Repurposing Turing’s “Human Brake”

Marie Hicks
Duke University

In 1945, Alan Turing theorized one of the major stumbling blocks to computing technology’s potential, coining the term “the human brake.” The human brake slowed computing processes by delegating actions to human operators that should ideally reside within the capabilities of a machine and its programming.

Turing went on to outline a stored program computer that aimed to revolutionize the current state of the art. Holding programmatic instructions for data manipulation within the machine would vastly increase the rate of speed at which input and output could be run through the central processor. Eliminating “all that is done by the normal human operator” and transferring these functions to the machine, not only increased speed but also reduced the level of error.1

In the strictest sense, Turing’s human brake as then articulated is barely recognizable today as an impediment to computing speed and efficacy. But in a broader sense, the human brake has never left the field of electronic computing. It has become transformed over the course of its historical trajectory, becoming increasingly abstract rather than being outright eliminated.

This essay aims to provide a brief historical discussion of incarnations of the human brake throughout the 20th century. My hope is that by slightly stretching and reorienting the concept functionally, while trying to remain true to its theoretical roots, we can create a useful tool for rethinking aspects computing’s future as well as its past, as Turing did in 1945.

Starting from this materially based analogy may hold great benefits for understanding the increasingly abstracted work processes made possible by successive generations of data processing systems. The historical vignettes offered come from the British context, both because of the pioneering work done there and the focus of my research. I conclude with an attempt to relate the human brake concept back to modern American, and global, computing systems and solutions.

In the 1950s, early computers propagated into a wide variety of administrative uses in commercial and government offices, moving away from being used primarily in scientific and military operations. Arguably, labor interaction with the technology became increasingly complex and multifaceted as a result. As the expansion of electronic computing’s applications enabled it to replace electromechanical office automation, both its scope and purpose began to shift.

At that point, the human brake still existed, even in the realm of stored program computers. Their function-

ing, though ideally automatic, often required an enormous amount of behind-the-scenes operational troubleshooting. The earliest business machines were adapted for use by the British bakery, Lyons. Modeling them on the Cambridge EDSAC, the company engineered and programmed them in-house to tailor them to specific work processes; primarily inventory and payroll needs.2 Nevertheless, one LEO operator recalled, “the earliest machines were very temperamental... Some of the early applications needed a lot of nursing.”3

Clearly, the human brake had not evaporated with the advent of stored-program computing, though its contours had begun to change. Its presence was now much less a result of functional incapabilities engineered into the computer. Rather, intermittently faulty programs or hardware, combined with lagging peripheral design, continued to animate the concept.

The idea could also be expanded to encompass other kinds of “braking” effects. One clear-cut legacy of the human brake present in Britain throughout the 1960s involved requirements ancillary to machine operation and maintenance, but crucial components of the system nonetheless.4 Even when programs ran seamlessly, hardware was in good repair, and operators worked efficiently, machines might fail due to the built environment of machine rooms.

Recalling an overly-spectacular failure at LEO, operator Colin Hobson painted an image of a machine room barely recognizable under current definitions:

One very hot summer we were working, stripped to the waist, with all the windows open, when a plague of newly hatched black flies came in from the playing field opposite. They got everywhere and caused data failures when the card readers tried to read them.5

A host of similar infrastructure problems were regularly present. Dampness ruined input by deforming cards, improper electrical supply or grounding damaged machines and peripherals, rising humidity and lack of air-conditioning stopped programs before completion, and so on. Poorly designed built environments regularly hindered British systems’ functioning into the 1970s. The efforts to integrate office building design with the computing system in some ways parallels the earlier attempts to reduce variability and error by internalizing programs into memory.

At the London Central Electricity Generating Board, several IBM computers installed in the early 1970s (continued on p. 106)
presented another avatar of the human brake. An IBM 360/50 and 360/75 occupied one office in the CEGB, forming parts of a relatively new system called Attached Support Processor (ASP). To wring maximum efficiency from the machines, they were used in tandem, automated to perform different tasks in the processing chain. While the 360/50 did I/O, the 360/75 ran the programs. The purpose of ASP was to speed up program runs beyond what a human manually controlling the processing and I/O functions individually could achieve. Still, the system relied on operator input and scheduling to an extent. Indeed, operators competed to produce runs that set speed records for processing the data. That they did so shows the human brake still lurked in the increasingly automated system.

But does the human brake still exist today? Its clearest ancestry might be drawn to our current system and network administrators, but this trajectory fails to adequately capture the concept's utility for modern systems. Infrastructure problems and buggy software may likewise remain, but as an apologia of the human brake, these do not shed light on contemporary problems. Instead, perhaps we can use the concept to help think about the goals and pitfalls attendant on the rise of extensible systems and distributed programming. In these systems, the more clear-cut legacies of the human brake described above are all but moot for the user-programmer who designs software solutions with their aid. In extensible systems, like Amazon’s Elastic Compute Cloud (EC2), Oracle On Demand, and to a much lesser extent the distributed programming solutions beginning to be offered by Google Apps, the virtual computing environment offered begins to approach Turing's ideal system. Such delegation to system authority frees potential users from hardware constraints, rather than limiting them to particular possibilities.

With the increasing provisions for maximally extensible systems, available on demand to a variety of user-programmers who have the ability to load a custom environment and maintain root control, the human brake both as originally theorized and as derived through later historical examples is effectively being erased. This, however, brings us to the crux of the problem, and the reason that the human brake maintains conceptual utility more than 60 years later, even as it may seem to be disappearing.

At its heart, the question of the human brake is one of the proper role of machine authority as much as one of machine capability. Can we build these new systems in a way that actually allows us to take advantage of them as quickly as the technology will allow? Whereas we were once hindered by thinking beyond what systems could do, we now may find ourselves hindered by our inability to think as far as their full potential. Similarly, the very necessity of further specialization in distributed programming skill sets may reduce the utility of these new environments. A significant tension currently exists between creating systems with maximal ability to scale that must be programmed according to a distributed model, and continuing to build and improve upon systems that can be programmed traditionally. In turn, this raises questions about not only how to make programming more automated, but how automated it can or should become as new virtual machines become ever easier to create.

Originally, the human brake was the result of a lack in machines' capabilities to operate without the slowness of a human operator's intervention. Now, its cogency turns on the lack of human capability to effectively capitalize on extensible, scalable computing power that threatens to disrupt boundaries previously taken for granted. In some ways, it's fair to say that the human brake is now far more human than it ever was. If this is the case, then where might attempts to solve the problem of the human brake lead us next?

### References and notes


Contact Marie Hicks at meh20@duke.edu.

Contact department editor Nathan Ersenmger at annals-thinkpiece@computer.org.