The Evolution of Computing: A Historical Perspective and Relevance

Anthony J. Coots

National University

TIM-8102: Principles of Computer Science

Dr. *******

April 13th, 2025

The Evolution of Computing: A Historical Perspective and Relevance

In a world increasingly driven by technological advancement, it's relatively easy to forget that computers were once modest tools built only to do precisely that: compute. Though the adoption of computers has carried on, whether they take shape in a refrigerator, smartwatch, cellphone, and so on, it is still important to understand the history of computing and computer science to comprehend just how computers have evolved to this point. These evolutions have come rather quickly, with major ones happening every decade from Charles Babbage's Difference Engine in the early 1820s to modern quantum computers. However, progress has not always been met with open arms, then or now, and future research will likely face very similar contention. Even with these challenges, the core principles of computer science, such as but not limited to abstraction, decomposition, algorithm development, and automata, will continue to develop. For these reasons, considering the discipline's history remains essential to both current and future research and practice in computer science. This paper will explore just how the historical development of computing has shaped modern computer science and will continue to do so.

Historical Evolution of Computing and Computer Science

Early Foundations: 1800s-1940s

While simple computational methods, such as the abacus and early mechanical calculators, date back centuries, it was in the early 19th century that the most important conceptual shift toward programmable computing occurred. Thanks to individuals who had a vision that was greater than the landscape at the time, new technology came to life to process input and produce a designed and desired output. Though these machines, towards the end of this era, were significantly larger yet less practical than modern computers, the conceptual

contributions before and coming from these machines helped lay the foundation for generations to come. These early developments helped make way for the coming of mainframes, programming languages, and the field of computer science that would continuously inspire innovation.

Charles Babbage is one of the early pioneers of computing and computer science. He is explicitly known for designing the Difference Engine in 1824, a device that was designed to compute mathematical functions and print the results, though the machine was finally constructed in 1991. He also conceptualized the Analytic Engine, more ambitious than the Difference Engine that gave way to modern computing features such as programmability and memory, though never built. Despite Babbage's work remaining incomplete in his lifetime, his designs, more specifically the Analytic Engine, laid the groundwork for modern computers (Strawn, 2023). Additionally, his work inspired Ada Lovelace, who is often recognized as the first computer programmer.

Ada Lovelace, in a similar light to Charles Babbage, was also a pioneer in computing and computer science. Though introduced to important figures such as Babbage himself and Augustus De Morgan, she was an impressive mathematician with a genuine interest in abstraction, or at least abstract ideas. Lovelace saw more to Babbage's Analytic Engine since she developed what is considered the first computer program. Lovelace, though considered the first programmer, is also one of the first to conceptualize general-purpose computing and abstract data processing, from input to desired output.

The successor of Babbage and Lovelace was Alan Turing, who advanced computing to a new level with the development of the Turing machine. In 1936, over 100 years after the invention of the Analytic Engine, he proposed a device capable of executing any computable

function to establish a model for processing, programmability, and the definition of what is "computable". Turing had contributed great things ultimately to computing and computer science with this model. According to Lara et al. (2022), though Turing was not the only contributor nor the first contributor, given Charles Babbage, Ada Lovelace, and many others, Turing remains the most influential in both scope and depth due to the universality and sustained relevance of the model.

As the early foundations of computing and computer science were coming to an end, and early practical machines were coming to be, one of the most significant developments was the ENIAC, the Electronic Numerical Integrator and *Computer*, built by John W. Mauchly and J. Presper Eckert. Built during the Second World War, ENIAC was designed to calculate artillery trajectories using varying conditions, which was a major leap from conceptual machines like the Difference and Analytic Engine to a fully functional computing device. Though no explicit record, Mauchly and Eckert, like many others at the time, were influenced by ideas of predecessors such as Babbage, Lovelace, and Turing, whose contributions set the foundation for their achievements, which ultimately received buy-in from the United States Department of Defense (Barton, 2021).

Mainframe and Early Programming: 1950s-1960s

Following World War II, interest in computing grew quickly. The success of the ENIAC, which was massive, centralized, and a mainframe in spirit, led to a shift from experimental one-off machines to the development of mainframe computers and the coming of programming as an abstract principle in computer science. IBM launched the first commercially available mainframe in 1952 and went on to make notable models, such as the IBM System/360 in 1964 (Webb, 2021). According to Crutcher, Singh, and Tiegs (2021), many programming languages have been

developed since the 1950s, which evolved from low-level assembly languages that communicated directly with the hardware of the machine.

The International Business Machines Corporation, or IBM, helped bring a new era of innovation, much like the contributions of Babbage, Lovelace, Turing, and others in the early foundations. With the release of the IBM 701 in 1952, the corporation became a leader in commercial computer sales, bringing success to the corporation while also contributing to the continued evolution of computing and computer science. As computing became more accessible around the world, IBM played an important role in its global expansion. Only five years later, engineers at IBM developed the first high-level language, FORTRAN (Formula Translation), which was used by developers to write programs using math notation (Anderson, 2025). This capability reflects a concept that Ada Lovelace envisioned more than 100 years earlier: an abstract system for instructing a machine, regardless of its original design. Anderson also notes that in the 1960s, an IBM computer made up 7 of every 10 computers across the world.

Personal Computing Era: 1970s-1980s

The 1970s and 1980s marked an inflection point in computing history as personal computers began entering homes, schools, and small businesses, alongside the early development of the networking between computers. In the 1970s, Intel developed the first general-purpose microprocessor, which later played a very important role in the creation of the IBM Personal Computer (PC), the first that was just the right size for personal use for the time (Crutcher et al., 2021). Around the same time, the Advanced Research Projects Agency Network (ARPANET) was established as the first operational packet-switching network, which is the precursor to the internet today.

Networking and Globalization: 1990s-2000s

The 1990s and early 2000s led the way for the age of the internet and open-source software, which had completely reshaped the way that computers were used, shared, or understood by the average enthusiast. Although not as widely recognized as, say, Steve Jobs or Bill Gates, who were two of the time's most iconic names, Tim Berners-Lee made one of the most important contributions when he invented the World Wide Web in 1991. The World Wide Web changed global communication and access to information, which earned him the Draper Prize in 2007, called "engineering's Nobel Prize" (Tim Berners-Lee Receives Draper Prize, 2007). With the World Wide Web came even more advancements in computing and computer technologies that would define the next era, from 2010 and beyond.

Modern and Emerging Technologies: 2010s-Present

Computer scientists and historians may debate which technological era we are currently living in, and both bring valid perspectives. What is certain, however, is that we are living in a very important time in the history of computing and computer science, one defined by modern and emerging technologies. Smart devices and their adoption are widespread and are now embedded into the daily lives of millions, if not billions, of people. In parallel, data generation and consumption have grown exponentially, with remarkable volumes of information being read and written every nanosecond. Literally, Microsoft SQL time-precision data types exist down to the nanoseconds to measure datetime value. Although artificial intelligence and machine learning are not new concepts, having been explored theoretically as far back as the 1950s, recent breakthroughs have made them central to contemporary computing. As Zarei et al. (2024) reaffirm in their case report, these technologies are ever so integrated into real-world applications. What began as a pursuit of improving computation has evolved to exactly where we

are today, in a field that has shaped many facets of modern life, which only underscores computing and computer sciences responsibly guiding the future.

Computational Thinking Principles – Then and Now

Abstraction

Generally speaking, abstraction is the process of identifying key features within a usually complex system and representing those features as generalized, higher-level concepts. In computing, this allows systems to be understood and manipulated without requiring low-level knowledge of their internal mechanics. For example, a computer can be seen in terms of processing power and input/output capabilities, abstracting away the circuitry or machine code that underlies these functions. The principle exists to aid developers in building layered systems and software to interact with the hardware through high-level interfacing rather than dealing with the low-level operations directly.

Historically, abstraction was central to Ada Lovelace's contributions to Charles Babbage's Analytic Engine, theoretically since the engine was never created. Babbage saw the machine as a tool to calculate numerical tables, but Lovelace wrote instructions, effectively what was the first computer program, that demonstrated the potential for general-purpose computation. Lovelace's ability to abstract the machine's functionality from its original theoretical design is often credited with establishing the foundation of programming today. Today, software developers apply a very similar principle by writing applications with higher-level functionality without demanding complete knowledge of the system at a low level. Similarly, abstraction is done; for example, in object-oriented programming, when a class is

created for a guitar, and that class has a function for playing a tune, you do not need to know how to perfectly tune the strings, though your ears may appreciate it.

Decomposition

Decomposition is the process of breaking down complex problems into smaller, more manageable pieces. Early on, this principle was shown in Alan Turing's design of the Turing Machine. In his 1936 paper, *On Computable Numbers*, Turing uses decomposition to represent computation as a set of steps starting by reading a symbol, changing state, writing a symbol, and moving a tape head (Turing, 1936). When combined, these steps are like any computable process. Today, programmers, whether they are aware of Turing's influence or not, continue to apply this principle by writing modular code, which divides large systems into functions, methods, or classes to make complex software systems easier to build, maintain, and understand.

Algorithm Design

Closely tied to both abstraction and decomposition is the principle of algorithm design. One of the earliest known examples comes from Ada Lovelace when she designed an algorithm for calculating Bernoulli numbers, intended for Charles Babbage's Analytic Engine; Her method of doing so on a machine that was never created designed an algorithm that was designed for a machine that was not hers. The algorithm, though purely theoretical, was important for defining a step-by-step process that a machine could follow to compute a desired result from a given input.

Today, algorithm design remains an important aspect of computer science, supported by high-level programming languages such as Python. Languages like these provide tools to help modern developers create algorithms that are far more expensive than what was possible in

Lovelace's time. By abstracting away low-level operations, developers can focus on the design of an algorithm rather than the hardware involved.

Automata, Algorithms, and Programs

Automata, algorithms, and programs form the backbone of computer science. The three concepts are the organs in the systems that power modern smart devices, software, and computing models. Any smart device that one wears or uses, such as a smartwatch or smartphone, can be seen as a form of automaton, responding to different inputs by transitioning between defined internal states. These transitions are administered by algorithms, which are structured by the steps designed to produce outputs. Together, automata and algorithms are dressed as programs, which execute the logic necessary to perform tasks desired by users. By both theory and application, automata, algorithms, and programs last due to their research and effective real-world daily use.

Automata

Automata are theoretical models that are used to describe different forms of computation. An automaton operates through states, each representing a distinct configuration of a system at a given moment. Transitions define the rules by which an automaton moves from one state to another, typically, these are designed at the beginning with a start state and proceeding to an accept state, assuming valid input. In his 1936 paper *On Computable Numbers,* Alan Turing introduced the concept of a machine capable of performing computation through a finite set of states and transitions, an idea to progress automata theory (Turing, 1936). Turing's work is used in the field today, for example, with applications that use automata in areas such as the design of e-commerce websites (Sai Nikhil et al., 2022).

Algorithms

Algorithms are well-ordered steps designed to solve a problem or achieve a desired outcome. While usually associated with computing, the concept of an algorithm predates computer science; for example, the steps a person takes each morning to arrive at work on time can be viewed as a type of algorithm, where repeated inputs yield a desired result. In computing, algorithms function similarly but operate within formal systems, ranging from narrowly focused tasks, such as calculating Bernoulli numbers, to more complex processes like those found in artificial intelligence. In modern applications, algorithms simulate tasks that were once performed by humans; even a ‘computer’ was once the job title for a human.

Programs

Programs are structured sets of instructions written to implement algorithms and support the behavior of automata. Theoretically, programs represent concealed logic that automata follow and those algorithms define, serving as the bridge between abstract concepts and machine-level operations. Early programs were written in low-level machine or assembly languages, which required an extensive understanding of the hardware used. As programming languages evolved, abstraction increased significantly by allowing developers to write more efficient code. Today, programs range from simple scripts, such as those used to launch a user’s favorite music app when a computer starts, to complex and layered applications built using libraries and frameworks developed by others. Despite the advancements in these tools and languages, all programs operate on the same principle: to bridge algorithmic logic and machine performing the task.

Conclusion

From early concepts to the intelligent systems that exist today, computing and computer science have relied on the ongoing interconnectedness of theory, application, and innovation. Figures such as Babbage, Lovelace, and Turing, alongside many others whose contributions also may go unrecognized, continue to influence current practitioners whether they are aware of it or not. Understanding abstraction, decomposition, algorithm design, and the history thereof as both historical and contemporary principles is key to appreciating the evolution of the field. As computing continues to advance, these advanced systems still rely on core concepts from generations ago. The research of tomorrow is built upon the knowledge written and refined today.

References

Anderson, D. (2025). International Business Machines Corporation (IBM). *Salem Press Encyclopedia*.

Barton, J. (2021). High Performance Computing for Science and Engineering in the Department of Defense. *Computing in Science & Engineering*, *Comput. Sci. Eng*, *23*(6), 58–62. https://doi.org/10.1109/MCSE.2021.3112288

Crutcher, P. D., Singh, N. K., & Tiegs, P. (2021). Programming. In *Essential computer science*. Apress. https://doi.org/10.1007/978-1-4842-7107-0_2

Lara, J. A., Pazos, J., de Sojo, A. A., & Aljawarneh, S. (2022). The Paternity of the Modern Computer. *Foundations of Science*, *27*(3), 1029–1040. https://doi.org/10.1007/s10699-021-09797-y

Sai Nikhil, M. M. V., Sarrin, A., Nair, G. S., & Supriya, M. (2022). *Design and implementation of e-commerce website using automata theory*. In *2022 6th International Conference on Trends in Electronics and Informatics (ICOEI)* (pp. 957–963). IEEE. https://doi.org/10.1109/ICOEI53556.2022.9777191

Strawn, G. (2023). Masterminds of Computer Design: Charles Babbage and Ada Lovelace. *IT Professional*, *IT Prof*, *25*(4), 7–10. https://doi.org/10.1109/MITP.2023.3297386

Tim Berners-Lee Receives Draper Prize. (2007). *Engineers Journal*, *61*(1), 38.

Turing, A. M. (1936). On computable numbers, with an application to the Entscheidungsproblem. *Proceedings of the London Mathematical Society*, *s2-42*(1), 230–265. https://doi.org/10.1112/plms/s2-42.1.230

Webb, C. (2021). Microprocessor Advances and the Mainframe Legacy. *IEEE Micro*, *Micro, IEEE*, *41*(6), 68–70. https://doi.org/10.1109/MM.2021.3115871

Zarei Ardestani, Z., Mao, E., & Krueger, R. (2024). Demystifying artificial intelligence for the global public interest: establishing responsible AI for international development through training. *Journal of Integrated Global STEM*, *1*(2), 142–152. https://doi.org/10.1515/jigs-2024-0013