Selected Papers of AoIR 2016:
The 17th Annual Conference of the
Association of Internet Researchers
Berlin, Germany / 5-8 October 2016

EMBEDDED DANGERS: THE HISTORY OF THE YEAR 2000 PROBLEM AND THE POLITICS OF TECHNOLOGICAL REPAIR

Dylan Mulvin
Microsoft Research New England

Introduction

Apple recently confirmed a rumor that a design flaw in their mobile operating system, iOS, would permanently disable an iPhone if a user set the date to 1970 or earlier (Kelly 2016). The glitch materialized an awareness that our devices only work within strict calendrical parameters. More than any other recent event, the Year 2000 problem (better known as the Y2K bug) established the public awareness of the temporal contingencies of embedded computer systems. This paper revisits the Y2K bug to see what lessons can be drawn from this (non)event. Using archival research conducted at the Charles Babbage Institute, this paper undertakes an analysis of the Year 2000 Problem and the large-scale practices of technological repair and management that addressed it.

I argue that the ways the Y2K bug was addressed set the groundwork for the large-scale infrastructural management of technological contingency in the early 21st century. I approach the organized response to the perceived threat of the Y2K bug as one of the greatest, public-facing attempts to educate and train individuals and organizations to manage the unforeseen, and potentially devastating, effects old computer code can have on contemporary computerized infrastructures.

I begin with a history of the bug's origins and proceed to examine three key effects of the crisis: 1) the massive resource investment and funding expenditures on computerized infrastructures that few other crises have compelled; 2) the changes to American insurance and tort law developed as a dimension of the crisis' legal repair; and 3) the proliferation of risk management training around computerized infrastructures, forming what I call a "pedagogy of preparedness." By studying these three effects, this paper reconfigures the role of Y2K in the history of computers, infrastructure, and information systems by placing the bug within the larger contexts of infrastructure renewal, public works, and technological literacy.

Suggested Citation (APA): Mulvin, D. (2016, October 5-8). *Embedded dangers: the history of the year 2000 problem and the politics of technological repair*. Paper presented at AoIR 2016: The 17th Annual Conference of the Association of Internet Researchers. Berlin, Germany: AoIR. Retrieved from http://spir.aoir.org.

Background

Leading up to the year 2000, news outlets, governments, and technological experts warned that the world might experience near-total computer failure due to a common coding practice originating from the 1960s. In this time, computer programmers chose to save memory capacity, a scarce resource at that time, by coding dates in six digits instead of eight: DD-MM-YY. The rollover to the year 2000 presented the very real possibility of global, systemic chaos as computers still running older software were forced to reckon with a year written simply as "00." To address the threat the coding bug posed, governments, corporations, community groups, and non-governmental organizations launched massive campaigns to prepare populations for systemic collapse and these campaigns generated new precedents for addressing future threats of this kind.

From the vantage point of the present, the Y2K crisis may look like a very costly false alarm, what some have called a "nutty cocktail of digital overthink and Luddite millennialism" (Menand 2015, 75). But from another perspective it is a significant moment of crisis planning and management, a population-wide event of computer literacy campaigning and technological repair. The threat of Y2K created acute awareness of computing and computer code as a fundamental part of basic infrastructure and the organized response to the Y2K crisis is one of the world's largest attempts at fundamental technological repair. The fact that the Y2K crisis was attached to a precise date and time that computer infrastructures could be expected to fail made it a unique event in the public discourse surrounding computer technologies in the late 20th century—what Paul Edwards called, "a unique opportunity to document and study the course of a technological crisis before it actually occurs" (Edwards 1998, 26)

The history of the Y2K problem demonstrates how technological creation and innovation emerge from major sites of potential failure. Few studies of technological failure, however, historicize the discrete practices of planning, repair, and management around technological crises. As such, this paper is a first attempt to reveal how practices of technological repair structure the relationship between software programming, massive social and technological infrastructures, and the institutional configurations that shape, transform, re-direct, and reproduce them.

Theoretical approach

In this paper, I conceptualize the Y2K bug as both a problem of potential infrastructural collapse (Edwards 1998) and a problem of educating the public on infrastructural inter-dependence. I approach repair as a central component of technological history. And, finally, I approach liability and tort law as a part of the domain of repair operations that policy makers use to mend the social damage threatened by technological failure.

I approach computer code as a fundamental part of contemporary information infrastructures and builds on existing research in this area. In addition to this focus on infrastructure, I also consider repair and maintenance as central components of

technological history. Some recent work examines the repair and maintenance necessary for any technology or infrastructure to operate properly (Henke 2000; Graham & Thrift 2007; Jackson et al. 2012; Jackson 2014). By studying maintenance and repair operations these scholars aim to understand how "order and meaning in complex sociotechnical systems are maintained and transformed" (Jackson 2014, 222) and the need to repair and reuse existing materials is a necessary solution to imminent environmental collapse. This paper offers a new consideration of maintenance and repair by uncovering the importance of public pedagogy in technological remediation.

Finally, I consider the social effects of infrastructural repair through an examination of the Y2K bug's legal consequences. To do this, I draw upon socio-legal studies and use legal literature on insurance and tort law related to the Y2K crisis, which was a much discussed and written about topic of the late 1990s (Kerr 1998; Jinnet and Greene 1999; Williams and Smyth 1999). As Y2K remediation efforts accelerated in the 1990s, the legal ramifications of embedded computer failure came into focus as "lawyers [awakened] to the same sense of ubiquity, unpredictability and entanglement that engineers have long seen as the core of the Y2K problem" (Dunn 1998).

Conclusions

The Y2K bug began as a design choice: the decision by early computer programmers to code four-digit years using only two digits. Though the problem affected many early programs, the Common Business Oriented Language (COBOL) was often singled out as the source of vulnerability when the bug became a public issue in the 1990s. COBOL was a coding language that grew out of the late 1950s' "software turmoil" and the apparent need for a simplified, shared language in the face of growing chaos among computer hardware manufacturers (McCracken 1962; Ceruzzi 2003; Ensmenger 2010). Out of this choice to compress dates in COBOL, however, the Y2K bug publically surfaced the interdependency of various systems in the 1990s (Fisher 1999; Gordon 1999). Drawing on archives of technical literature as well as governmental and non-governmental risk estimates, I reconstruct the Y2K bug's apparent threat to the smooth functioning of a global information infrastructure and aim to answer the question of how the risks of Y2K were understood, how they were predicted to unfold, and how they were ultimately exaggerated.

One lasting effect of the Y2K bug is its changes to liability law, as it related to computer technology and failure. In the months leading up to the year 2000, insurance companies and technology companies were shielded by policy makers who passed legislation to protect these companies from liability in the event of widespread computer failures from the Y2K bug (Barr 1999; Clausing 1999). The changes to insurance liability heralded the growing political power of Silicon Valley technology companies and this section details how the threat of Y2K was taken up by the U.S. Congress and how insurance liability became a major issue in the American election of 2000. Y2K insurance liability became a political instrument and created a precedent for exceptionalizing corporate responsibility in the case of massive technological failure.

References

Barr, S. (1999, February 3). Industry's 'fix' for Y2K liability: Legislative proposal would limit lawsuits and damages. *The Washington Post,* p. A15.

Ceruzzi, P. E. (2003). *A history of modern computing*. London; Cambridge, MA: MIT Press.

Clausing, J. (1999, February 24). Legislation limiting year 2000 liability is introduced. *New York Times,* p. 2.

Dunn, A. (1998, September 08). Year 2000 Bug likely to infest courtrooms. *Los Angeles Times,* p. A1.

Edwards, P. N. (1998). Y2K: Millennial reflections on computers as infrastructure. *History and Technology, 15*(1-2), 7-29.

Ensmenger, N. (2010). *The computer boys take over: computers, programmers, and the politics of technical expertise*. Cambridge, MA: MIT Press.

Fisher, G. H. (1999). Embedded systems: the forgotten Y2K problem. *INCOSE International Symposium, 9*(1), 119–125.

Gordon, P. D. (1999). A call to action: The national and global implications of the year 2000 embedded systems crisis. *Logistics Information Management, 12*(3), 239–245.

Graham, S., & Thrift, N. (2007). Out of order. *Theory, Culture & Society, 24*(3), 1–25.

Henke, C. R. (2000). The mechanics of workplace order: Toward a sociology of repair. *Berkeley Journal of Sociology, 44*, 55–81.

Jackson, S. (2014). Rethinking repair. In T. Gillespie, P. Boczkowski, & K. Foot (Eds.), *Media technologies: Essays on communication, materiality, and society* (pp. 221–239). Cambridge, MA: MIT Press.

Jackson, S. J., Pompe, A., & Krieshok, G. (2012). *Repair worlds: Maintenance, repair, and ICT for development in rural Namibia*. Paper presented at the Proceedings of the ACM 2012 conference on Computer Supported Cooperative Work.

Jinnet, J., Greene, L. J. *Year 2000 law deskbook*. Miamisburg, OH: LEXIS Publishing.

Kelly, G. (Feb 16, 2016). Apple confirms ridiculous but serious iOS problem. *Forbes*. Retrieved from http://www.forbes.com/sites/gordonkelly/2016/02/16/apple-confirms-ios8-ios9-problem/#7fe4c5bbdd44

Kerr, C. L. (1998). *Understanding, preventing, and litigating year 2000 issues: What every lawyer needs to know now*. New York: Practising Law Institute.

Menand, L. (2015). Thinking sideways. *The New Yorker, 91*(6), 73–75.

McCracken, D. (1962). The software turmoil. *Datamation, 8*(1), 21–22.

Paula, D. G. (1999). A call to action: The national and global implications of the year 2000 embedded systems crisis. *Logistics Information Management, 12*(3), 239–245.

Williams, R. D., & Smyth, B. T. (1999). *Law of the year 2000 problem*. Gaithersburg: Aspen Law and Business.