Wasiak, P. (2023). Rejection of learning how to code and the problem of 'non-use' in the history of computer cultures. medien & zeit, 38(1), 32–42.

Rejection of learning how to code and the problem of 'non-use' in the history of computer cultures

Patryk Wasiak
Institute of History, Polish Academy of Sciences

Abstract

This paper investigates how a host of social actors, such as computer science experts and educators, discursively constructed both positively valued 'user-programmers' and negatively valued ‚non-programmers,' that is computer users who reject the practice of writing programs on their computers. I argue that the central theme of such a strategy was user agency and the question of having control over the technology that one is using in everyday life. Firstly, I investigate two key themes of the discursive construction of non-programmers in the era of the microcomputer of the 1980s, the discourses towards economies and social development related to computer literacy programs, and next, the key role of programming as a developmental tool for children's education. Later, I compare that historical era with the contemporary 'learn to code' movement and investigate how it outlines the disadvantages of the neglect of learning programming.

Keywords: *non-use, programming, computer culture, retroprogramming, computer literacy, LOGO, learn to code*

This paper aims to investigate how a host of social actors, primarily computer science experts and educators, discursively constructed negatively valued "non-programmers," that is computer users who reject the practice of writing programs on their computers. This is a longitudinal study that seeks to compare the cultural logic of the construction of this specific form of "non-use" of computers in two historical settings. The first setting is the era of microcomputers of the 1980s with the emergence of the popularity of BASIC and LOGO programming languages accompanied by the mass market for books and magazines dedicated to the learning of coding skills. The second setting is contemporary computer culture with the 'learn to code' movement that pathologies the rejection of learning how to write program code in the époque of the convenience of the availability of instantly downloadable programs summarized by 'there is an app for that' slogan.

While discussing these two historical settings I aim to highlight both continuity and change between the campaigns for the popularization of programming among computer users in the 1980s and the contemporary "learn to code" movement. To do so, I investigate how three campaigns were structured with the key themes and strategies of convincing computer users how they will benefit by learning coding skills. Such an investigation also highlights the discursive construction of those who do not learn how to code as irrational and unwilling to learn new valuable skills related to the use of digital technologies. It helps to better grasp the historical trajectory of the social imaginary of the digital divide that is regularly described only as a recent development in the Internet era (Warschauer, 2003).

Digital studies scholar David Golumbia noted that "The computer encourages a Hobbesian conception of this political relation: one is either the person who makes and gives orders (the sovereign) or one follows orders" (Columbia, 2009, 224). My paper will illustrate how the discursive construction of practices of non-use of computers as tools for programming, and 'giving orders' to computers became embedded in the public imagery of computer use.

I argue that in all three discourses, the central theme was the user agency and the question of having control over the technology that one is using in everyday life. I will discuss how both discourses in the 1980s were structured upon the utopian imagery of the positive impact of computers on personal lives and economies if computer users would exercise their agency by learning how to control computers by programming them. Differently, the contemporary 'learn to code' movement's discourse is structured upon the anti-utopian vision of the dominance of actors that form the economy of digital capitalism and exert control over the lives of computer users who lack knowledge, that is coding skills, necessary to resist them.

This is also an exploratory paper and I intend to elaborate on the problems of studying how to research computer "non-users." Various forms of computer use are linked with the emergence of easily recognizable cultural identities such as a hacker, gamer, or a colloquial 'computer nerd.' Differently, except for contemporary 'digital detox' campaigns, the non-use of computers usually does not stimulate the emergence of such identities. For that reason, Eric Baumer and colleagues in their paper on technology non-use have a problem with how to refer to those non-users. As they note, "the *non* prefix seems ill-suited. *Non-hackers*? *Non-players*?" (Baumer, et al., 2015b). For the sake of clarity in this article, I will use a similar 'non-programmer' term.

Aside from investigating a historical trajectory of the case of "non-use" related to computerization, I aim to contribute to *medien & zeit* special issue by highlighting the lack of scholarly interest and theoretical considerations of the 'non-use' and 'computer laggards' in the history of computer cultures. The history of computing is dominated by the innovators, such as computer science professionals, hardware and game designers, and early adopters such as hackers, computer hobbyists, and individuals who innovatively appropriated computers in a variety of professional settings. However, little attention has been paid to the late majority and even less to the laggards.

In my investigation, I draw from the interpretative framework of studies on the figure of technology user in Science and Technology Studies, such as the seminal *How users matter?* edited volume (Oudshoorn & Pinch, 2003) and media studies. Such a toolset helps me to deconstruct key elements of the repertoire used by social actors who publically argued about the benefits of being a user-programmer for both personal development and social and economic progress. Such a positively valued model of the future was juxtaposed with the imagery of possible future perils related to the negligence of the development of coding skills by computer users.

My paper is structured as follows. First, I make a brief review of relevant works that help to understand the cultural logic behind the non-programmer figure. I will also discuss why not only such a figure but more broadly, the non-use of computer technologies has been overlooked in the scholarship on the history of computing. In the next sections, I will investigate two key themes of the discursive construction of non-programmers in the era of the microcomputer of the 1980s, that is the discourse towards economies and social development related to computer literacy programs, and next, the key role of programming as a developmental tool for children education. Finally, I will discuss the contemporary 'learn to code' movement and investigate how it outlines the disadvantages of the neglect of learning programming.

Computer non-users and computer history

Not only the history of computing but also the broader field of history of technology traditionally share the same focus on technological innovation and those who design and commercialize new technologies. Only recently have scholars turned their attention to technology users and diverse practices of using, or possibly rejecting to use technological innovations. The seminal edited volume *How users matter* (Oudshoorn & Pinch, 2003a) includes not only theoretical consideration on practices of technology use but also two papers on the rejection of using technologies such as telephones in the early Twentieth Century United States (Kline, 2003) and a study on the rejection to use the internet (Wyatt, 2003). Nelly Oudshoorn and Trevor Pinch in the introduction to the volume express their agenda: "One important research question addressed in this book is how users are defined and by whom" (Oudshoorn & Pinch, 2003b, 2). As this *medien & zeit* issue shows there is still an urgent need to supplement such an agenda by asking how non-users are defined and by whom.

The history of computing is still primarily focused on researching those who pioneered technological innovation, both technology designers and a range of actors that played a role in the early stages of the dissemination of such an innovation. Historian of computing Patricia Galloway in her theoretically oriented paper discusses "Inventor-Early Adopter Dialectic" (Galloway, 2011). This title summarizes the dominant theme of historical studies of computing which focus on these two groups. Firstly, the inventors such as those who design and commercialize computers and to a lesser extent software (Campbell-Kelly & Aspray, 2004; Ceruzzi, 2003). Secondly, early adopters who appropriate technologies in professional and household settings, or a broad category of 'hackers' (Alberts & Oldenziel, 2014). However, for now, there is still little research on further groups from the diffusion of innovation model: early majority, late majority, and particularly laggards.

Aside from a range of contemporary studies on the 'digital disconnection' particularly the rejection of social media (for instance, Moe & Madsen, 2021; Woodstock, 2014; Goodin, 2017; Hesselberth, 2017), there are virtually no historical studies on non-use of computers. The closest historical work that addresses the social choices of using or rejecting computers in professional and home environments is a monumental edited volume by Rob Kling (Kling, 1996a). Contemporary cultures of non-use leave some traces such as manifestos and personal testimonies over social media detox. There are two reasons for the lack of virtually any historical sources which were produced by non-programmers as well as any documents that explicitly address them. As Baumer and colleagues note, cultures of use can influence the emergence of some cultural identities: "Moving beyond the individual, the voluntary non-use of technology may function as the production or performance of a particular sociocultural identity" (Baumer et al., 2015a). Differently computer laggards who were not interested in the adoption of computers in their professional and private lives did not produce a similar identity. As Wyatt notes, studying non-use poses a particular problem due to the lack of sources but also the inability to apply the classical paradigm of social sciences:

> *"Non-users may not be a very cohesive group as people may have very different reasons for not using the Internet. This invisible group is another instance of the difficulties posed by an over-literal interpretation of the dictum to 'follow the actors.'" (Wyatt, 2003, 78)*

For that reason, we can only investigate how other social actors defined users and not-users according to their agenda. According to a historian of science Adele Clarke, in such a case both users, and particularly non-users, are 'implicated actors.' As Clarke and colleagues note:

> *"**Implicated actors** are actors silenced or only discursively present in situations. In discourse data, they are usually constructed by others for others' purposes. There are at least two kinds. The first, while physically present, are silenced, ignored, or made invisible by those having greater power in the situation. Second are those **not** physically present but **solely** discursively constructed by others, usually disadvantageously. **Neither** kind of implicated actor is actively involved in self-representation." (Clarke et al., 2015, 16, cf Oushdorn and Pinch, 2003, 6.)*

In my case, 'non-programmers' were obviously present since only a small percentage of computer users wrote even a rudimentary program. However, they were silenced by those who shaped the public discourse on programming. The most suitable term to discuss the host of social actors who took part in such construction is 'the network of technology promoters' (Rip & Talma, 1998, 313).

As I will discuss in subsequent sections, the discourse towards the necessity of using computers to learn how to write programs, and the relevant silencing of 'non-programmers' was shaped by the key theme of user's control over technologies they use. While discussing the issue of control, I will refer to the imagery of technological utopianism and anti-utopianism. As Rob Kling insightfully summarizes the core elements of both imaginaries:

"Technological utopianism does not refer to a set of technologies. It refers to analyses in which the use of specific technologies plays a key role in shaping a Utopian social vision, in which their use easily makes life enchanting and liberating for nearly everyone. In contrast, ***technological anti-utopianism*** *examines how certain broad families of technology facilitate a social order that is relentlessly harsh, destructive, and miserable." (Kling, 1996b, 42).*

Later I will highlight how such imagery was used as a point of reference in all three historical cases.

Computer literacy projects of the 1980s

The key moment for the cultural history of computing in the 1980s was the shift from the computer as a professional device located and used in professional, institutional, or scientific settings into a home technology. Such a shift was accompanied by a substantial effort to shape a new computer user according to guidance by a network of technology promoters: hardware and software manufacturers, computer science professionals, and educators. Historians of computing Tom Lean and Alison Gazzard in their studies of popular computing in Great Britain in the 1980s grasped the cultural logic of such public campaigns towards raising awareness of the need for mass 'computer literacy' (Lean, 2016; Gazzard, 2016). Home computers such as the Apple II, Commodore 64, and ZX Spectrum were extensively used to play computer games and run available software. However, technology promoters engaged in shaping the social imagery of the microcomputer emphasized the critical role of programming as a practice that has tremendous benefits not only for those who will decide to learn how to code but also more broadly for societies and economies in the years to come. There were some discussions on what exactly computer users should learn to succeed in the new economy. As Kling summarizes such debates on what exactly 'computer literacy' education should focus on:

"Must all effectively educated citizens have any special knowledge of computer systems? If so, what kinds of insights and skills are most critical, those that are akin to computer programming or those that are akin to understanding how organizational information systems function?" (Kling, 1996a, 13)

My research suggests that both programming and learning how to operate information systems such as databases were equally considered key skills for a computer literate person.
While a range of primary sources for the history of computing such as computer magazines, popular books, and television programmers offer extensive coverage of programming techniques and the benefits of programming, they very scarcely include any depictions of those who were not interested in coding. They only addressed those who 'do not code *yet*' with a selection of arguments on the purposefulness of learning how to write code. Going back to the aforementioned Kling's question, "Must all effectively educated citizens have any special knowledge of computer systems?", in the 1980s the publically accepted answer was clearly "yes, all educated citizens are obliged to have a degree of knowledge on computer systems" More specifically, such desired degree of knowledge included at least rudimentary coding skills.
It is necessary to mention the exception of the emerging gaming culture that to some extent legitimized gaming as a publically accepted form of computer use. As Graeme Kirkpatrick notes the 1980s was an era when game publishers and game magazine editors successfully carried out a campaign of "Making Games Normal" (Kirkpatrick, 2014, 2015). Computer magazines dedicated exclusively to computer games, as well as majority of computer press, recognized gaming as the legitimate sole purpose of using a computer at home. But evengaming magazines encouraged gamers to learn how to code and regularly included tutorials on how to modify available games or even encouraged readers to design and code their own games.
In the decade of microcomputers programming became also identified as a rudimentary form

of interaction with a computer: As Lean notes: "At the time programming was seen as key to developing a working knowledge of computers" (Lean, 2016). As an instance of how in the 1980s technology producers imagined programming as a natural form of computer use, I can bring the user's manual for the Commodore VIC-20, the predecessor of the Commodore 64. First, the user was instructed on how to connect the computer to the monitor and power supply. Just after that the manual presented a very simple program to type in and the user was instructed: "Try typing this program: type this program exactly as shown and see what happens!" (*Personal Computing on the VIC 20*, Commodore Electronics Ltd., 1982, 2). Aside from dedicated computer periodicals and books even magazines that had nothing to do with computers published short tutorials on programming in BASIC, and some public broadcasters introduced short-lasting television courses on BASIC programming.

Historian of technology Janet Abbate discusses how programming became identified as a form of social empowerment embedded in utopian visions of social change (Abbate, 2018). As she notes: "Coding was a path to intellectual awakening or immediate social goals" (138). She also remarks how learning how the network of computer technology promoters claimed how learning coding will address several social problems:

> *"programming skill has been variously constructed as a shift of power from management to labor, a means of economic uplift for minorities, or a thinking tool for children. I argue that coding initiatives have always been embedded in politics and that the specific types of power associated with computer skill have been tied to the social identities of coding proponents and their intended beneficiaries." (Abbate, 2018, 134)*

Abbate mentions several different campaigns that claimed an optimistic future for those who will make an effort to learn how to code. Here I would like to add that in the 1980s, programming became presented as a skill that will be necessary for most white-collar jobs (Kling, 1996a). Such campaigns usually did not explicitly mention what will happen to those who reject learning such a new skill. However, using the aforementioned summary by Abbate we may conclude that those who will reject it would not be able to control their life projects and still will be powerless labor, minorities devoid of opportunities for economic uplift, and children that did not learn how to think.

Technology promoters presented programming as a key skill that can help to find a place as a highly paid programmer. Here I would like to bring an example of how a utopian vision of programming skills was performed in commercial imagery. An advert for a software publisher in *Byte* magazine in 1981 presented a comic strip about a bored young white-collar worker with a passion for programming titled "How I made it big writing microcomputer software" (*Byte*, December 1981, 313). First, the protagonist of the strip complained "I have so much fun writing programs for my little micro [...] The only trouble is, I still have to get up at 7:00 and go to my boring job." After finding a publisher for his programs he was able to change his life: "As so, here I am today, on my newly purchased yacht somewhere off Greece." Those who do not learn how to program were missing out on such a lucrative opportunity. We may use this advert to learn that those who would not learn to program will still have to carry on their boring jobs instead of sailing their own yachts. As Lean summarizes, "Computer advertising explained that the information technology age was coming and that people risked being left behind if they did not adapt" (Lean, 2016).

The aforementioned instances of the discourse towards programming do not include specific references to the figure of the non-programmer. However, they clearly show that in the decade of microcomputers in the 1980s using computers for programming became a highly desirable and obvious form of computer use similarly to using computers to access the internet in the 2000s, which has been grasped by Wyatt as "a worldview in which adoption of new technology is the norm" (Wyatt, 2003, 78). Despite the availability of such knowledge, only a minor part of computer users became actually interested in coding. Referring to Clarke and colleagues this is an instance when social actors while physically present, were ignored and made invisible by those who had an impact on shaping public imagery of computing (Clarke et al., 2015, 16).

Programming as children's development

The aforementioned campaign for computer literacy equally emphasized both intellectual and practical elements of learning how to program computers. Here I would like to focus on an accompanying campaign of programming as a way of revolutionizing children's development. The imagery of 'child-programmer' was an instance of how learning programming was a way of reaching an 'intellectual awakening' for children (Abbate, 2018, 138). Generally, children were identified as those who particularly easily and naturally acquire computer skills. For instance, the *Byte* magazine editor pictured the desired vision of how adolescents should interact with computers: "a typical high school student could use computers to write compositions, memorize facts and vocabulary, understand relationships and concepts in mathematics and science, and write computer programs." (*Byte*, February 1987, 149)

American *Family Computing* magazine regularly published reports on 'computing families." In one of such stories we can find what parents expect from their children: "Both Tony and Penny Morris are obviously pleased that their kids program their own games (fairly simple ones), or at least can if they want to." (Frenkel, C., How to Program Success Into Your Computer, *Family Computing*, September 1983, 46).

Referring to Wyatt (2003, 78), this is another instance of a normative model of computer use. Both aforementioned sources inexplicitly shape pathological imagery of children non-programmers, as those who will not benefit from the educational opportunities offered by computers and also disappoint their parents.

The most notable case for a normative approach toward children-programmers can be found in works by mathematician and educator Seymour Papert who developed the LOGO program language at the MIT inspired by Jean Piaget's cognitive development theory. His most influential work is *Mindstorms. Children, Computers, and Powerful Ideas* (1980) in which Papert presented his concept of using computers in children's development drawing from the philosophical dichotomy of being controlled by technology/controlling technology. As he noted in *Mindstorms...*, in the ordinary educational environment "The computer programming the child" (19). Differently, "In the LOGO environment the relationship is reversed: The child, even at preschool ages, is in control: The child programs the computer" (19). Papert did not elaborate on what exactly means that the computer is programming the child, but here we can see a discursive construction of a child non-programmer as someone who does not have his or her agency and simply follow orders. As we may assume simply using available software meant that the child passively follows orders given by those who designed such software. Lean in his book discusses the impact of Papert's work on using computers in education:

> *"In common with microcomputing in general, programming was an important part of school computing in the 1980s. Educational computing articles of the time have an underlying rational of empowering children, an idea that they should program computers rather than be programmed **by** computers." (Lean 2016)*

In Papert's work we may find several claims that programming is natural for children:

> *"When I have thought about what these studies mean I am left with two clear impressions. First, that all children will, under the right conditions, acquire a proficiency with programming that will make it one of their more advanced intellectual accomplishments. Second, that the 'right conditions' are very different from the kind of access to computers that is now becoming established as the norm in schools." (1980, 16)*

Similarly, as another educator engaged in the LOGO project claimed, offering children access to LOGO is like "leading fish to water" (Higginson, W., Leading Fish to Water, *Byte*, August 1982, 328). Practical explanation of how Papert's educational program works can be found in highly influential work by Sherry Turkle (1984). She studied how a small group of "child programmers" took part in computer courses by using BASIC and LOGO under the supervision of MIT educators. Turkle's work provides a classical study for human-computer interaction and educational and philosophical studies. However, the children from her study definitely didn't come from an average American school and her study does not situate them in any social, cultural, and economic contexts of the United States of the early 1980s.

It is important to note that other studies on the history of computer education in schools are frequently written by educators themselves (for instance Tatnall & Davey, 2014). They primarily investigate the use of computers in teaching computer science and programming. However, such investigations are mostly biased since the authors enthusiastically discuss the successes of such educational campaigns and do not address children who struggle to learn to program or declare their lack of interest in their subject. A much more down-to-earth study of computers in education has been written by Larry Cuban (2001) and the title of his book: *Oversold and Underused. Computers in the Classroom* tells much about the realities of computer classrooms.

Cuban in his work discusses a key issue relevant to the use/non-use dichotomy. Previously, Turkle in her influential study showed a classroom environment where computers are at the same time physically present and used exactly as intended by educators who designed computer education curriculum based on Papert's work. Cuban, drawing from his fieldwork in the educational system in the US showed that the physical presence of computers in classrooms does not automatically guarantee that computers will be used as proponents of computer education expect to.

Similarly, in more recent work on the much-hyped One Laptop Per Child project Morgan Ames (2019) shows how MIT educators and policy-makers from the Global South expected that simply providing every child in the Globa South with access to a laptop would solve a range of educational, social and economic problems. Both Cuban and Ames articulate the issue of the "non-use" of computers that are physically available in educational environments. For both authors "non-users" are not silenced actors (Clarke et al., 2015, 16,). Instead, they offer complex investigations into why some educational programs shaped by technology enthusiasts do not work as intended.

Such an approach significantly differs from works by computer science educators. Such studies on computer use in education usually do not include any elaboration on those who for some reason do not embrace computer science teaching offered in educational systems (Tatnall & Davey, 2014). Such works imagine non-users as late majority or laggards that simply require more effort from educators to successfully evolve into "users."

'Learn to code' movement

Both discursive constructions of programmers and relevant silencing and ignoring of non-programmers discussed above have been strongly influenced by the utopian visions of the positive impact of computers on the future along with the promises of "making life enchanting and liberating for nearly everyone" (Kling, 1996b, 42). Differently, the contemporary 'learn to code' movement which emphasized the need to learn how to write code is still equally driven by a utopian and anti-utopian vision of the positive impact of computing on the future. If we look at the arguments of those who promote the movement's objectives, we see a much a dark vision of computing in the contemporary world. As I will discuss further, such a vision includes non-programmers.

The 'learn to code' movement is a prominent part of contemporary computer culture. Computers are ubiquitous and virtually everyone has some basic knowledge of using computers to run programs and browse the internet, but only a small percentage of users ever learned at least rudimentary coding skills. In such a context 'learn to code' movement identifies the ability of programming as a forgotten knowledge and identifies the problem of the convenience of 'there is an app for that' (Miller and Matviyenko, 2014) culture that results in the dominance of the model of casual computer use and the neglect

of understanding of the role of software as the backbone of contemporary technological infrastructures.
The most prolific member of this movement is Code.org (https://code.org/), a nonprofit that, according to its website is "dedicated to expanding access to computer science in schools and increasing participation by young women and students from other underrepresented groups." This website also prominently includes slogans such as "Learn computer science. Change the world." and "Learn today, build a brighter tomorrow." In 2013 during the 'Hour of Code" initiative, the movement received an influential celebrity endorsement by president Barack Obama, who became 'the first president-programmer' (Finley, 2014). In his public speech Obama directly confronted 'there is an app for that' culture': "Don't just buy a new video game, make one. Don't just download the latest app, help design it" (President Obama asks America to learn computer science, uploaded by Code.org, https://www.youtube.com/watch?v=6XvmhE1J9PY).
The aforementioned slogan "build a brighter tomorrow" clearly suggests that our contemporary world is not so bright since most computer users do not know how to code and do not know how the software they use every day works. As software studies scholar Wendy Chun put it "Knowing software, ... enable us to fight domination or rescue software from "evil-doers" such as Microsoft" (Chun, 2011, 21). And what exactly digital 'evil-doers' do, can be summarized by the title of digital studies scholar Jathan Sadowski's book: *Too Smart. How Digital Capitalism Is Extracting Data, Controlling Our Lives, and Taking Over the World* (2020). Such imagery corresponds with remarks by Kling on the dark side of computerization:

> *"Much less frequently, authors examine a darker social vision when any likely form of computerization will amplify human misery—people sacrificing their freedom to businesses and government agencies, people becoming dependent on complex technologies that they don't comprehend, and sometimes the image of inadvertent global thermonuclear war." (Kling, 1996b, 41)*

The movement perceives learning how to understand and write software as a method of challenging the domination of the digital economy potentates and empowering individual users. As Abbate notes: "Code.org began to fuse the concerns of corporate interest, education, and social justice into a single discourse equating coding with empowerment (Abbate, 2018, 147). Going back to the previously quoted remark by Wyatt, the movement identifies the fact that the adoption of technology, that is programming, is not a norm (Wyatt, 2003, 78) as an acute social and economic problem. As a remedy, the movement intends to popularize the model of computer use in the 1980s and bring back programming 'as key to developing a working knowledge of computers" (Lean, 2016).
Interestingly, the 'learn to code movement is countered by a strong opposition of those who bring 'non-programmers' into the spotlight and explain why mass learning how to write rudimentary code is not a cure for acute social and economic problems (Shapiro, 2016; Farag, 2016). Moreover, in the era of public knowledge of the exploitation of employees in the digital economy, it is clear that programming would not guarantee a yacht, only plausibly a poorly paid job in an open space with compulsory overtime during 'crunch times.'

Discussion

In all three cases, I have discussed how a host of actors promoted not only computers as a technology of the future but also programming as a specific desirable form of computer use that will realize the full potential of technology. All three campaigns towards the need for learning how to write programs strongly resembles Wyatt's study of the Internet:

> *"Everyone is clearly understood as a potential user of the Internet. Access to the technology is seen as necessarily desirable, and increasing access is the policy challenge to be met in order to realize the economic potential of the technology" (Wyatt, 2003, 68).*

Going back to the question "how non-users are defined and by whom?" We have seen that such promoters of computerization did not explicitly define 'non-programmers' but rather made them silent and invisible (Clarke et al., 2015, 16). However, by investigating their visions of the benefits of the ability of programming and, thus, having control over one's life, we can conclude that 'non-programmers' will not be able to control their lives in the upcoming era of the information society.

All three historical cases shows how definition of use/non-use of technology was structured with a dichotomy of having control/being controlled. However, there is a key difference between the 1980s and the contemporary world. In the 1980s the enthusiasm towards programming was driven by the optimistic utopian vision of the future with promises based on the benefits of the skillful use of new technology. Differently in contemporary culture, the imagery of programming is equally structured with similar utopian visions of the possible future and anti-utopian vision of contemporary currents with digital economy 'evil-doers' (Chun, 2011, 21).

Finally, I would like to highlight the key difference between non-use of social media and 'non-programmers.' According to contemporary 'digital disconnection' campaigns, those who use social media are being controlled by those who design such technology with specific ways of forming an addiction to such use and applying non-transparent algorithms. As Baumer and colleagues note: "non-use may represent an individual's attempt to regain (a sense of) self-control over their own technology use. [...] In many of these cases, the discourse is one of control." (Baumer et al., 2015a).

In all three historical cases discussed in my paper, the situation was reversed. Actors who shaped the discourse equaled 'use,' that is learning how to program, with having control over technology. Contrary, 'non-use' was equaled with the situation when technology, or some malevolent social forces, have control over computer users.

Finally, I would like to highlight the lack of scholarly investigations on how social actors perceive not only the use/non-use dichotomy but rather a hierarchy of different forms of using digital technologies. There is extensive literature from game studies scholars on the history of controversies over gaming as a legitimate form of using digital technologies (Kirkpatrick, 2014, 2015; Madigan, 2016; Kowert & Quandt, 2015). However, those studies focus primarily on the issue of legitimization, a single form of computer use. We need more works that would offer investigations on a continuum between programming as an idealized purposeful and creative form of computer use and non-use.

References

Abbate, J. (2018). Code Switch: Alternative Visions of Computer Expertise as Empowerment from the 1960s to the 2010s. *Technology and Culture*, 59 (1), 134-159.

Alberts, G., & Oldenziel, R. (Ed.). (2014). *Hacking Europe. From Computer Cultures to Demoscenes*. Springer.

Ames, M. (2019). *The Charisma Machine. The Life, Death, and Legacy of One Laptop per Child*. MIT Press.

Baumer, E., Morgan G. Ames, M., Burrell, J., Brubaker, J., & Dourish, P. (2015a). Why study technology non-use? *First Monday*, 20 (11), 2 November 2015 https://firstmonday.org/ojs/index.php/fm/article/download/6310/5137

Baumer, E., Morgan G. Ames, M., Burrell, J., Brubaker, J., & Dourish, P. (2015b). On the importance and implications of studying technology non-use. ACM *Interactions, XXII, 2,* March-April 2015, https://interactions.acm.org/archive/view/march-april-2015/on-the-importance-and-implications-of-studying-technology-non-use

Campbell- Kelly, M. & Aspray, W. (2004). *Computer: A History of the Information Machine.* Westview Press.

Ceruzzi, P. (2003 [1998]). *A History of Modern Computing,* 2nd ed. MIT Press.

Chun, W. (2011). *Programmed Visions Software and Memory*. MIT Press.

Clarke, A., Friese, C., & Washburn, R. (Ed.). (2015). *Situational Analysis in Practice. Mapping Research with Grounded Theory*. Left Coast Press.

Cuban, L. (2001). *Oversold and Underused. Computers in the Classroom*. Harvard University Press.

Farag, B., (2016). Please don't learn to code, TechCrunch website, May 11, 2016, https://techcrunch.com/2016/05/10/please-dont-learn-to-code/).

Finley, K. (2014). Obama Becomes First President to Write a Computer Program, *Wired*, Dec. 8 2014, https://www.wired.com/2014/12/obama-becomes-first-president-write-computer-program/)

Galloway, P. (2011). 'Personal Computers, Microhistory, and Shared Authority: Documenting the Inventor-Early Adopter Dialectic. *IEEE Annals of the History of Computing*, 33)2), 60-74.

Gazzard, A. (2016) *Now the Chips Are Down: The BBC Micro*. MIT Press.

Golumbia, D. (2009). *The Cultural Logic of Computation*. Harvard University Press.

Goodin T. (2017). *OFF: Your Digital Detox for a Better Life*. Octopus.

Hesselberth P. (2017). Discourses on disconnectivity and the right to disconnect. *New Media & Society*, 20 (5), 1994–2010.

Kirkpatrick, G. (2014). Making Games Normal: Computer Gaming Discourse in the 1980s. *New Media & Society* 18 (8): 1439–1454.

Kirkpatrick, G. 2015. *The Formation of Gaming Culture: UK Gaming Magazines, 1981–1995*. Palgrave Macmillan.

Kline, R. (2003). Resisting Consumer Technology in Rural America: The Telephone and Electrification. In N. Oudshoorn & T. Pinch (Ed.). *How users matter: The co-construction of users and technology*. MIT Press, 51-66.

Kling, R. (Ed.). (1996a [1991). *Computerization and Controversy: Value Conflicts and Social Choices*. Academic Press.

Kling, R. (1996b). Hopes and Horrors: Technological Utopianism and Anti-Utopianism in Narratives of Computerization. In Kling, R. (Ed.). (1996a [1991). *Computerization and Controversy: Value Conflicts and Social Choices*. Academic Press, 40-58.

Lean, T. (2016). *Electronic Dreams. How 1980s Britain Learned to Love the Computer*. Bloomsbury.

Madigan, J. (2016). Getting Gamers. The Psychology of Video Games and Their Impact on the People Who Play Them. Rowman & Littlefield.

Miller, P. & Matviyenko, S. (Ed.). (2014). T*he Imaginary App*. MIT Press.

Moe, H., Madsen, O. (2021). Understanding digital disconnection beyond media studies. *Convergence: The International Journal of Research into New Media Technologies*, 27(6), 1584–1598.

Oudshoorn, N. & Pinch, T. (Ed.). (2003a). *How users matter: The co-construction of users and technology*. MIT Press.

Oudshoorn, N. & Pinch, T. (2003b). Introduction: How users and non-users matter. In: Oudshoorn, N. & Pinch, T. (Ed.). (2003). *How users matter: The co-construction of users and technology*. MIT Press, 1–25.

Papert, S. (1980). *Mindstorms. Children, Computers, and Powerful Ideas*, Basic Books.

Rip, A. & Talma, S. (1998). Antagonistic Patterns and New Technologies In Disco, C. & van der Meulen, B. (Ed.) *Getting New Technologies Together Studies in Making Sociotechnical Order*. De Gruyter, 299-322.

Sadowski, J. (2020). *Too Smart. How Digital Capitalism Is Extracting Data, Controlling Our Lives, and Taking Over the World*. MIT Press.

Shapiro, J. (2016). President Obama Wants Every Kid To Learn Coding--For All The Wrong Reasons, *Forbes*, January 31, 2016, https://www.forbes.com/sites/jordanshapiro/2016/01/31/president-obama-wants-every-kid-to-learn-coding-for-all-the-wrong-reasons/

Tatnall, A. & Davey, B. (Ed.) (2014). *Reflections* on the History of Computers in Education Early Use of Computers and Teaching about Computing in Schools. Springer.

Turkle, S. (1984). *The Second Self: Computers and the Human Spirit*. Simon & Schuster.

Warschauer, M. (2003). *Technology and Social Inclusion. Rethinking the Digital Divide*. MIT Press.

Woodstock, L. (2014). Media resistance: opportunities for practice theory and new media research. *International Journal of Communication,* 8, 1983-2001.

Wyatt, S. (2003). Non-users also matter: The construction of users and non-users of the Internet In: Oudshoorn, N. & Pinch, T. (Ed.). (2003). *How users matter: The co-construction of users and technology*. MIT Press, 67–79.

Research for this article was supported by the National Science Centre, Poland, grant 2020/37/B/HS3/03610. I would like to express my gratitude to the editors and two anonymous reviewers for their comments that helped me to revise and improve my manuscript.

Patryk Wasiak,
Dr., associate professor, Institute of History, Polish Academy of Sciences, holds MA titles in sociology and art history (Warsaw University) and PhD in cultural studies (Warsaw School of Social Sciences and Humanities). Former fellow of the Volkswagen Foundation, the Center for Contemporary History Potsdam, the Netherlands Institute of Advanced Study, and the Andrew W. Mellon Foundation. His research interests include the cultural history of the Cold War, history of home technologies and history of computing. Currently he works on the the history of amateur programming and neo-liberal socio-economic order in the 1980s. His current project is supported with a 4-year research grant from the National Science Centre of Poland. He has published articles in IEEE Annals of the History of Computing, International Journal of Communication, and History and Technology.