In their work the ENIAC team frequently encountered clashes between (1) the theory of computation, (2) their theory about how computers should work, and (3) the ENIAC computer itself. Eckert and Mauchly had a clear idea of how computers should be designed, but the world *resisted.* When the ENIAC team encountered clashes between their expectations and the resistance of the world, they *accommodated* those clashes by either revising their theory about how computers should work or by revamping parts of the ENIAC. For instance, the original plan of the ENIAC escalated from 5,000 to 18,000 vacuum tubes, the costs grew from $150,000 to $400,000, the operating voltages were dropped significantly in order to reduce stress on the vacuum tubes, and quite early in the construction project, Eckert and Mauchly understood that the ENIAC designs were insufficient for generic computation (Campbell-Kelly and Aspray 2004, 76–83). The stored-program paradigm grew out of the accommodations to the problems in building the ENIAC, and possibly from the theoretical

insights of Alan Turing (1936)–yet it is not known if Eckert and Mauchly were aware of Turing's ideas (although von Neumann might have been).

After the crossing of the Newton-Maxwell Gap, in the early days of modern computing, there was great uncertainty about research directions. Most computing machines of the time differed from each other in their architecture, design, constraints, and working principles (Ceruzzi 2003, passim). Over the course of time knowledge about the directions of computing accumulated, and researchers increasingly followed paths that others had already tread. This is traditionally called the growth of knowledge, yet it is also characteristic of the growth of technological momentum (cf. Hughes 1994). The term *technological momentum* refers to the phenomenon whereby younger, developing systems tend to be more open to extra-technological influences, while older, more mature systems prove to be less susceptible to outside influences and therefore more deterministic by nature (Smith and Marx 1994, 101; Hughes 1994).

Previous design choices and compatibility issues did not hinder early computer designers, so they were able to begin with a tabula rasa. But, over time, computing systems became more complex and more interdependent. As the number of computer installations grew, the design decisions of computing had to be increasingly based on compatibility. But only the first united architecture, the IBM System/360, marked a definite turning point from somewhat unconstricted pioneer work to a certain kind of technologically determined rigidity in the development of computing machinery.

Interdisciplinary Situations

Today, work in computing fields also follows the "mangle of practice" between problems (resistance) and solutions to them (accommodation), as described by Pickering (1995). Through a continuous cycle of corrections to theories, honing of techniques, improvements of mechanisms, negotiations between stakeholders, debates, power struggles, and constant criticism, some theories and techniques gradually become a part of the relatively stable core of knowledge about automatic computing. For instance, Eckert and Mauchly's work on revamping their computer and their theories at the same time is a prime example of the mangle of practice. Similarly, the knowledge that researchers gain when they construct and revamp instruments is gradually crystallized into theories about instruments, which, in part, gradually become a part of the relatively stable core of computing knowledge. That is, theories, techniques, and theories about instruments gradually increase their epistemological objectivity; knowledge becomes more firmly entrenched and believed more strongly. Practical work in computing is a mixture

of receptivity and tenacity; computing professionals can at the same time draw on a number of disciplines and be stubborn in the face of multidisciplinary opposition.

The convergence of a number of different ideas from several disciplines was crucial for the development of early computing. When several experts from different fields work together to achieve a common goal, they need to communicate the necessary aspects of their art or science to the others, which leads to the experts explaining their art or science in the clearest and simplest way possible. The experts need to be able to present their art or science without theoretical or metatheoretical issues, counterarguments, alternatives, or disciplinary controversies.

In interdisciplinary situations, simple and straightforward knowledge about powerful ideas enables the participants to use concepts or innovations without getting mired in field-specific debates. If the collaborators are ignorant about the traditional boundaries of specific arts or sciences, they may unexpectedly cross those boundaries, especially boundaries of applicability of ideas. However, it seems that superficial knowledge about ideas can also be counter-productive. Using a powerful idea without knowledge about its pitfalls, (meta-)theoretical issues, counterarguments, disciplinary controversies, or alternatives, may cause the researcher problems that experts of the discipline would avoid.

It seems that when different cultures, domains, disciplines, experts, and concepts clash and interact, the situation is inherently eclectic and opportunistic. That is, when there are many cultures, domains, experts, disciplines, and concepts together, demands for reducibility and instant clarity must be dropped. In an eclectic combination of expertise, every person cannot become knowledgeable about everyone else's art or science. An eclectic combination of incommensurable arts and sciences creates an ontological, epistemological, and methodological anarchy in the sense that no ontology, epistemology, or methodology can be claimed superior over others. In many aspects, that anarchy has been the driving force of the rapid development of computing beginning from the early 1900s.

IV. Conclusions

In the light of the history of computing, the birth of modern computing was a contingent outcome of a convergence of a large number of ideas in a turbulent time. In the early 20th century, the sciences progressed on a broad front, yielding useful tools, ideas, and techniques that designers of computing machinery were able to use. The computing researchers of the early 20th century were opportunists: they took ideas from an eclectic selection of fields, and used those ideas to the appropriate extent. Also,

the people who brought the ENIAC to life stubbornly defied the academic establishment, which at the time was not convinced of the advantages of fully electronic computing. All in all, the birth of modern computing was not the straightforward outcome of planned, successive development steps, but a convergence of organizational issues, personalities, contingencies, interdisciplinarity, and cultural, political, and economic influences.

The U.S. Army had a strong influence over the directions of automatic computing in the 1940s. The military gave clear objectives for research, they provided continuous research funding, and they offered markets for the new machinery. The U.S. Army also brought together the people needed for the ENIAC project. But although the ENIAC was built for the Army, after its completion the ENIAC and subsequent EDSAC designs were unexpectedly freed from military classifications. The famous Moore School lectures made the stored-program concept public, and several universities all over the world were able to build their own stored-program computers around the turn of the 1950s.

The development of computing machinery from the early 1900s on has been a triumph of anarchism and opportunism: the pioneers of computing used whatever means were necessary to achieve their goals, and used knowledge from several disciplines in order to advance theoretical, technological, and practical knowledge of automatic computing. The mode of working of computing professionals at Moore School in the 1940s is well suited with the “mangle of practice” framework. The researchers working with the ENIAC project had their fundamental theories compatible with Turing’s work; they had their theory of how automatic computing machinery should be designed; and they had the ENIAC and its peripherals. In their work they struggled between problems (resistances) and solutions to them (accommodations). They employed a variety of different material, conceptual, and social accommodation strategies that gave rise to different kinds of new resistances. The aim of the researchers was to find a robust fit between science, model, and apparatus–which was finally embodied in the blueprints of EDVAC, in the computers BINAC and EDSAC, and in the birth of the stored-program paradigm.

MATTI TEDRE
Tumaini University
Tanzania
matti.tedre@acm.org
(corresponding)

ERKKI SUTINEN
University of Joensuu
Finland
sutinen@cs.joensuu.fi

REFERENCES

Aspray, William. 2000. Was Early Entry a Competitive Advantage? US Universities That Entered Computing in the 1940s. *IEEE Annals of the History of Computing* 22(3): 42–87.

Bowles, Mark D. 1996. U.S. Technological Enthusiasm and the British Technological Skepticism in the Age of the Analog Brain. *IEEE Annals of the History of Computing* 18(4): 5–15.

Campbell-Kelly, Martin and William Aspray. 2004. *Computer: A History of the Information Machine.* 2nd ed. Oxford: Westview Press.

Campbell-Kelly, Martin and Michael R. Williams, eds. 1985[1946]. *The Moore School Lectures*. Cambridge, MA: MIT Press.

Ceruzzi, Paul E. 1997. Crossing the Divide: Architectural Issues and the Emergence of the Stored Program Computer, 1935-1955. *IEEE Annals of the History of Computing* 19(1): 5–12.

Ceruzzi, Paul E. 2003. *A History of Modern Computing.* 2nd ed. Cambridge, MA: MIT Press.

Comrie, Leslie J. 1944. Recent Progress in Scientific Computing. *Journal of Scientific Instruments* 21(8): 129–135.

Croarken, Mary G. 1992. The Emergence of Computing Science Research and Teaching at Cambridge, 1936–1949. *IEEE Annals of the History of Computing,* 14(4): 10–15.

Croarken, Mary G. 1993. The Beginnings of the Manchester Computer Phenomenon: People and Influences. *IEEE Annals of the History of Computing,* 15(3): 9–16.

Dijkstra, Edsger W. 1982. How do we tell truths that might hurt? In *Selected Writings on Computing: A Personal Perspective,* ed. Edsger W. Dijkstra, 129–131. Germany: Springer-Verlag.

Feyerabend, Paul. 1970. Consolations for the Specialist. In *Criticism and the Growth of Knowledge,* eds. Imre Lakatos and Alan Musgrave, 197–230. London: Cambridge University Press.

Flamm, Kenneth. 1988. *Creating the Computer: Government, Industry, and High Technology*. Washington, DC: Brookings Institution.

Glass, Robert L., V. Ramesh and Iris Vessey. 2004. An Analysis of Research in Computing Disciplines. *Communications of the ACM* 47(6): 89–94.

Grier, David A. 1996. The ENIAC, the Verb ”to program” and the Emergence of Digital Computers. *IEEE Annals of the History of Computing* 18(1): 51–55.

Hacking, Ian. 1999. *The Social Construction of What?* Cambridge, MA: Harvard University Press.

Hempel, Carl G. 1965. *Aspects of Scientific Explanation And Other Essays in the Philosophy of Science*. New York: The Free Press.

Hughes, Thomas P. 1994. Technological Momentum. In *Does Technology Drive History? The Dilemma of Technological Determinism,* eds. Merrit Roe Smith and Leo Marx, 101–114. Cambridge, MA: MIT Press.

Kuhn, Thomas. 1970. Reflections on My Critics. In *Criticism and the Growth of Knowledge,* eds. Imre Lakatos and Alan Musgrave, 231–278. London: Cambridge University Press.

Kuhn, Thomas. 1996 [1962]. *The Structure of Scientific Revolutions.* 3rd ed. Chicago: University of Chicago Press.

Marcus, Mitchell and Atsushi Akera. 1996. Exploring the Architecture of an Early Machine: The Historical Relevance of the ENIAC Machine Architecture. *IEEE Annals of the History of Computing* 18(1): 17–24.

Martin, C. Dianne. 1993. The Myth of the Awesome Thinking Machine. *Communications of the ACM* 36(4): 120–133.

Mauchly, John W. 1979. Amending the ENIAC Story. *Datamation* (October): 217–220.

Naur, Peter. 1992. *Computing: A Human Activity*. New York: ACM Press.

Pickering, Andrew. 1995. *The Mangle of Practice: Time, Agency, and Science*. Chicago: University of Chicago Press.

Polachek, Harry. 1997. Before the ENIAC. *IEEE Annals of the History of Computing* 19(2): 25–30.

Popper, Karl. 1959[1935]. *The Logic of Scientific Discovery*. London: Routledge.

Puchta, Susann. 1996. On the Role of Mathematics and Mathematical Knowledge in the Invention of Vannevar Bush's Early Analog Computers. *IEEE Annals in the History of Computing* 18(4): 49–59.

Pugh, Emerson W. and William Aspray. 1996. Creating the Computer Industry. *IEEE Annals of the History of Computing* 18(2): 7–17.

Searle, John R. 1996. *The Construction of Social Reality.* England: Penguin Press.

Smith, Merritt Roe and Leo Marx, eds. 1994. *Does Technology Drive History? The Dilemma of Technological Determinism*. Cambridge, MA: MIT Press.

Solla Price, Derek J. de. 1959. An Ancient Greek Computer. *Scientific American* (June): 60–67.

Stibitz, George. 1985. Introduction to the Course on Electronic Digital Computers. In *The Moore School Lectures,* eds. Martin Campbell-Kelly and Michael R. Williams, 5–16. Cambridge, MA: MIT Press.

Tedre, Matti. 2006. *The Development of Computer Science: A Sociocultural Perspective*. University of Joensuu, Computer Science, Dissertations XIV. Joensuu, Finland: Yliopistopaino.

Turing, Alan M. 1936. On Computable Numbers, With an Application to the Entscheidungsproblem. *Proceedings of the London Mathematical Society, Series 2,* 42: 230–265.

von Neumann, John. 1945. First Draft of a report on the EDVAC. In *The Origins of Digital Computers: Texts and Monographs in Computer Science,* ed. Brian Randell, 355–364. New York, NY: Springer.

Williams, Michael R. 1985. *A History of Computing Technology*. New Jersey: Prentice-Hall.

Winegrad, Dilys. 1996. Celebrating The Birth Of Modern Computing: The Fiftieth Anniversary of a Discovery At The Moore School of Engineering of the University of Pennsylvania. *IEEE Annals of the History of Computing* 18(1): 5–9.

Worsley, Beatrice H. 1975. The E.D.S.A.C. Demonstration. In *The Origins of Digital Computers: Texts and Monographs in Computer Science,* ed. Brian Randell, 395–401. New York, NY: Springer.