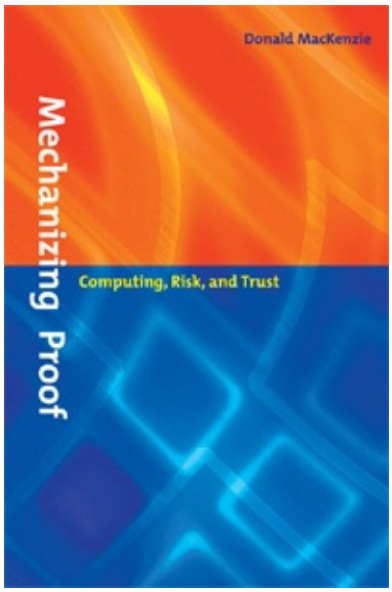
Cover



title:	Mechanizing Proof : Computing, Risk, and Trust Inside Technology
author:	MacKenzie, Donald A.
publisher:	MIT Press
isbn10 | asin:	0262133938
print isbn13:	9780262133937
ebook isbn13:	9780585436739
language:	English
subject	Computer systems--Reliability, Computers and civilization, Systáemes informatiques--Fiabilitâe, Ordinateurs et civilisation.
publication date:	2001
lcc:	QA76.76.R44M36 2001eb
ddc:	004/.2/1
subject:	Computer systems--Reliability, Computers and civilization, Systáemes informatiques--Fiabilitâe, Ordinateurs et civilisation.

Inside Technology

edited by Wiebe E. Bijker, W. Bernard Carlson, and Trevor Pinch

Janet Abbate, *Inventing the Internet*

Charles Bazerman, *The Languages of Edison's Light*

Marc Berg, *Rationalizing Medical Work: Decision Support Techniques and Medical Practices*

Wiebe E. Bijker, *Of Bicycles, Bakelites, and Bulbs: Toward a Theory of Sociotechnical Change*

Wiebe E. Bijker and John Law, editors, *Shaping Technology/Building Society: Studies in Sociotechnical Change*

Stuart S. Blume, *Insight and Industry: On the Dynamics of Technological Change in Medicine*

Geoffrey C. Bowker, *Science on the Run: Information Management and Industrial Geophysics at Schlumberger, 1920–1940*

Geoffrey C. Bowker and Susan Leigh Star, *Sorting Things Out: Classification and Its Consequences*

Louis L. Bucciarelli, *Designing Engineers*

H. M. Collins, *Artificial Experts: Social Knowledge and Intelligent Machines*

Paul N. Edwards, *The Closed World: Computers and the Politics of Discourse in Cold War America*

Herbert Gottweis, *Governing Molecules: The Discursive Politics of Genetic Engineering in Europe and the United States*

Gabrielle Hecht, *The Radiance of France: Nuclear Power and National Identity after World War II*

Kathryn Henderson, *On Line and On Paper: Visual Representations, Visual Culture, and Computer Graphics in Design Engineering*

Eda Kranakis, *Constructing a Bridge: An Exploration of Engineering Culture, Design, and Research in Nineteenth-Century France and America*

Pamela E. Mack, *Viewing the Earth: The Social Construction of the Landsat Satellite*

System

Donald MacKenzie, *Inventing Accuracy: A Historical Sociology of Nuclear Missile Guidance*

Donald MacKenzie, *Knowing Machines: Essays on Technical Change*

Donald MacKenzie, *Mechanizing Proof: Computing, Risk, and Trust*

Maggie Mort, *Building the Trident Network: A Study of the Enrolment of People, Knowledge, and Machines*

Susanne K. Schmidt and Raymund Werle, *Coordinating Technology: Studies in the International Standardization of Telecommunications*

Page iii

Mechanizing Proof
Computing, Risk, and Trust

Donald MacKenzie

The MIT Press
Cambridge, Massachusetts
London, England

© 2001 Massachusetts Institute of Technology

All rights reserved. No part of this book may be reproduced in any form or by any electronic or mechanical means (including photocopying, recording, or information storage and retrieval) without permission in writing from the publisher.

This book was set in Sabon by Achorn Graphic Services, Inc. on the Miles 33 system and was printed and bound in the United States of America.

Library of Congress Cataloging-in-Publication Data

MacKenzie, Donald A.
Mechanizing proof : computing, risk, and trust / Donald A. MacKenzie.
p. cm. — (Inside technology)
Includes bibliographical references and index.
ISBN 0-262-13393-8 (HC : alk. paper)
1. Computer systems—Reliability. 2. Computers and civilization. I. Title.
II. Series.

QA76.76.R44 M36 2001
004'. 2' 1—dc21

Contents

Page vii

to Caroline

Page viii

This page intentionally left blank.

Acknowledgments

The work upon which this book is based was made possible by a series of generous research grants: the U.K. Economic and Social Research Council (especially grant R000234031, Studies in the Sociology of Proof, but also grant R00029008, and the Council's Programme on Information and Communication Technologies); the Joint Committee of that Council and the Science and Engineering Research Council (grant GR/H74452); the U.K. Safety Critical Systems Research Programme (GR/J58619); and the Engineering and Physical Sciences Research Council (EPSRC; GR/L37953). The final stages of the writing were supported by EPSRC's grant (GR/N13999) to DIRC, the Interdisciplinary Research Collaboration on the Dependability of Computer-Based Systems.

These grants supported several coworkers without whose help I could not have written the book, especially Eloína Peláez (whose Ph.D. thesis, drawn on heavily in chapter 2, first sparked my interest in the topic), Maggie Tierney, Tony Dale (who conducted the majority of the interviews drawn on here), and Garrel Pottinger (whose interviews form the basis for chapter 5, and who has kindly allowed me in that chapter to draw upon material from an earlier joint article). Further background interviews were conducted by Dave Aspinall, Savitri Maharaj, Claudio Russo, and Colin Smart. This large body of interview material was transcribed by Antoinette Butler, Jean Goldring, and (with exceptional dedication and skill) Dominic Watt. Moyra Forrest gathered literally hundreds of (sometimes obscure) primary sources, and Barbara Silander heroically word-processed several drafts of the text as well as turning my rough sketches into neat figures. Much of this work was done in Edinburgh University's Research Centre for Social Sciences, with the unstinting support of its directors, Frank Bechhofer and, especially, Robin Williams. My colleagues in the Department of Sociology shouldered the burden when the research grants permitted me time free of

teaching and administration, and the Department of the History of Science, Harvard University, provided a friendly home during part of the writing. My warm thanks to all.

The research also would not have been possible without the support and assistance of many members of the communities whose work is studied here. Several of them became fellow investigators in the grants listed above: Stuart Anderson, Alan Bundy, and Bernard Carré played particularly active roles. I am deeply grateful for other forms of help I received from Kenneth Appel, Wiebe Bijker, David Bloor, Bob Boyer, Alan Bundy, Bob Constable, Edsger W. Dijkstra, Moritz Epple, Rudolf Fritsch, Susan Gerhart, Mike Gordon, Wolfgang Haken, John Harrison, Pat Hayes, Tony Hoare, Cliff Jones, Roger Jones, Martin Kusch, Leslie Lamport, Carl Landwehr, Mike Mahoney, Ursula Martin, John McLean, Richard DeMillo, Robin Milner, J Strother Moore, Trevor Pinch, Randy Pollack, Dag Prawitz, Andrew Ranicki, Neil Robertson, Alan Robinson, John Rushby, Don Sannella, Richard Schwartz, Natarajan Shankar, Rob Shostak, Herbert Simon, Keith Stenning, Colin Stirling, Edward Swart, Sherry Turkle, Sam Valentine, Martina Weiss, and Tim Williamson. Many of those on this list read drafts of sections of the book or of chapters of it; a few even read the entire draft. Their comments helped me improve it greatly. Since I did not always take their advice, however, and since many sections were rewritten comprehensively after I received comments, responsibility for the remaining mistakes and errors of interpretation is mine alone.

Having coworkers who did so much of the interviewing meant that I was able to avoid being away from home too often, but writing a book demands concentration that is not always easily compatible with family life. Above all I am grateful to my family, Caroline, Alice, and Iain, who shared with me all the ups and downs of the decade that this research took.

The early sections of chapter 2 draw upon my chapter in *Systems, Experts, and Computers: The Systems Approach in Management and Engineering, World War II and After*, eds. Agatha C. Hughes and Thomas P. Hughes (Cambridge, Mass.: MIT Press, 2000). Chapters 3 and 8 draw upon my article "The Automation of Proof: A Historical and Sociological Exploration," *IEEE Annals of the History of Computing* 17(3) (1995): 7–29; an earlier version of chapter 4 appeared as "Slaying the Kraken: The Sociohistory of a Mathematical Proof," *Social Studies of Science* 29 (1999): 7–60; chapter 5 draws upon sections (drafted originally by me) of my joint article with Garrel Pottinger, "Mathematics, Technology, and

Trust: Formal Verification, Computer Security, and the U.S. Military," *IEEE Annals of the History of Computing* 19(3) (1997): 41–59. I am grateful to the copyright holders for permission to use this material here.

I also thank all those who provided me with or permitted me to reproduce illustrations, in particular the American Mathematical Society, Kenneth Appel, Barry Boehm, Doug Bonin, Randal Bryant, Edsger W. Dijkstra, the Charles Stark Draper Laboratory, the Grand Hotel Sonnenbichl, Wolfgang Haken, Tom Hales, the Institute of Electrical and Electronics Engineers, John Koch, Leonard LaPadula, Daryl McCullough, the National Aeronautics and Space Administration, Peter Naur, the George C. Page Museum, Penguin Books, Roger R. Schell, Scientific American, Inc., Herbert A. Simon, Springer-Verlag, and Willis H. Ware.

Page xii

This page intentionally left blank.

1
Knowing Computers

Of all the technologies bequeathed to us by the twentieth century, one above all saturates our lives and our imaginations: the digital computer. In little more than half a century, the computer has moved from rarity to ubiquity. In the rich, Euro-American, world—and to a growing extent beyond it as well—computers now play an essential part in work, education, travel, communication, leisure, finance, retail, health care, and the preparation and the conduct of war. Increasingly, computing is to be found in devices that do not look like computers as one ordinarily thinks of them: in engines, consumer products, mobile telephones, and in the very fabric of buildings.

The benefits brought by all this computerization are clear and compelling, but it also brings with it dependence. Human safety, the integrity of the financial system, the functioning of utilities and other services, and even national security all depend upon computing. Fittingly, the twentieth century's end was marked both by an upsurge of enthusiasm for computing and by a wave of fear about it: the huge soon-to-be-reversed rise in the prices of the stock of the "dotcom" Internet companies and the widespread worries about the "millennium bug," the Y2K (year 2000) date change problem.

How can we know that the computing upon which we depend is dependable? This is one aspect of a question of some generality: how do we know the properties of artifacts? The academic field of the social studies of science and technology (a diverse set of specialisms that examine the social nature of the content of science and technology and their relations to the wider society) has for several decades been asking how we know the properties of the natural world.1 The corresponding question for the properties of artifacts is much less often asked; surprisingly so, given that a good part of recent sociological interest in technology derives ultimately from the sociology of scientific knowledge.2

Asking the question specifically for computers—how do we know the properties of computers and of the programs that run on them?—is of particular interest because it highlights another issue that sociological work on science has not addressed as much as it might: deductive knowledge.

Sources of knowledge, whether of the properties of the natural world or of those of artifacts, can usefully be classified into three broad categories:

• induction—we learn the properties by observation, experiment, and (especially in the case of artifacts) testing and use;

• authority—people whom we trust tell us what the properties are; and

• deduction—we infer the properties from other beliefs, for example by deducing them from theories or models.3

Social studies of science have focused primarily on the first of these processes, induction, and on its relations to the other two: on the dependence of induction upon communal authority4 and interpersonal trust;5 and on the interweaving of induction and deductive, theoretical knowledge.6

Deduction itself has seldom been the focus of attention: the sociology of mathematics is sparse by comparison with the sociology of the natural sciences; the sociology of formal logic is almost nonexistent.7 At the core of mathematics and formal logic is deductive proof. That propositions in these fields can be proved, not simply justified empirically, is at the heart of their claim to provide "harder," more secure, knowledge than the natural sciences. Yet deductive proof, for all its consequent centrality, has attracted remarkably little detailed attention from the sociology of science, the work of David Bloor and Eric Livingston aside.8 In the social studies of science more widely, the single best treatment of proof remains one that is now forty years old, by the philosopher Imre Lakatos in the 1961 Ph.D. thesis that became the book *Proofs and Refutations* (see chapter 4).9 Indeed, it has often been assumed that there is nothing sociological that can be said about proof, which is ordinarily taken to be an absolute matter. I encountered that presumption more than once at the start of this research. Even the founder of the modern sociology of knowledge, Karl Mannheim, excluded mathematics and logic from the potential scope of its analyses.10

Asking how we know the properties of computer systems (of hardware and/or of software) directly raises the question of deductive knowledge

and of proof. An influential current of thinking within computer science has argued that inductive knowledge of computer systems is inadequate, especially in contexts in which those systems are critical to human safety or security. The number of combinations of possible inputs and internal states of a computer system of any complexity is enormously large. In consequence it will seldom be feasible, even with the most highly automated testing, to exercise each and every state of a system to check for errors and the underlying design faults or "bugs" that may have caused them.11 As computer scientist Edsger W. Dijkstra famously put it in 1969, "Program testing can be used to show the presence of bugs, but never to show their absence!"12 Even extensive computer use can offer no guarantees, because bugs may lurk for years before becoming manifest as a system failure.

Authority on its own is also a problematic source of knowledge of the properties of computer systems. In complex, modern societies trust is typically not just an interpersonal matter, but a structural, occupational one. Certain occupations, for example, are designated "professions," with formal controls over membership (typically dependent upon passing professional examinations), norms requiring consideration of the public good (not merely of self-interest), mechanisms for the disciplining and possible expulsion of incompetent or deviant members, the capacity to take legal action against outsiders who claim professional status, and so on. Although many computer hardware designers are members of the established profession of electrical and electronic engineering, software development is only partially professionalized. Since the late 1960s, there have been calls for "software engineering" (see chapter 2), but by the 1990s it was still illegal, in forty-eight states of the United States, to describe oneself as a "software engineer," because it was not a recognized engineering profession.13

With induction criticized and authority weak, the key alternative or supplementary source of knowledge of the properties of computer systems has therefore often been seen as being deductive proof. Consider the analogy with geometry.14 A mathematician does not seek to demonstrate the correctness of Pythagoras's theorem15 by drawing triangles and measuring them: since there are infinitely many possible right-angled triangles, the task would be endless, just as the exhaustive testing of a computer system is, effectively, infeasible. Instead, the mathematician seeks a proof: an argument that demonstrates that the theorem must hold in all cases. If one could, first, construct a faithful mathematical representation of what a program or hardware design was intended to

do (in the terminology of computing, construct a "formal specification"), and, second, construct a mathematical representation of the actual program or hardware design, then perhaps one could prove deductively that the program or design was a correct implementation of its specification. It would not be a proof that the program or design was in an absolute sense "correct" or "dependable," because the specification might not capture what was required for safety, security, or some other desired property, and because an actual physical computer system might not behave in accordance with even the most detailed mathematical model of it. All of those involved in the field would acknowledge that these are questions beyond the scope of purely deductive reasoning.16 Nevertheless, "formal verification" (as the application of deductive proof to programs or to hardware designs is called) has appeared to many within computer science to be potentially a vital way of determining whether specifications have been implemented correctly, and thus a vital source of knowledge of the properties of computer systems.

The Computer and Proof

Formal verification is proof about computers. Closely related, but distinct, is the use of computers in proof. There are at least three motivations for this use. First, proofs about computer hardware designs or about programs are often highly intricate. Many of those who sought such proofs therefore turned to the computer to help conduct them. Mechanized proofs, they reasoned, would be easier, more dependable, and less subject to human wishful thinking than proofs conducted using the traditional mathematicians' tools of pencil and paper. This would be especially true in an area where proofs might be complicated rather than conceptually deep and in which mistakes could have fatal real-world consequences. Second, some mathematicians also found the computer to be a necessary proof tool in pure mathematics. Chapter 4 examines the most famous instance of such use: the 1976 computer-assisted solution of the long unsolved four-color problem: how to prove that four colors suffice to color in any map drawn upon a plane such that countries that share a border are given different colors. A third source of interest in using computers in proofs is artificial intelligence. What is often claimed to be the first artificial-intelligence program was designed to prove theorems in propositional logic (see chapter 3), and automated theorem proving has remained an important technical topic in artificial intelligence ever since. Underneath the more glamorous

applications of artificial intelligence, such as robotics, are often" inference engines" based upon theorem proving.

Although the theorems they prove would not be regarded by mathematicians as interesting or challenging, more sophisticated mechanized provers (far broader in their scope than the ad hoc programs used in the four-color proof) have been used to solve problems that human mathematicians found intractable, such as the 1996 solution of the Robbins problem, an open question in Boolean algebra (see chapter 3). Automated theorem proving, therefore, raises the question of whether there is a sense in which a computer can be an artificial mathematician. To the proponents of artificial intelligence this has been an attractive possibility; to its opponents it is a repugnant one.

Historical Sociology

The topic of this book is the interrelations of computing, risk, and proof. It is neither a technical nor a philosophical treatment of them, but rather a historical sociology. The historical aspect is the more straightforward. What this book seeks to do is to describe salient features of the interrelations between computing and proof as they have evolved from the 1950s onward. Relevant archival material is still sparse, so I have used two main bodies of evidence. First is the technical literature of computer science and of artificial intelligence, which offers the central documentary record of the evolving thinking of the relevant technical communities. Second is an extensive series of "oral history" interviews with the main participants. Interview data must be treated with caution: interviewees' memories are fallible, and they may wish a particular version of events to be accepted. As far as possible, therefore, I have tried to use written sources to document factual assertions, using interviews primarily as evidence of interviewees' opinions and beliefs.

Even in this role, interviews are an imperfect source (interviewees' opinions may change through time in ways they may not wish to acknowledge or of which they may even be unaware). Nevertheless, they add an important dimension to what would otherwise be the overabstract and disembodied history that would result from the use of the technical literature alone. The interviews were conducted under conditions of confidentiality, but they were tape-recorded and transcribed, and intended quotations were sent to interviewees for review and permission to publish.17 In many cases, points made in interview were then elaborated in further exchanges, especially by electronic mail.

The result, I hope, is accurate narrative history. Nevertheless, it is selective history: I have not attempted comprehensive accounts of the development of the various technical specialisms bearing upon computing, risk, and proof.18 I have focused upon episodes, issues, and debates that appeared to be particularly interesting from the viewpoint of my twin concerns: the nature of knowledge of the properties of computer systems and the nature of deductive proof. I see these not just as technical matters (though plainly they are that) but as sociological questions, issues for the social studies of science and technology.

Since the social studies of science (less so those of technology) have recently become controversial in the debate called the "science wars,"19 some preliminary remarks on the sociological aspect of the approach taken here are also necessary. Although some of the diverse strands that make up social studies of science can rightly be classified as criticism of science (not just in the literary, but in the everyday meaning of "criticism"), that is emphatically not true of the kind of historical sociology pursued here. To investigate the variety of meanings of mathematical proof, for example, is simply to inquire, not to denigrate. The fundamental point—almost always missed in "science wars" debates—is that the analysis of science, technology, and mathematics is not a zero-sum game in which the greater the weight of social influence the less the weight of empirical input or other aspects of what used to be called "internal" factors, such as intellectual consistency and rigor.20 For example, that modern mathematicians do not usually work in isolation, but as members of specialist mathematical communities, deeply affecting each other's work, has surely the typical effect of improving the mathematics they generate. That kind of social influence, which arises from the way in which science and mathematics are conducted within communities, is in no sense in tension with empirical input or with matters such as consistency. In an appropriate community with an appropriate orientation, social processes surely generate rigor rather than undermine it.

Nor should one assume a priori the existence of a zero-sum trade-off when social influence arises from outside the scientific community. In a sense, most of this book is a study of social influence of that second kind, of the effects upon deductive knowledge of the desire to be able reliably to predict the behavior of computer systems upon which human life and security depend. Those effects, I would argue and I think this book demonstrates, have in general been beneficial intellectually as well as practically. Concern for safety or security does not diminish concern for rigor or for intellectual consistency; it increases it.