Software Reliability: Failures, Errors and Risks

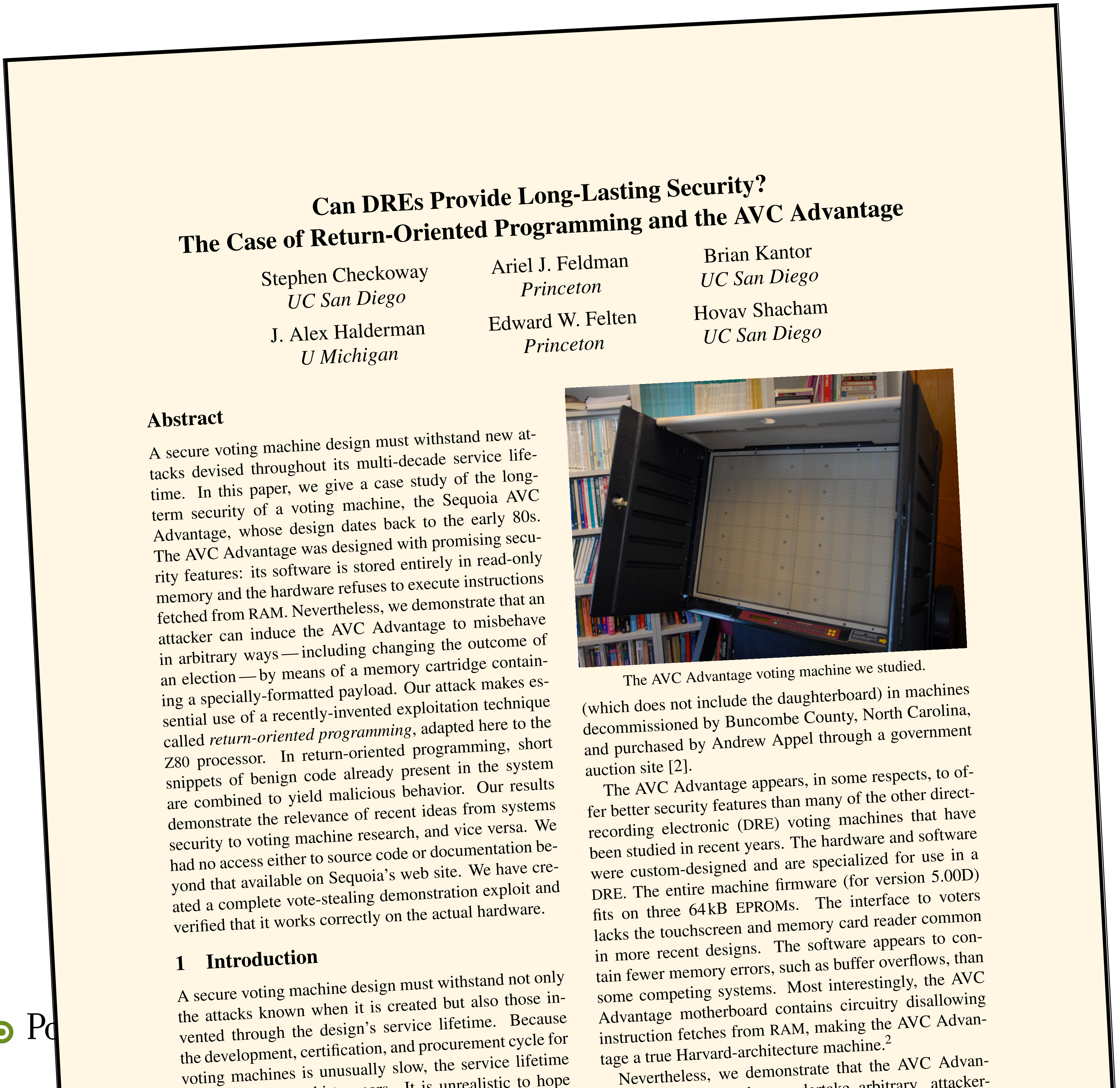
CS 305 — Ethics

Portland State UNIVERSITY

Ethical Issue

- Computer systems are so complex, we can't hope to make them perfect.
- How can we distinguish between:
  - errors in a system that are acceptable as trade-offs for its benefits, and
  - errors that are due to inexcusable carelessness, incompetence, or dishonesty?

Reliability of Voting Machines



Can DREs Provide Long-Lasting Security?
The Case of Return-Oriented Programming and the AVC Advantage

Stephen Checkoway
UC San Diego

Ariel J. Feldman
Princeton

Brian Kantor
UC San Diego

J. Alex Halderman
U Michigan

Edward W. Felten
Princeton

Hovav Shacham
UC San Diego

Abstract

A secure voting machine design must withstand new attacks devised throughout its multi-decade service lifetime. In this paper, we give a case study of the long-term security of a voting machine, the Sequoia AVC Advantage, whose design dates back to the early 80s. The AVC Advantage was designed with promising security features: its software is stored entirely in read-only memory and the hardware refuses to execute instructions fetched from RAM. Nevertheless, we demonstrate that an attacker can induce the AVC Advantage to misbehave in arbitrary ways — including changing the outcome of an election — by means of a memory cartridge containing a specially-formatted payload. Our attack makes essential use of a recently-invented exploitation technique called *return-oriented programming*, adapted here to the Z80 processor. In return-oriented programming, short snippets of benign code already present in the system are combined to yield malicious behavior. Our results demonstrate the relevance of recent ideas from systems security to voting machine research, and vice versa. We had no access either to source code or documentation beyond that available on Sequoia's web site. We have created a complete vote-stealing demonstration exploit and verified that it works correctly on the actual hardware.

1 Introduction

A secure voting machine design must withstand not only the attacks known when it is created but also those invented through the design's service lifetime. Because the development, certification, and procurement cycle for voting machines is unusually slow, the service lifetime

The AVC Advantage voting machine we studied.

(which does not include the daughterboard) in machines decommissioned by Buncombe County, North Carolina, and purchased by Andrew Appel through a government auction site [2].

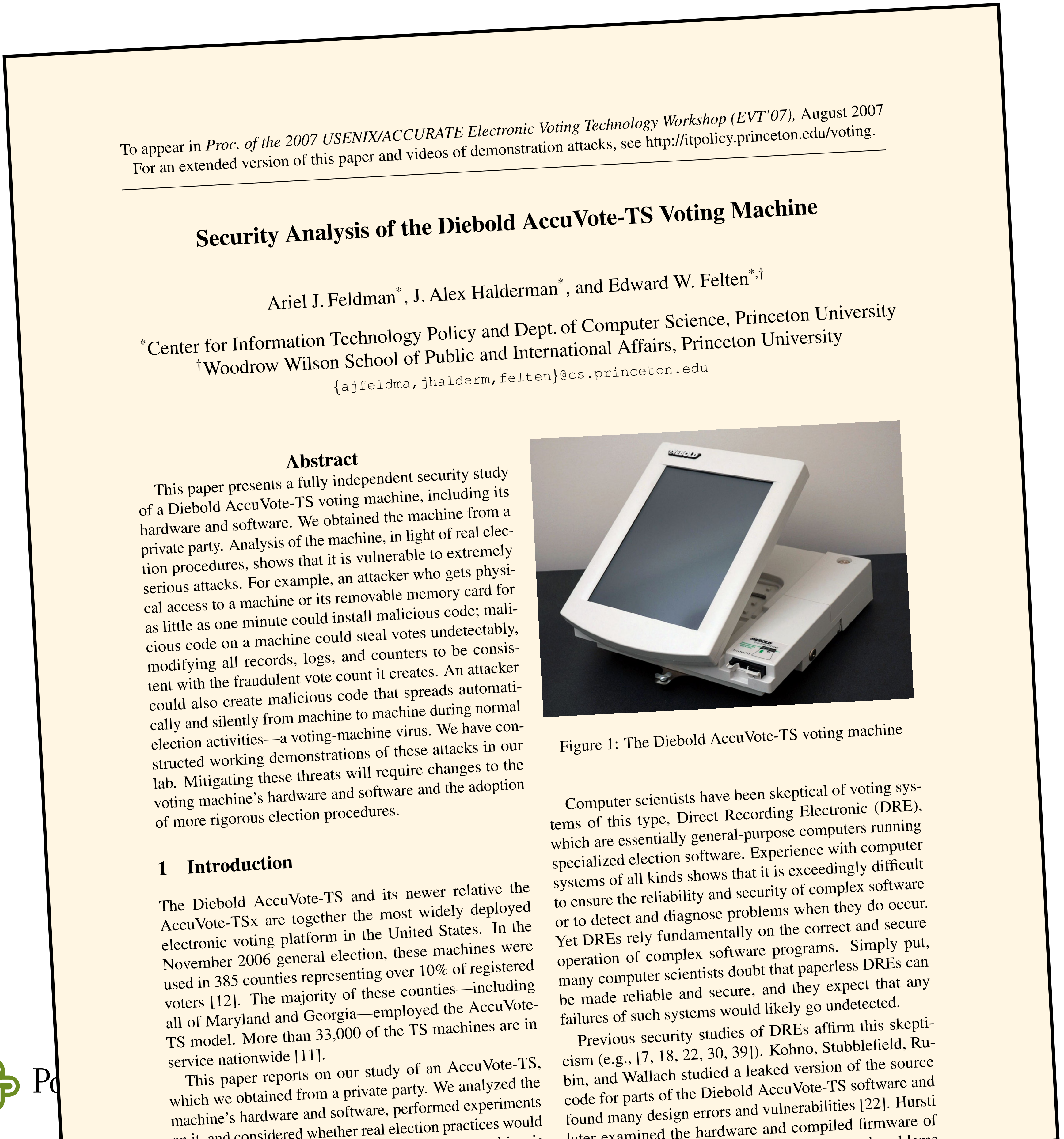
The AVC Advantage appears, in some respects, to offer better security features than many of the other direct-recording electronic (DRE) voting machines that have been studied in recent years. The hardware and software were custom-designed and are specialized for use in a DRE. The entire machine firmware (for version 5.00D) fits on three 64 kB EPROMs. The interface to voters lacks the touchscreen and memory card reader common in more recent designs. The software appears to contain fewer memory errors, such as buffer overflows, than some competing systems. Most interestingly, the AVC Advantage motherboard contains circuitry disallowing instruction fetches from RAM, making the AVC Advantage a true Harvard-architecture machine.[2]

Nevertheless, we demonstrate that the AVC Advan-

Proceedings of EVT/WOTE 2009. USENIX/ACCURATE/IAVoSS, August 2009.

YouTube version:

http://www.youtube.com/watch?v=lsfG3KPrD1I



To appear in *Proc. of the 2007 USENIX/ACCURATE Electronic Voting Technology Workshop (EVT'07)*, August 2007
For an extended version of this paper and videos of demonstration attacks, see http://itpolicy.princeton.edu/voting.

Security Analysis of the Diebold AccuVote-TS Voting Machine

Ariel J. Feldman[*], J. Alex Halderman[*], and Edward W. Felten[*,†]

[*]Center for Information Technology Policy and Dept. of Computer Science, Princeton University
[†]Woodrow Wilson School of Public and International Affairs, Princeton University
{ajfeldma, jhalderm, felten}@cs.princeton.edu

Abstract

This paper presents a fully independent security study of a Diebold AccuVote-TS voting machine, including its hardware and software. We obtained the machine from a private party. Analysis of the machine, in light of real election procedures, shows that it is vulnerable to extremely serious attacks. For example, an attacker who gets physical access to a machine or its removable memory card for as little as one minute could install malicious code; malicious code on a machine could steal votes undetectably, modifying all records, logs, and counters to be consistent with the fraudulent vote count it creates. An attacker could also create malicious code that spreads automatically and silently from machine to machine during normal election activities—a voting-machine virus. We have constructed working demonstrations of these attacks in our lab. Mitigating these threats will require changes to the voting machine's hardware and software and the adoption of more rigorous election procedures.

1 Introduction

The Diebold AccuVote-TS and its newer relative the AccuVote-TSx are together the most widely deployed electronic voting platform in the United States. In the November 2006 general election, these machines were used in 385 counties representing over 10% of registered voters [12]. The majority of these counties—including all of Maryland and Georgia—employed the AccuVote-TS model. More than 33,000 of the TS machines are in service nationwide [11].

This paper reports on our study of an AccuVote-TS, which we obtained from a private party. We analyzed the machine's hardware and software, performed experiments on it, and considered whether real election practices would

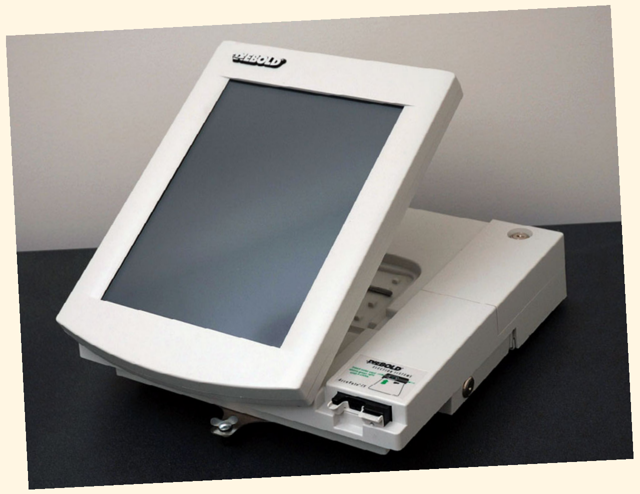


Figure 1: The Diebold AccuVote-TS voting machine

Computer scientists have been skeptical of voting systems of this type, Direct Recording Electronic (DRE), which are essentially general-purpose computers running specialized election software. Experience with computer systems of all kinds shows that it is exceedingly difficult to ensure the reliability and security of complex software or to detect and diagnose problems when they do occur. Yet DREs rely fundamentally on the correct and secure operation of complex software programs. Simply put, many computer scientists doubt that paperless DREs can be made reliable and secure, and they expect that any failures of such systems would likely go undetected.

Previous security studies of DREs affirm this skepticism (e.g., [7, 18, 22, 30, 39]). Kohno, Stubblefield, Rubin, and Wallach studied a leaked version of the source code for parts of the Diebold AccuVote-TS software and found many design errors and vulnerabilities [22]. Hursti later examined the hardware and compiled firmware of

Feldman, A. J., Halderman, J. A., and Felten, E. W. 2006. Security analysis of the Diebold AccuVote-TS voting machine. In Proc. 2007 USENIX/ACCURATE Electronic Voting Technology Workshop (EVT '07).

YouTube version:

http://www.youtube.com/watch?v=aZws98jw67g

Bug-free Software

- Should software manufacturers be able to disclaim responsibility for damages caused by defective software?
  - Mortenson *v*. Timberline Software
- What about systems containing embedded software (cars, medical devices, æroplanes)?
- What about pure hardware systems?

Mortenson Company, Inc., v. Timberline Software Corporation

May 2000: Supreme Court of the state of Washington upheld a lower court ruling that validated a shrinkwrap software license.

- In this case, Mortenson (a contractor) purchased bid-making software from Timberline that was governed by a shrinkwrap license agreement. The license agreement contained the following clause that purported to limit consequential damages.

LIMITATION OF REMEDIES AND LIABILITY

NEITHER TIMBERLINE NOR ANYONE ELSE WHO HAS BEEN INVOLVED IN THE CREATION, PRODUCTION OR DELIVERY OF THE PROGRAMS OR USER MANUALS SHALL BE LIABLE TO YOU FOR ANY DAMAGES OF ANY TYPE, INCLUDING BUT NOT LIMITED TO, ANY LOST PROFITS, LOST SAVINGS, LOSS OF ANTICIPATED BENEFITS, OR OTHER INCIDENTAL, OR CONSEQUENTIAL DAMAGES ARISING OUT OF THE USE OR INABILITY TO USE SUCH PROGRAMS, WHETHER ARISING OUT OF CONTRACT, NEGLIGENCE, STRICT TORT, OR UNDER ANY WARRANTY, OR OTHERWISE, EVEN IF TIMBERLINE HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES OR FOR ANY OTHER CLAIM BY ANY OTHER PARTY. TIMBERLINE'S LIABILITY FOR DAMAGES IN NO EVENT SHALL EXCEED THE LICENSE FEE PAID FOR THE RIGHT TO USE THE PROGRAMS.

Mortenson used Timberline's software to prepare a construction bid. A defect in the program produced an erroneous bid price that was off by $1.95 million. Mortenson sued Timberline for breach of implied and express warranties. Timberline contended that the license agreement clause limiting consequential damages would prevent Mortenson from recouping damages.

Washington State's Supreme Court affirmed a lower court ruling which found that

- “the license terms were part of the contract,” and
- “the limitation of remedies clause was not unconscionable and, therefore, enforceable.”

In a dissenting opinion, Judge Sanders stated:

“Although the majority states ‘this is a case about contract formation, not contract alteration,’ Majority at 17, the majority abandons traditional contract principles governing offer and acceptance and relies on distinguishable cases with blind deference to software manufacturers’ preferred method of conducting business. Instead of creating a new standard of contract formation—the majority's nebulous theory of ‘layered contracting’—I would look to the accepted principles of the Uniform Commercial Code (U.C.C.) and the common law to determine whether Timberline's licensing agreement is enforceable against Mortenson. Because the parties entered a binding and enforceable contract prior to the delivery of the software, I would treat Timberline's license agreement as a proposal to modify the contract requiring either express assent or conduct manifesting assent to those terms.”

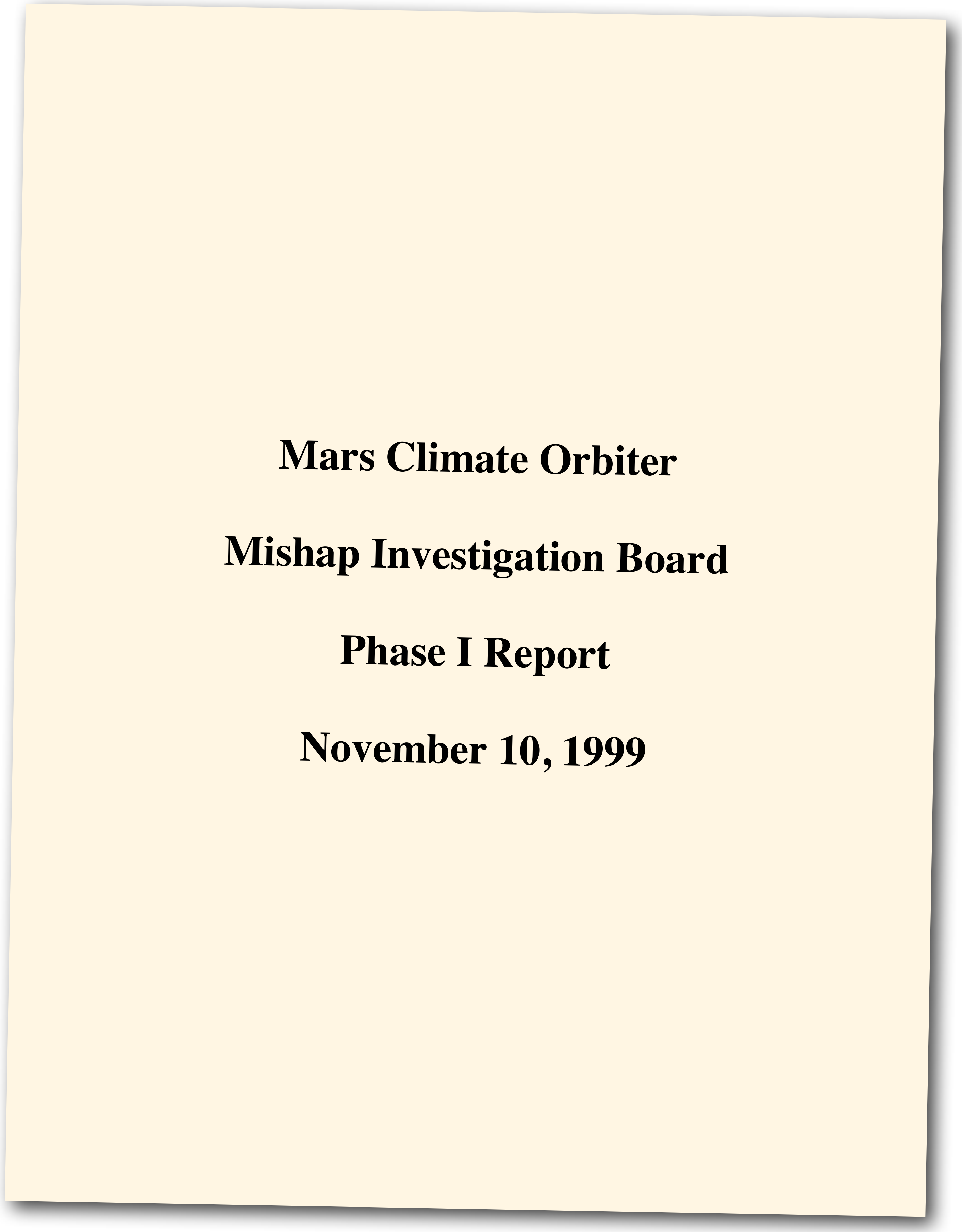
Responsibilities of software makers

- Stand behind their product?
- Notify customers of *known* errors?
- Best practice?
- Best effort?
- No liability?
- Charging for bug fixes?

Mars Climate Orbiter

- Loss of orbiter due to mix-up between British and Metric units.
- *Not* because of poor specification:
  - *“a poorly specified interface allowed this error to remain undetected…”*

Quinn, p392

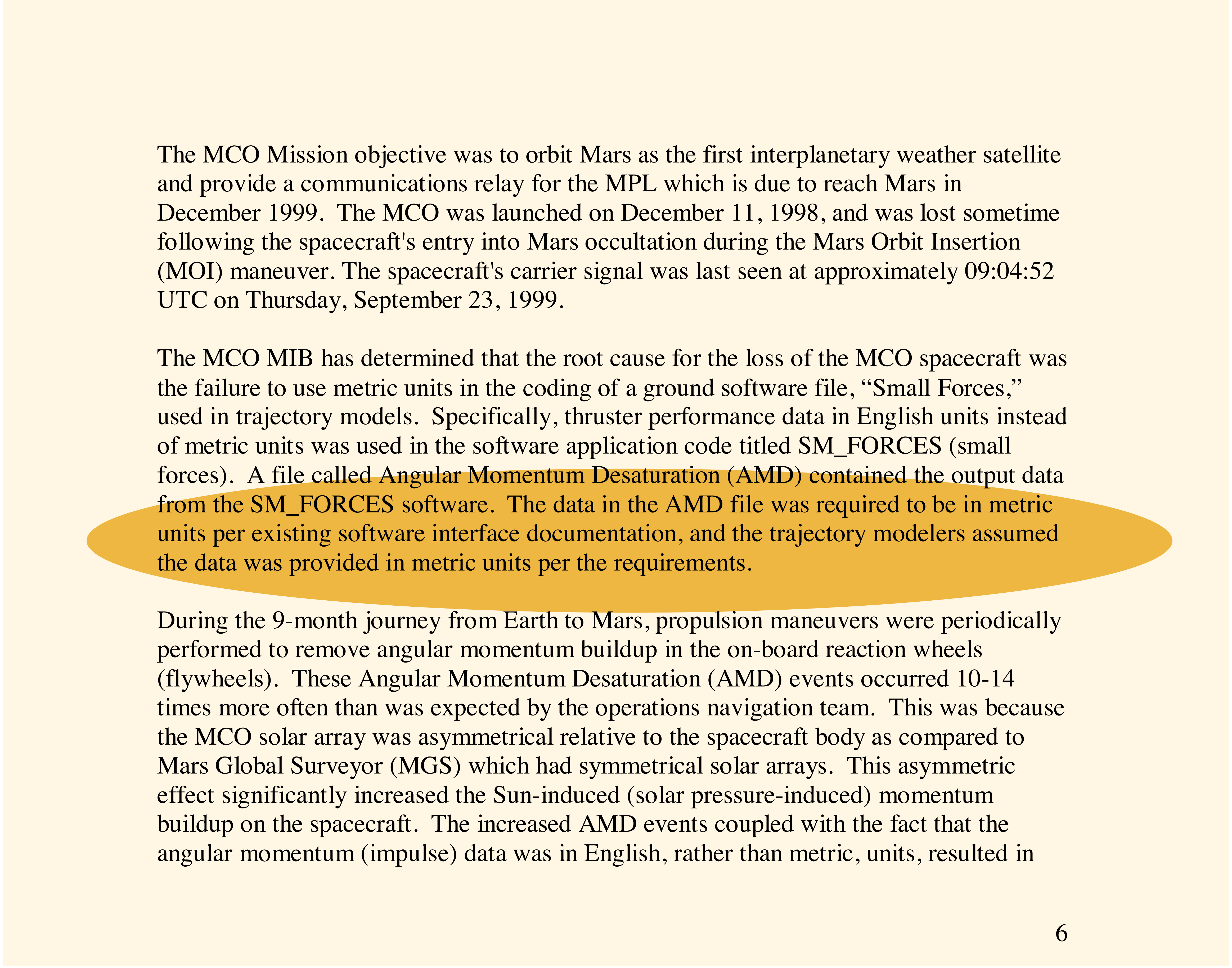


Mars Climate Orbiter

Mishap Investigation Board

Phase I Report

November 10, 1999



The MCO Mission objective was to orbit Mars as the first interplanetary weather satellite and provide a communications relay for the MPL which is due to reach Mars in December 1999. The MCO was launched on December 11, 1998, and was lost sometime following the spacecraft's entry into Mars occultation during the Mars Orbit Insertion (MOI) maneuver. The spacecraft's carrier signal was last seen at approximately 09:04:52 UTC on Thursday, September 23, 1999.

The MCO MIB has determined that the root cause for the loss of the MCO spacecraft was the failure to use metric units in the coding of a ground software file, “Small Forces,” used in trajectory models. Specifically, thruster performance data in English units instead of metric units was used in the software application code titled SM_FORCES (small forces). A file called Angular Momentum Desaturation (AMD) contained the output data from the SM_FORCES software. The data in the AMD file was required to be in metric units per existing software interface documentation, and the trajectory modelers assumed the data was provided in metric units per the requirements.

During the 9-month journey from Earth to Mars, propulsion maneuvers were periodically performed to remove angular momentum buildup in the on-board reaction wheels (flywheels). These Angular Momentum Desaturation (AMD) events occurred 10-14 times more often than was expected by the operations navigation team. This was because the MCO solar array was asymmetrical relative to the spacecraft body as compared to Mars Global Surveyor (MGS) which had symmetrical solar arrays. This asymmetric effect significantly increased the Sun-induced (solar pressure-induced) momentum buildup on the spacecraft. The increased AMD events coupled with the fact that the angular momentum (impulse) data was in English, rather than metric, units, resulted in

6

Root Cause: Failure to use metric units in the coding of a ground software file, "Small Forces," used in trajectory models

Contributing Causes:

1. Undetected mismodeling of spacecraft velocity changes
2. Navigation Team unfamiliar with spacecraft
3. Trajectory correction maneuver number 5 not performed
4. System engineering process did not adequately address transition from development to operations
5. Inadequate communications between project elements
6. Inadequate operations Navigation Team staffing
7. Inadequate training
8. Verification and validation process did not adequately address ground software

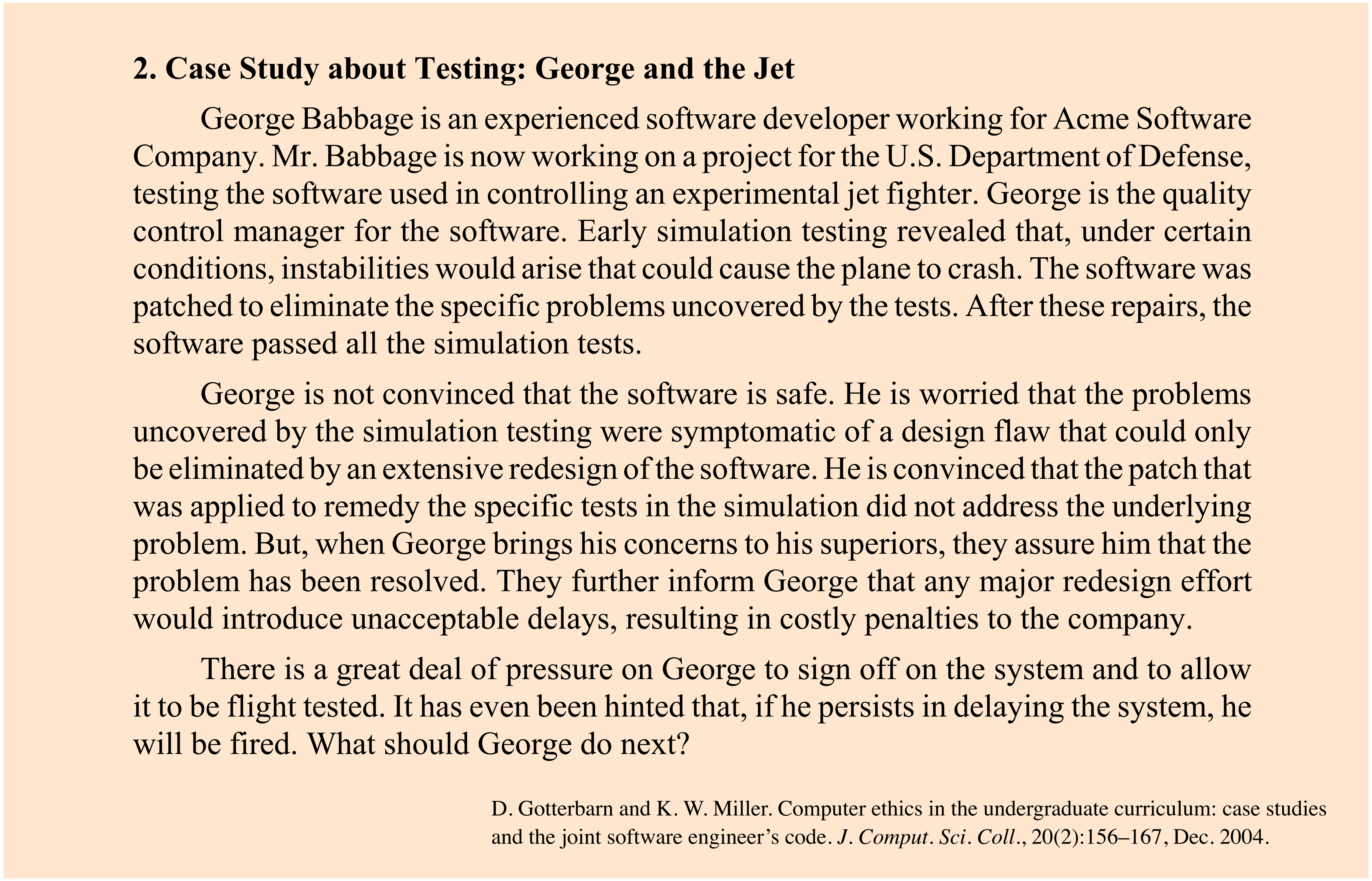
the thruster manufacturer. The calculation of the thruster performance is carried out both on-board the spacecraft and on ground support system computers. Mismodeling only occurred in the ground software.

The Software Interface Specification (SIS), used to define the format of the AMD file, specifies the units associated with the impulse bit to be Newton-seconds (N-s). Newton-seconds are the proper units for impulse (Force x Time) for metric units. The AMD software installed on the spacecraft used metric units for the computation and was correct. In the case of the ground software, the impulse bit reported to the AMD file was in English units of pounds (force)-seconds (lbf-s) rather than the metric units specified. Subsequent processing of the impulse bit values from the AMD file by the navigation software underestimated the effect of the thruster firings on the spacecraft trajectory by a factor of 4.45 (1 pound force=4.45 Newtons).

During the first four months of the MCO cruise flight, the ground software AMD files were not used in the orbit determination process because of multiple file format errors and incorrect quaternion (spacecraft attitude data) specifications. Instead, the operations navigation team used email from the contractor to notify them when an AMD desaturation event was occurring, and they attempted to model trajectory perturbations on

17

Ethical Responsibilities of Software Engineers



2. Case Study about Testing: George and the Jet

George Babbage is an experienced software developer working for Acme Software Company. Mr. Babbage is now working on a project for the U.S. Department of Defense, testing the software used in controlling an experimental jet fighter. George is the quality control manager for the software. Early simulation testing revealed that, under certain conditions, instabilities would arise that could cause the plane to crash. The software was patched to eliminate the specific problems uncovered by the tests. After these repairs, the software passed all the simulation tests.

George is not convinced that the software is safe. He is worried that the problems uncovered by the simulation testing were symptomatic of a design flaw that could only be eliminated by an extensive redesign of the software. He is convinced that the patch that was applied to remedy the specific tests in the simulation did not address the underlying problem. But, when George brings his concerns to his superiors, they assure him that the problem has been resolved. They further inform George that any major redesign effort would introduce unacceptable delays, resulting in costly penalties to the company.

There is a great deal of pressure on George to sign off on the system and to allow it to be flight tested. It has even been hinted that, if he persists in delaying the system, he will be fired. What should George do next?

D. Gotterbarn and K. W. Miller. Computer ethics in the undergraduate curriculum: case studies and the joint software engineer’s code. *J. Comput. Sci. Coll.*, 20(2):156–167, Dec. 2004.

Therac-25 Incidents

- Marietta, Georgia, June 1985
  - breast-cancer patient burned on collarbone
  - Oncology center personnel contacted AECL
  - Patient suffered crippling injuries, sues AECL and center
- Hamilton, Ontario, July 1985
  - Patient burned. Died of cancer Nov 1985
  - AECL investigated, unable to reproduce malfunction
- Yakima, Washington, Dec 1985
  - Radiation burns in parallel stripe pattern
  - AECL claimed that Therac-25 could not have administered an overdose, and that *no similar incidents had been reported*.

Therac-25 Incidents

- Tyler, Texas: March 1986
  - Male patient getting 9th in a series of treatments
    - Video camera and intercom not operating
  - Operator corrects “X” to “E”
  - patient receives massive overdose and dies
  - Hospital shuts down Therac-25
  - AECL engineers said that it was impossible for overdose to be caused by the Therac-25

Therac-25 Incidents

- Tyler, Texas: April 1986
  - A different male patient
  - Operator again corrects “X” to “E”
  - patient receives massive overdose and dies

Portland State UNIVERSITY

Who’s at fault?

- Radiation Technician?
- Hospital Director?
- Programmers who wrote the code?

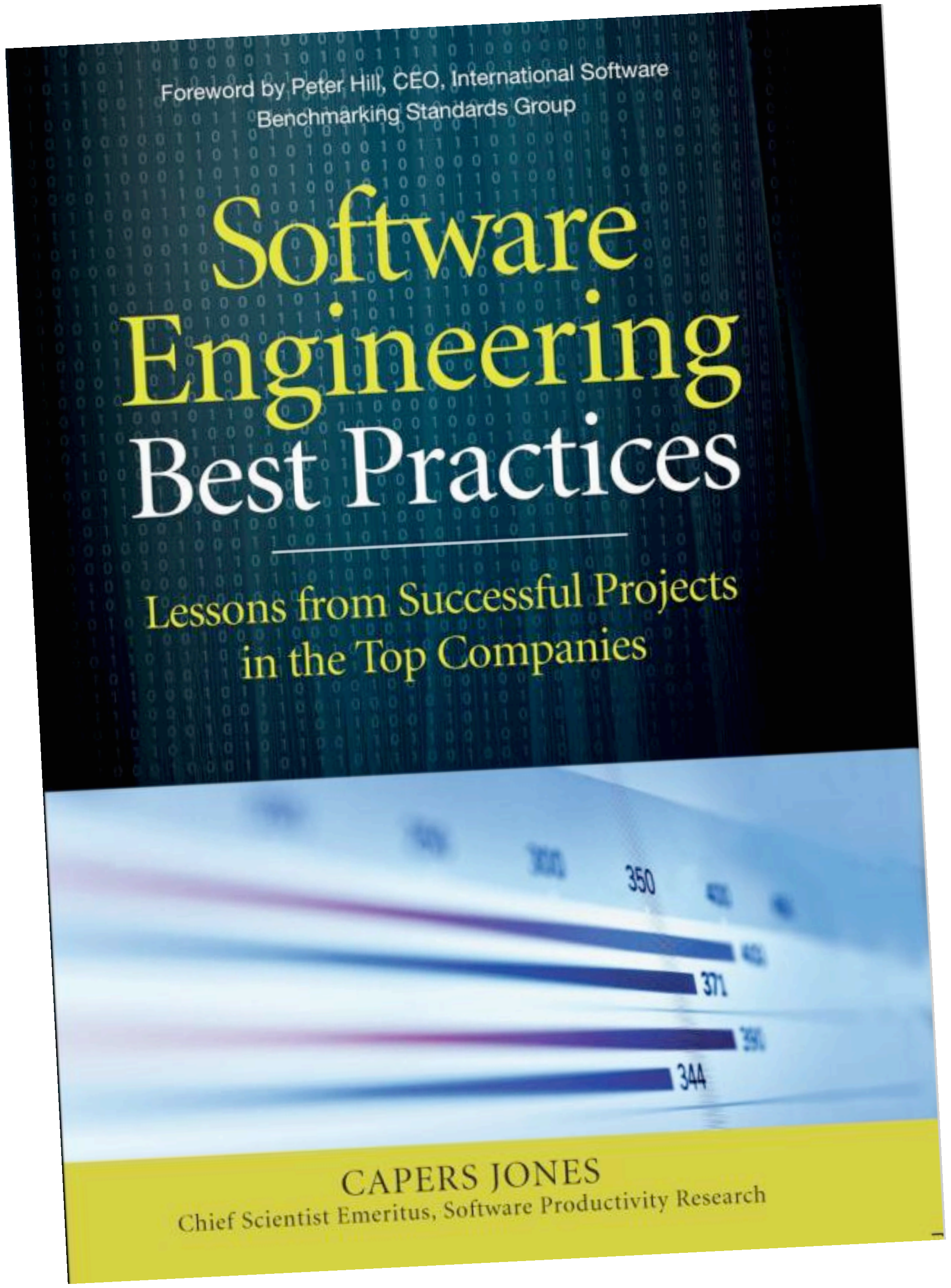
Who's at fault?

	at fault?	not at fault?
Radiation Technician	1	2
Hospital Director	3	4
Programmers	5	6

What should have been done differently?

1. Design was not *fail-safe*
    - no single point of failure can lead to catastrophe
    - economized by omitting hardware interlocks that had been present in previous generations
2. No subsystem for overdose detection
3. Reusing code does not necessarily make a system safer
4. Communicate!
    - with your customers, engineers, operators, …

Best Practices



Foreword by Peter Hill, CEO, International Software Benchmarking Standards Group

Software Engineering Best Practices

Lessons from Successful Projects in the Top Companies

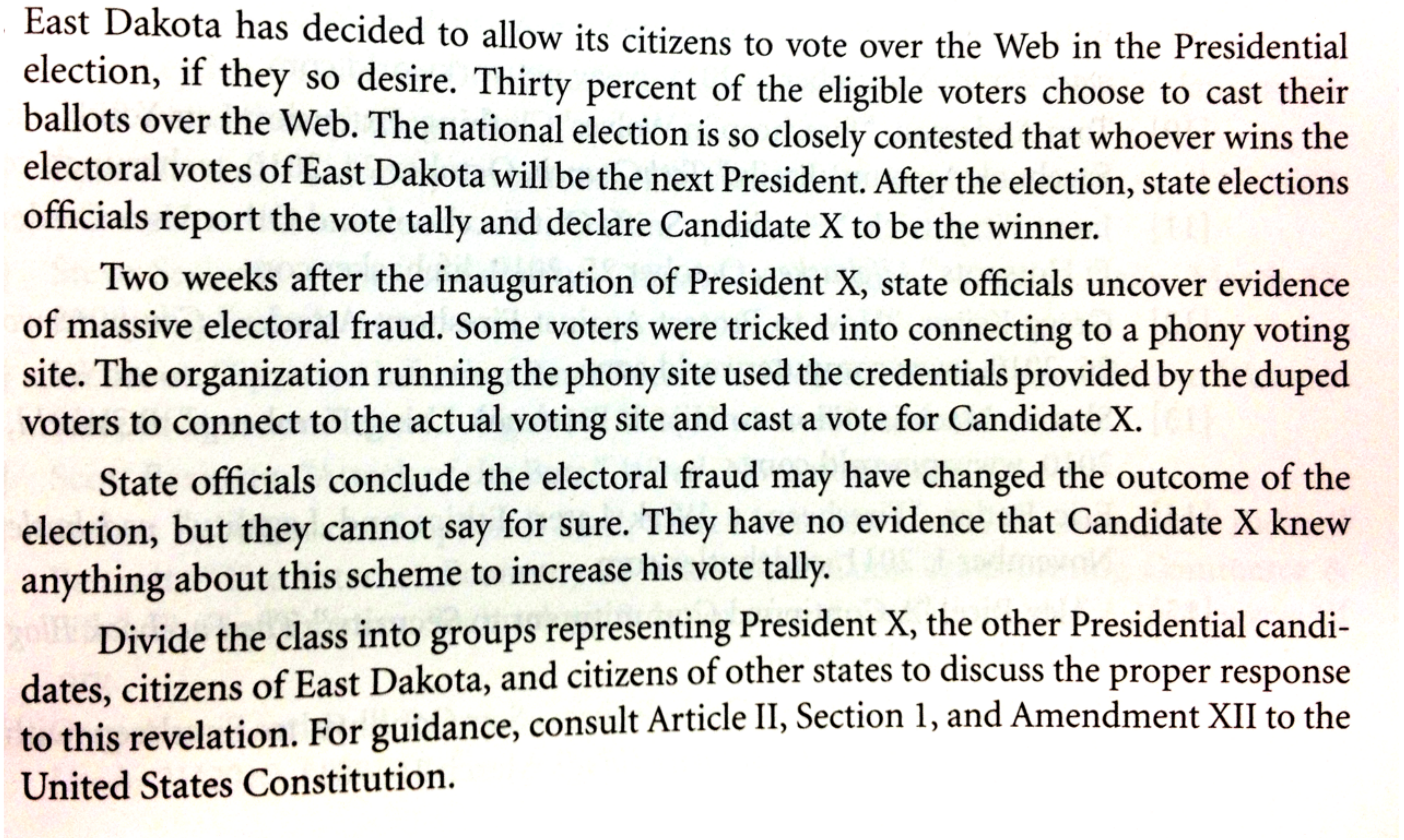
CAPERS JONES

Chief Scientist Emeritus, Software Productivity Research

- Best Practices help, but are not a panacea
  - ‘Best’ depends on context
  - “Depending on the context, sometimes a practice is ‘best’ and sometimes it’s not”

    Scott Ambler

Portland State UNIVERSITY

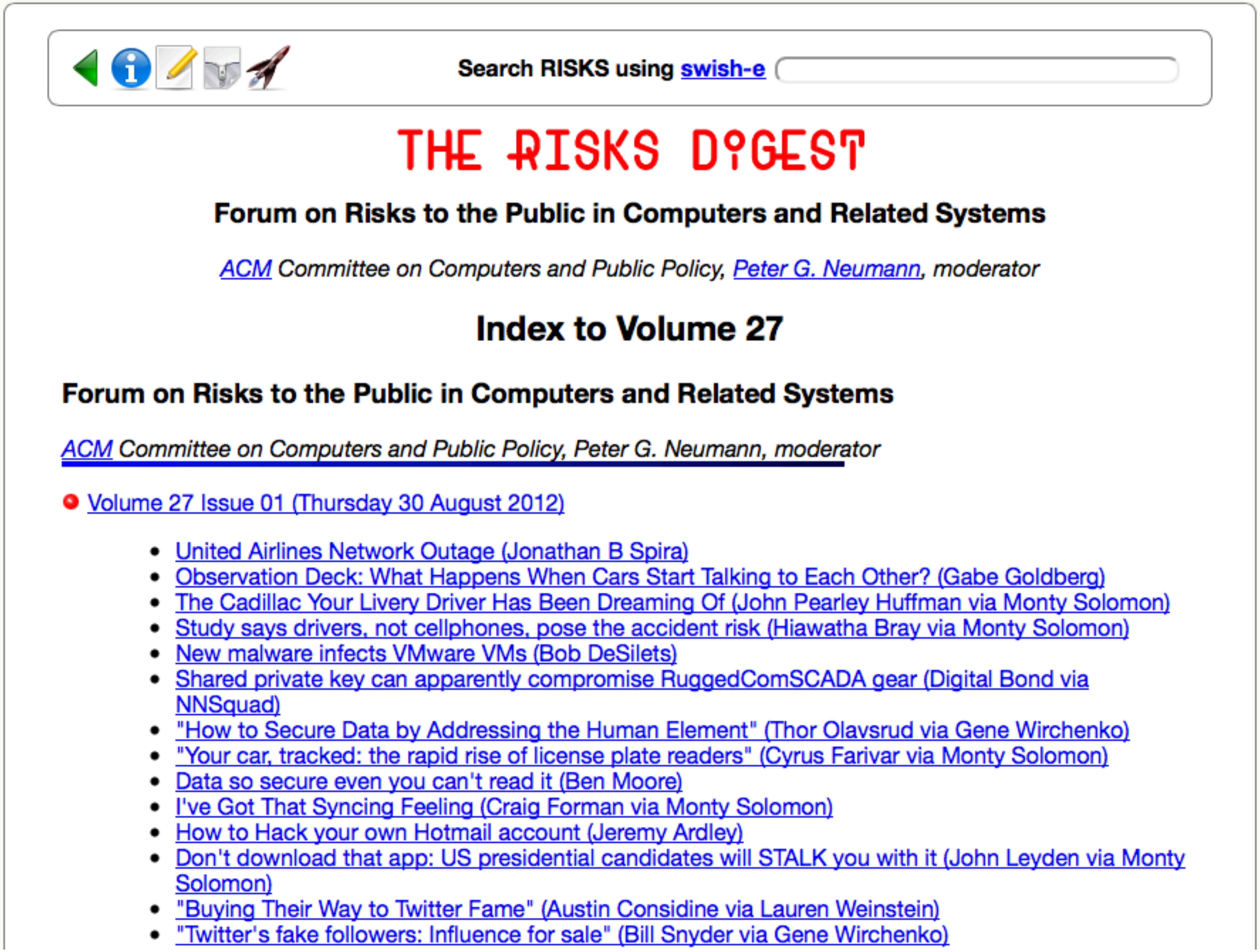


East Dakota has decided to allow its citizens to vote over the Web in the Presidential election, if they so desire. Thirty percent of the eligible voters choose to cast their ballots over the Web. The national election is so closely contested that whoever wins the electoral votes of East Dakota will be the next President. After the election, state elections officials report the vote tally and declare Candidate X to be the winner.

Two weeks after the inauguration of President X, state officials uncover evidence of massive electoral fraud. Some voters were tricked into connecting to a phony voting site. The organization running the phony site used the credentials provided by the duped voters to connect to the actual voting site and cast a vote for Candidate X.

State officials conclude the electoral fraud may have changed the outcome of the election, but they cannot say for sure. They have no evidence that Candidate X knew anything about this scheme to increase his vote tally.

Divide the class into groups representing President X, the other Presidential candidates, citizens of East Dakota, and citizens of other states to discuss the proper response to this revelation. For guidance, consult Article II, Section 1, and Amendment XII to the United States Constitution.


Search RISKS using swish-e

THE RISKS DIGEST

Forum on Risks to the Public in Computers and Related Systems

ACM Committee on Computers and Public Policy, Peter G. Neumann, moderator

Index to Volume 27

Forum on Risks to the Public in Computers and Related Systems

ACM Committee on Computers and Public Policy, Peter G. Neumann, moderator

● Volume 27 Issue 01 (Thursday 30 August 2012)

- United Airlines Network Outage (Jonathan B Spira)
- Observation Deck: What Happens When Cars Start Talking to Each Other? (Gabe Goldberg)
- The Cadillac Your Livery Driver Has Been Dreaming Of (John Pearley Huffman via Monty Solomon)
- Study says drivers, not cellphones, pose the accident risk (Hiawatha Bray via Monty Solomon)
- New malware infects VMware VMs (Bob DeSilets)
- Shared private key can apparently compromise RuggedComSCADA gear (Digital Bond via NNSquad)
- "How to Secure Data by Addressing the Human Element" (Thor Olavsrud via Gene Wirchenko)
- "Your car, tracked: the rapid rise of license plate readers" (Cyrus Farivar via Monty Solomon)
- Data so secure even you can't read it (Ben Moore)
- I've Got That Syncing Feeling (Craig Forman via Monty Solomon)
- How to Hack your own Hotmail account (Jeremy Ardley)
- Don't download that app: US presidential candidates will STALK you with it (John Leyden via Monty Solomon)
- "Buying Their Way to Twitter Fame" (Austin Considine via Lauren Weinstein)
- "Twitter's fake followers: Influence for sale" (Bill Snyder via Gene Wirchenko)

Portland State UNIVERSITY

Risks Digest

http://catless.ncl.ac.uk/Risks/index.27.html

Volume 27 Issue 83 (Friday 11 April 2014)

- For once. a good-news story about social media (Mark Brader)
- Problems with Big Data (Gary Marcus and Ernest Davis)
- Clapper Acknowledges Backdoor Searches (Ellen Nakashima)
- "Beware: The cloud's Ponzi schemes are here" (David Linthicum via Gene Wirchenko)
- OpenSSL Heartbleed vulnerability (Alex Hern)
- TA14-098A: OpenSSL 'Heartbleed' vulnerability (US-CERT)
- Experts Find a Door Ajar in an Internet Security Method (Nicole Perlroth)
- "The Heartbleed OpenSSL flaw is worse than you think" (Roger A. Grimes via Gene Wirchenko)
- NSA monitors Wi-Fi on US planes 'in violation' of privacy laws (RT USA via Dewayne Hendricks)
- Yahoo breaks every mailing list in the world including the IETF's (John Levine via NNSquad)
- Technology's Man Problem (Claire Cain Miller via Lauren Weinstein)
- Details of how Turkey is intercepting Google Public DNS (Bortzmeyer via NNSquad)