SOFTWARE AND MIND

Andrei Sorin

EXTRACT

Chapter 8: *From Mechanism to Totalitarianism*
Section *The End of Responsibility*

This extract includes the book's front matter and part of chapter 8.

This section shows how our mechanistic culture fosters a deterministic view of human affairs, which undermines the notion of individual responsibility and promotes totalitarianism.

The entire book, each chapter separately, and also selected sections, can be viewed and downloaded free at the book's website.

www.softwareandmind.com

SOFTWARE AND MIND

The Mechanistic Myth and Its Consequences

Andrei Sorin

ANDSOR BOOKS

Published by Andsor Books, Toronto, Canada (www.andsorbooks.com)
First edition 2013. Revised 2019.

For disclaimers see pp. vii, xvi.

Designed and typeset by the author with text management software developed by the author and with Adobe FrameMaker 6.0. Printed and bound in the United States of America.

Acknowledgements

Excerpts from the works of Karl Popper: reprinted by permission of the University of Klagenfurt/Karl Popper Library.

Excerpts from *The Origins of Totalitarian Democracy* by J. L. Talmon: published by Secker & Warburg, reprinted by permission of The Random House Group Ltd.

Excerpts from *Nineteen Eighty-Four* by George Orwell: Copyright ©1949 George Orwell, reprinted by permission of Bill Hamilton as the Literary Executor of the Estate of the Late Sonia Brownell Orwell and Secker & Warburg Ltd.; Copyright ©1949 Harcourt, Inc. and renewed 1977 by Sonia Brownell Orwell, reprinted by permission of Houghton Mifflin Harcourt Publishing Company.

Excerpts from *The Collected Essays, Journalism and Letters of George Orwell*: Copyright ©1968 Sonia Brownell Orwell, reprinted by permission of Bill Hamilton as the Literary Executor of the Estate of the Late Sonia Brownell Orwell and Secker & Warburg Ltd.; Copyright ©1968 Sonia Brownell Orwell and renewed 1996 by Mark Hamilton, reprinted by permission of Houghton Mifflin Harcourt Publishing Company.

Excerpts from *Doublespeak* by William Lutz: Copyright ©1989 William Lutz, reprinted by permission of the author in care of the Jean V. Naggar Literary Agency.

Excerpts from *Four Essays on Liberty* by Isaiah Berlin: Copyright ©1969 Isaiah Berlin, reprinted by permission of Curtis Brown Group Ltd., London, on behalf of the Estate of Isaiah Berlin.

Library and Archives Canada Cataloguing in Publication

Sorin, Andrei
Software and mind : the mechanistic myth and its consequences / Andrei Sorin.

Includes index.
ISBN 978-0-9869389-0-0

1. Computers and civilization. 2. Computer software – Social aspects.
3. Computer software – Philosophy. I. Title.

QA76.9.C66S67 2013 303.48'34 C2012-906666-4

Don't you see that the whole aim of Newspeak is to narrow the range of thought?… Has it ever occurred to you … that by the year 2050, at the very latest, not a single human being will be alive who could understand such a conversation as we are having now?

George Orwell, *Nineteen Eighty-Four*

Disclaimer

This book attacks the mechanistic myth, not persons. Myths, however, manifest themselves through the acts of persons, so it is impossible to discuss the mechanistic myth without also referring to the persons affected by it. Thus, all references to individuals, groups of individuals, corporations, institutions, or other organizations are intended solely as examples of mechanistic beliefs, ideas, claims, or practices. To repeat, they do not constitute an attack on those individuals or organizations, but on the mechanistic myth.

Except where supported with citations, the discussions in this book reflect the author's personal views, and the author does not claim or suggest that anyone else holds these views.

The arguments advanced in this book are founded, ultimately, on the principles of demarcation between science and pseudoscience developed by philosopher Karl Popper (as explained in "Popper's Principles of Demarcation" in chapter 3). In particular, the author maintains that theories which attempt to explain non-mechanistic phenomena mechanistically are pseudoscientific. Consequently, terms like "ignorance," "incompetence," "dishonesty," "fraud," "corruption," "charlatanism," and "irresponsibility," in reference to individuals, groups of individuals, corporations, institutions, or other organizations, are used in a precise, technical sense; namely, to indicate beliefs, ideas, claims, or practices that are mechanistic though applied to non-mechanistic phenomena, and hence pseudoscientific according to Popper's principles of demarcation. In other words, these derogatory terms are used solely in order to contrast our world to a hypothetical, ideal world, where the mechanistic myth and the pseudoscientific notions it engenders would not exist. The meaning of these terms, therefore, must not be confused with their informal meaning in general discourse, nor with their formal meaning in various moral, professional, or legal definitions. Moreover, the use of these terms expresses strictly the personal opinion of the author – an opinion based, as already stated, on the principles of demarcation.

This book aims to expose the corruptive effect of the mechanistic myth. This myth, especially as manifested through our software-related pursuits, is the greatest danger we are facing today. Thus, no criticism can be too strong. However, since we are all affected by it, a criticism of the myth may cast a negative light on many individuals and organizations who are practising it unwittingly. To them, the author wishes to apologize in advance.

Contents

Preface

This revised version (currently available only in digital format) incorporates many small changes made in the six years since the book was published. It is also an opportunity to expand on an issue that was mentioned only briefly in the original preface.

Software and Mind is, in effect, several books in one, and its size reflects this. Most chapters could form the basis of individual volumes. Their topics, however, are closely related and cannot be properly explained if separated. They support each other and contribute together to the book's main argument.

For example, the use of simple and complex structures to model mechanistic and non-mechanistic phenomena is explained in chapter 1; Popper's principles of demarcation between science and pseudoscience are explained in chapter 3; and these notions are used together throughout the book to show how the attempts to represent non-mechanistic phenomena mechanistically end up as worthless, pseudoscientific theories. Similarly, the non-mechanistic capabilities of the mind are explained in chapter 2; the non-mechanistic nature of software is explained in chapter 4; and these notions are used in chapter 7 to show that software engineering is a futile attempt to replace human programming expertise with mechanistic theories.

A second reason for the book's size is the detailed analysis of the various topics. This is necessary because most topics are new: they involve either

entirely new concepts, or the interpretation of concepts in ways that contradict the accepted views. Thorough and rigorous arguments are essential if the reader is to appreciate the significance of these concepts. Moreover, the book addresses a broad audience, people with different backgrounds and interests; so a safe assumption is that each reader needs detailed explanations in at least some areas.

There is some deliberate repetitiveness in the book, which adds only a little to its size but may be objectionable to some readers. For each important concept introduced somewhere in the book, there are summaries later, in various discussions where that concept is applied. This helps to make the individual chapters, and even the individual sections, reasonably independent: while the book is intended to be read from the beginning, a reader can select almost any portion and still follow the discussion. In addition, the summaries are tailored for each occasion, and this further explains that concept, by presenting it from different perspectives.

The book's subtitle, *The Mechanistic Myth and Its Consequences*, captures its essence. This phrase is deliberately ambiguous: if read in conjunction with the title, it can be interpreted in two ways. In one interpretation, the mechanistic myth is the universal mechanistic belief of the last three centuries, and the consequences are today's software fallacies. In the second interpretation, the mechanistic myth is specifically today's mechanistic *software* myth, and the consequences are the fallacies *it* engenders. Thus, the first interpretation says that the past delusions have caused the current software delusions; and the second one says that the current software delusions are causing further delusions. Taken together, the two interpretations say that the mechanistic myth, with its current manifestation in the software myth, is fostering a process of continuous intellectual degradation – despite the great advances it made possible.

The book's epigraph, about Newspeak, will become clear when we discuss the similarity of language and software (see, for example, pp. 409–411).

Throughout the book, the software-related arguments are also supported with ideas from other disciplines – from the philosophies of science, of mind, and of language, in particular. These discussions are important, because they show that our software-related problems are similar, ultimately, to problems that have been studied for a long time in other domains. And the fact that the software theorists are ignoring this accumulated knowledge demonstrates their incompetence.

Chapter 7, on software engineering, is not just for programmers. Many parts

(the first three sections, and some of the subsections in each theory) discuss the software fallacies in general, and should be read by everyone. But even the more detailed discussions require no previous programming knowledge. The whole chapter, in fact, is not so much about programming as about the delusions that pervade our programming practices, and their long history. So this chapter can be seen as a special introduction to software and programming; namely, comparing their true nature with the pseudoscientific notions promoted by the software elite. This study can help both programmers and laymen to understand why the incompetence that characterizes this profession is an inevitable consequence of the mechanistic software ideology.

The book is divided into chapters, the chapters into sections, and some sections into subsections. These parts have titles, so I will refer to them here as *titled* parts. Since not all sections have subsections, the lowest-level titled part in a given place may be either a section or a subsection. This part is, usually, further divided into *numbered* parts. The table of contents shows the titled parts. The running heads show the current titled parts: on the right page the lowest-level part, on the left page the higher-level one (or the same as the right page if there is no higher level). Since there are more than two hundred numbered parts, it was impractical to include them in the table of contents. Also, contriving a short title for each one would have been more misleading than informative. Instead, the first sentence or two in a numbered part serve also as a hint of its subject, and hence as title.

Figures are numbered within chapters, but footnotes are numbered within the lowest-level titled parts. The reference in a footnote is shown in full only the first time it is mentioned within such a part. If mentioned more than once, in the subsequent footnotes it is abbreviated. For these abbreviations, then, the full reference can be found by searching the previous footnotes no further back than the beginning of the current titled part.

The statement "italics added" in a footnote indicates that the emphasis is only in the quotation. Nothing is stated in the footnote when the italics are present in the original text.

In an Internet reference, only the site's main page is shown, even when the quoted text is from a secondary page. When undated, the quotations reflect the content of these pages in 2010 or later.

When referring to certain individuals (software theorists, for instance), the term "expert" is often used mockingly. This term, though, is also used in its normal sense, to denote the possession of true expertise. The context makes it clear which sense is meant.

The term "elite" is used to describe a body of companies, organizations, and individuals (for example, the software elite). The plural, "elites," is used when referring to several entities within such a body.

The issues discussed in this book concern all humanity. Thus, terms like "we" and "our society" (used when discussing such topics as programming incompetence, corruption of the elites, and drift toward totalitarianism) do not refer to a particular nation, but to the whole world.

Some discussions in this book may be interpreted as professional advice on programming and software use. While the ideas advanced in these discussions derive from many years of practice and from extensive research, and represent in the author's view the best way to program and use computers, readers must remember that they assume all responsibility if deciding to follow these ideas. In particular, to apply these ideas they may need the kind of knowledge that, in our mechanistic culture, few programmers and software users possess. Therefore, the author and the publisher disclaim any liability for risks or losses, personal, financial, or other, incurred directly or indirectly in connection with, or as a consequence of, applying the ideas discussed in this book.

The pronouns "he," "his," "him," and "himself," when referring to a gender-neutral word, are used in this book in their universal, gender-neutral sense. (Example: "If an individual restricts himself to mechanistic knowledge, his performance cannot advance past the level of a novice.") This usage, then, aims solely to simplify the language. Since their antecedent is gender-neutral ("everyone," "person," "programmer," "scientist," "manager," etc.), the neutral sense of the pronouns is established grammatically, and there is no need for awkward phrases like "he or she." Such phrases are used in this book only when the neutrality or the universality needs to be emphasized.

It is impossible, in a book discussing many new and perhaps difficult concepts, to anticipate all the problems that readers may face when studying these concepts. So the issues that require further discussion will be addressed online, at *www.softwareandmind.com*. In addition, I plan to publish there material that could not be included in the book, as well as new ideas that may emerge in the future. Finally, in order to complement the arguments about traditional programming found in the book, I have published, in source form, some of the software I developed over the years. The website, then, must be seen as an extension to the book: any idea, claim, or explanation that must be clarified or enhanced will be discussed there.

The End of Responsibility

The irresponsibility that characterizes the world of programming, and the apathy of the rest of society, form an extraordinary phenomenon – a phenomenon that warrants closer analysis. Why are we ready to tolerate, in the domain of programming, failures that in other domains would be easily recognized to be due to incompetence or corruption? In the present section, I will try to explain this phenomenon by showing that it is an inevitable consequence of our mechanistic culture.

Software Irresponsibility

1

"No one ever got fired for buying IBM." This famous saying illustrates perfectly the irresponsibility that defines corporate computing. The saying dates from the 1970s, and perhaps even earlier, so it also serves to remind us that corporate computing was always controlled by bureaucrats.

What the saying implied was that buying IBM hardware and software was a *safe* decision. If the project failed, no one would be blamed; the conclusion would be that everything humanly possible had been done, and the project would also have failed with any other kind of hardware or software. Buying another brand, on the other hand, was a *risky* decision. A failure in this case might have led to the conclusion that the project would have succeeded had IBM been chosen instead. (In reality, the saying exaggerates IBM's role in absolving decision makers from their responsibility: there were many cases where managers did *not* buy IBM, the project failed, and still no one got fired. Hardly anyone in the world of corporate computing is blamed for failures.)

The saying is about IBM because it was coined at a time when IBM was dominating the computer industry. Here we are concerned, however, with the mentality behind it – a mentality that has not changed.

Because of their ignorance, software practitioners perceive application development as a dangerous phenomenon, over which human beings have little control. Primitive people, when recognizing their own ignorance, invent superstitions and magic systems to cope with difficult situations. Similarly, managers and programmers have created a rich culture of *computer* beliefs, myths, and magic systems. (We discussed this in "Anthropology and Software" in the introductory chapter.) They attribute the success of computing projects, not to knowledge and skills, but to selecting the correct systems and performing the proper acts. Their ignorance has given rise to the idea that the success of a project is due to some unexplainable power, which appears to inhere in certain types of hardware or software but not in others. The bureaucratic mentality is similar to the primitive mentality: both stem from ignorance, and both result in intellectual stagnation. Instead of expanding their knowledge to deal with difficult problems, the software bureaucrats, like the primitives, develop elaborate systems of belief.

IBM's high profits in the early period (profits which allowed it to establish its monopolistic position) were due in large measure to the systematic exploitation of this bureaucratic mentality. IBM was aware of the ignorance of its customers, and their perception of computers not as business tool but as status symbol. Thus, as corporate decision makers could not assess rationally the cost and benefits of business computing, IBM was able to maintain arbitrarily high prices. Instead of helping companies to develop expertise in the programming and use of computers, IBM fostered ignorance and irresponsibility by encouraging everyone to view the perpetual adoption of new equipment, regardless of cost, as a business necessity.[1]

[1] Joan M. Greenbaum, *In the Name of Efficiency: Management Theory and Shopfloor Practice in Data-Processing Work* (Philadelphia: Temple University Press, 1979), pp. 133–135. These issues were brought up at IBM's antitrust trial.

Thus, the attitudes we see in today's decision makers were born in the first years of business computing. Recognizing that customer ignorance is more profitable than expertise, the computer companies played an important part in the evolution of the software bureaucracy. It was against their interest that software practitioners be knowledgeable, responsible, and creative, so they did all they could to reduce business computing to trivial tasks – tasks that appealed *only* to people with a bureaucratic mentality. This ensured that no talented individuals, no true professionals, could remain in the world of corporate computing. In particular, the computer companies contributed greatly to the redefinition of programming expertise as expertise in ways to *avoid* programming, and to depend instead on complicated development tools and ready-made pieces of software.[2]

As a result of this scheme, the typical programmer or data-processing manager emerging from that period is a person who knows very little about computers, programming, or business, but who enjoys a position of prestige in society thanks to the propaganda conducted on his behalf by the elites. These incompetents induce their employers to spend large amounts of money on hardware and software novelties, and in exchange, the elites flatter them by depicting them (with the assistance of the media) as skilled professionals whose expertise is vital to their company's success. Everyone gets to respect them and to depend on them, and they themselves are convinced that what they are doing is important and difficult, when in reality they are puppets manipulated by the elites.

Although IBM no longer dominates the computer industry, the saying can still be used to describe the software bureaucrats. Today no one gets fired for buying Microsoft, or for buying client/server, or object-oriented, or 4GL, or relational, or data warehouse, or cloud, or artificial intelligence, or anything else deemed to be the correct choice.

Take CASE, for example – the idea that it is possible to develop business applications without programming. For many years, CASE was promoted by the software companies, by professors and gurus, by the business and computer media, and by professional associations, as the indisputable next

[2] It is an old maxim in the computer industry that good software sells hardware. But, even if the promotion of various types of software to help sell hardware started as an honest business strategy, the computer companies realized very quickly that, while good software indeed sells hardware, *bad* software sells even more hardware. Most hardware expenses are induced by bad software; and since bad software is the result of programming incompetence, it is not surprising that the computer companies never encouraged programming expertise. There are many ways for programming incompetence to demand more expensive hardware; for example, a more powerful computer is needed to compensate for the inefficiency of an inferior application, or to permit the use of large development environments, database systems, and packaged applications – all intended as substitutes for programming expertise.

stage in "software engineering." We will never know how many billions of dollars were wasted by businesses on this idea, before abandoning it. And the fact that they could not recognize its absurdity just by looking at it demonstrates the ignorance that managers and programmers suffer from. CASE had to be actually tried in thousands of projects before they realized that it didn't work. Even then, they could not see why the idea was fundamentally mistaken, so they learned nothing from this experience. (We discussed the CASE fallacy in chapter 6; see pp. 465–469.)

Needless to say, no one got fired for buying CASE. Nor did its failure affect the credibility of the companies that sold CASE systems, or the experts that advocated it, or the publications that promoted it. They simply went on inventing and promoting other concepts and expedients, for the same managers and programmers who had bought CASE earlier. Like the primitives, the software bureaucrats are not perturbed by the failure of one magic system. Their belief in software magic unshaken, they are trying now other systems. And no one will get fired for buying those systems either.

2

Partial or total software failures are such a common spectacle that they are now taken for granted. Actually, we no longer think of these occurrences as failures: the failing application, or utility, or methodology, or theory, or development system is accepted anyway, or else is abandoned and another one tried, and all this is seen as a normal software-related activity. Anyone close to the world of "information technology" is familiar with disappointing projects – requirements that cannot be satisfied, applications that cannot be kept up to date, projects that take far longer and cost far more than anticipated – but these situations are not seen as failures. New software products are installed every year in millions of places without being seriously used – presumably because they are not, in fact, the "solutions" they were said to be – but they are not seen as failures either. Nor are seen as failures those software projects that were approved on the promise of increased profits, or savings, or efficiency, or productivity, or strategic advantages, or return on investment, when these promises are not even remembered, much less verified, years later.

More and more organizations are facing software-related problems, but no one is referring to them as software failures. If the problem is discussed at all, the next software change is described as an upgrade, or as a migration, or as deploying a more advanced application, or as rearchitecting the system, or as strategic business transformation, or simply as investing in technology. For obvious reasons, it is in no one's interest to dwell on a failure; besides, neither

the programmers, nor the managers, nor the consultants, nor the software companies appear to be responsible for it. So the whole affair is quietly forgotten, and the next project is initiated.

The only events reported in the media are the "success stories": isolated situations carefully selected from the thousands available, and described so as to create in the public eye a bright image of the world of computers and software. In reality we may well have a hundred failures for each success, but one wouldn't think so from reading business and computer periodicals. Only when the loss is in the tens or hundreds of millions of dollars is a software failure likely to be made public. And even then, no one is seriously reproved: after a brief investigation – conducted largely for the sake of appearances – the failure is forgotten.

We are *surrounded* by software failures, but the software propaganda has succeeded in convincing us that these are not failures but normal occurrences. Thus, since they are not failures, no one is to be blamed. Software is designed and programmed by people, is sold and bought by people, is installed and used by people, but when it fails no one is to be blamed. In most cases we know the actual individuals involved in its development, purchase, or installation; but we don't feel that these individuals must be reprimanded, that they are accountable for their work in the same way that physicians, pilots, or engineers are for theirs.

In other professions we have the notions of incompetence, negligence, and malpractice to describe performance levels that fall below expectations. In software-related matters, and particularly in programming activities, these notions do not exist. In other professions we have created codes of expertise based on high performance levels – levels usually attained only after much training and practice; and we have associations safeguarding these codes, watching over the practitioners' conduct, and issuing and revoking licences. No one expects equivalent codes and licences for software practitioners.[3]

For most products, we expect solid warranties regarding quality and

[3] Some attempts *have* been made to establish codes for programming expertise. What these codes promote, though, is not the difficult skills required to solve real problems, but the trivial skills required to use the programming aids prescribed by the software elites. And the purpose of these aids, we saw, is to act as *substitutes* for programming expertise. Such codes, therefore, while appearing to promote professionalism, are in reality a fraud: they are invented by the elites as part of their effort to *prevent* expertise, and to create a dependence on development systems which they control. These are codes befitting a software bureaucracy.

performance; for software, the only warranty we get is that the bytes in the box will be read successfully into our computer. Instead of warranty we get *disclaimers*, and we sign a software agreement which specifies that, regardless of the product's performance, we alone are responsible for the consequences of its use. When other products do not work as promised, we are outraged, complain, return them, and even consider a lawsuit. When software products have deficiencies, or fail to provide the promised "solutions," we gladly expend time and effort dealing with their shortcomings, pay for "technical support," and look forward to the next version – which, we are told, will solve the problems created by the present one.

What is extraordinary, again, is not so much the incompetence and irresponsibility of the software practitioners, as the apathy of the rest of society: our acceptance, our belief that what they are doing represents the utmost that human beings can accomplish in the domain of programming.

If this incompetence and irresponsibility are found today mainly in programming, it is safe to predict that, as our dependence on software is growing, the same incompetence and irresponsibility, and the same tolerance and apathy, will spread into other fields. At issue, therefore, is more than just the current exploitation of society by the software bureaucrats. The corruption we are witnessing today in only one field may be the reflection of a trend that is affecting the entire society, so the study of software irresponsibility may reveal important truths about ourselves and our social future.

Determinism versus Responsibility

1

The idea of responsibility, and the closely related ideas of free will and determinism, are subjects studied by that branch of philosophy known as ethics, or moral philosophy. The *conflict* between free will and determinism, in particular, has troubled thinkers for centuries, and is still being debated.

The problem is simply this: Are human actions, choices, and decisions free, or are they causally determined? Usually, we feel that what we did on a certain occasion was the result of our own volition, that we could have acted differently had we chosen to. This perception is perhaps gratifying, but it is hardly sufficient when our acts affect other persons – when what we do has moral or legal significance. We need, therefore, a dependable method of determining, for a given act, whether it was performed freely or not. When a person performs a wrongful act, must he be blamed? Can we be sure that he could have acted differently and chose the wrongful act freely? And conversely,

are a person's right acts always worthy of praise? Can we be sure that he could have acted differently and made the right choice freely?

It is obvious, then, why the idea of responsibility is related to the conflict between free will and determinism: If we believe in free will, we admit that we are free to conduct ourselves according to our own will; so we must be held responsible for our acts. If, however, we believe in determinism, then our acts, like any event in the universe, are the result of previous events, over which we may have no control; so we must not be held responsible.

Determinism is the thesis that the future state of a system can be determined from its present state, because all events are necessarily caused by some other events. And a belief in mechanism entails a belief in determinism. The mechanistic doctrine is founded upon the principles of reductionism and atomism: it maintains that a complex phenomenon can be seen as the result of simpler phenomena, which in their turn can be broken down into even simpler ones, and so on, reaching ultimately some elementary phenomena that can be understood intuitively. Mechanism, thus, holds that everything can be explained: from phenomena involving objects and physical processes, to phenomena involving human beings and mental processes.

Every entity or event in the present is a result of past phenomena, so mechanism assures us that from a knowledge of the past we can explain fully and precisely the present. Similarly, since present phenomena are the cause of future ones, a knowledge of the present should allow us to predict fully and precisely every entity and event in the future. This fantastic claim is the essence of determinism, and a *necessary consequence* of mechanism. To put it differently, a person who believes in mechanism cannot believe at the same time in *indeterminism* without contradicting himself.

The best-known expression of determinism is the one formulated by Laplace some two centuries ago: The whole universe, and every occurrence in the universe, can be seen as a system of particles of matter acting upon one another according to Newton's law of gravitation. Thus, an infinitely intelligent being, capable of observing the state of all the particles at a particular instant and capable also of solving the pertinent equations, could determine their state – and hence the state of the universe – for any other instant in the past and in the future. This being (known as Laplace's demon) could, therefore, predict every future occurrence in the universe.

Note that it is irrelevant whether such a being exists; only the *idea* matters. The mechanists merely claim that it is possible to reduce to particle mechanics all physical, chemical, biological, psychological, and social phenomena, and hence omniscience – complete understanding of the past and present, and complete knowledge of the future – is within our reach. The mechanists admit that we are far from having achieved this reduction. Still, they say, we are

making progress, and one day we may well become omniscient. This too, however, is irrelevant: even if human beings are incapable of omniscience, the mechanists say, and even if no omniscient beings exist anywhere, this does not alter the fact that the future can be completely determined from the present. The particles and the law of gravitation continue to exist even if Laplace's demon is no more than an idea.

Thus, the mechanists conclude, even if no real or hypothetical being can know everything in the world, it is still true that the world can be explained; it is still true that the state of the universe at any instant is the result of earlier states, that every occurrence is caused by previous occurrences. And, since we are part of the universe, it must also be true that every decision and action displayed by human beings is the result of past occurrences. But most occurrences in the universe are beyond our control; so perhaps all our thoughts, feelings, and acts are due in fact to external causes, not our own volition. The idea of free will is therefore an illusion. And it is not only the possibility of free will that is an illusion, but also the possibility of original deeds, and hence of individual responsibility. Human beings are just mechanical parts in the great machine that is the universe, and all aspects of human existence can be explained and predicted, in principle, with the same equations that describe the motion of bits of matter.

This view, fallacious and distasteful as it is, has dominated science and philosophy since the seventeenth century. The mechanistic doctrine has been a complete failure in the human sciences, where it cannot explain even the simplest phenomena, let alone thoughts and feelings. But its spectacular success in the natural sciences has inspired scientists and philosophers with confidence, and few doubt, even today, that reductionism and atomism will eventually prove to be just as successful in explaining *human* phenomena. Thus, even though mechanism has been shown to work only in a narrow range of physical phenomena, researchers in *all* disciplines see it as their professional duty to adhere faithfully to the mechanistic doctrine. At the same time, because determinism creates such a demeaning view of humanity, many scientists and philosophers are now reluctant to call themselves determinists; and some even deny that they are mechanists.

As I have pointed out, today's mechanistic theories only *appear* less naive than the earlier ones (see pp. 76–78). In reality, current academic research is grounded, just like seventeenth-century science, on the assumption that all phenomena can be reduced to particle mechanics. The theories have not changed, and are as fallacious as ever; but because determinism is no longer fashionable, the bold and straightforward claims of earlier times have been replaced with a mass of sophisticated rhetoric.

Today's mechanists are in a difficult position. They want to give up the idea

of determinism, but to continue to practise mechanism. Mechanism, however, entails determinism, so the mechanists are caught in a self-contradiction. The only way out is to claim that mechanism can be reconciled with indeterminism, that we can have both. And indeed, countless theories have been advanced in the last one hundred years in an attempt to show that mechanism is, in fact, compatible with indeterminism. This notion is nonsensical, of course, but in a culture where so many absurdities have already been claimed in the name of mechanism, one more absurdity makes little difference. This is how Karl Popper puts it: "I personally find Laplacean determinism a most unconvincing and unattractive view.… But it is, perhaps, worth stressing that Laplace does draw the correct conclusions from his idea of a causally closed and deterministic [world]. If we accept Laplace's view, then we must not argue (as many philosophers do) that we are nevertheless endowed with genuine human freedom and creativity."[1]

2

The delusion of determinism is easy to understand if we recall the concept of simple and complex structures. All phenomena are in reality non-mechanistic, the result of many interacting processes; so they can be represented accurately only with *complex* structures. Still, some phenomena can be *approximated* with simple structures. If a phenomenon requires a system of interacting structures, we can usefully approximate it with one structure when the links between structures are weak enough to be ignored. But a simple structure provides both mechanistic and deterministic qualities; consequently, a phenomenon seen as mechanistic will be seen at the same time as deterministic. The determinism of a phenomenon, though, just like its mechanism, is an illusion, an *approximation* of reality. We find a deterministic approximation useful, presumably, for the same phenomena for which we find a mechanistic approximation useful.

We have discovered useful mechanistic approximations for many phenomena – physical ones, in particular; but these phenomena form only a small part of our world. The most important biological processes, and practically all mental and social processes, are complex phenomena; and for them no useful mechanistic representations have been found. Even some physical phenomena are only poorly approximated by mechanistic models (the apparently simple three-body system, for instance, see pp. 107–108). And, most

[1] Karl R. Popper, *The Open Universe: An Argument for Indeterminism* (London: Routledge, 1988), p. 124.

surprisingly perhaps, mechanism has not kept up with the discoveries of modern physics in the field of *elementary* particles, precisely where one would expect reductionism and atomism to work best.

Mechanistic approximation, thus, fails at both low and high levels of complexity: for elementary physical phenomena as much as mental and social phenomena. Ultimately, only a narrow range of phenomena are amenable to mechanistic approximation. So, if the others are non-mechanistic, they must also be indeterministic.

Recall the theories we studied in chapter 3, and their consequences. Those scientists claimed that all human phenomena can be explained with precision, so human beings are in reality deterministic systems. Scientists today are still working on this type of theories; but because the academic fashion has changed, we don't hear the claim that human beings are deterministic systems as often as we did in the past.

A recapitulation of these delusions will help us to understand how they affect our conception of individual responsibility. What I want to show is that, by degrading the concept of indeterminism, the theories of mind – and now also the theories of software and programming – are still claiming, in effect, that human beings are deterministic systems. The conclusion that we are not responsible for our acts follows then logically. Thus, there is no real difference between a theory that *denies* the existence of free will and creativity, as the older theories did, and one that *accepts* these qualities but *redefines* them to match a mechanistic model.

Minds and societies give rise to indeterministic phenomena, so mechanistic theories cannot explain them. The faith in mechanism, however, prevents the scientists from recognizing these failures as falsifications. So, instead of doubting and severely testing their theories, they end up *defending* them. The only way to defend a fallacious theory is by turning it into a pseudoscience: tests are restricted to situations that confirm it; and the theory is repeatedly expanded, by incorporating the falsifying instances and making them look like new features. Eventually, the theory becomes unfalsifiable, and hence worthless. But the scientists do not consider this activity to be dishonest, or unprofessional. On the contrary: because the principles of reductionism and atomism are accepted implicitly as the universal method of science, defending a mechanistic theory is perceived as important scientific work.

Then, even though their theories keep failing, the scientists draw demeaning conclusions about human beings and human societies – conclusions that would be warranted only if their theories were successful. They conclude, in

particular, that all manifestations of human knowledge and behaviour can be deduced from some innate faculties, which constitute a sort of alphabet of human capabilities; so the knowledge and behaviour displayed by each one of us can be described with precision as a function of these innate faculties, just as the operation of a machine can be described as a function of its parts.

Mechanistic theories of mind and society, thus, *inevitably* lead to the view that human beings are deterministic systems: we have no control over our innate capabilities, of course; and if, in addition, our knowledge and behaviour can be precisely deduced from these capabilities, then we can have no greater control over our decisions and actions than machines have over theirs. It is impossible to separate mechanism from determinism.

But, to adhere to the current academic fashion, the scientists claim that mechanistic theories do *not* preclude indeterminism in human affairs. Specifically, they claim that it is possible to explain mathematically all human acts, and all aspects of a human society, without denying free will and creativity. They avoid the contradiction by *redefining* these concepts: from the absolute qualities we take them to be, to some relative qualities, which match their theories. They continue to use the terms "freedom" and "creativity," but instead of *unpredictable* acts, the terms mean now just the freedom and creativity to select any act from a range of known alternatives. Still, the scientists argue, if the number of alternatives is sufficiently large, the new definition does not deny indeterminism. *Their fallacy is to misinterpret indeterminism as a large number of alternatives.* (See also the related discussion in chapter 3, pp. 281–284.)

Mechanistic models are logically equivalent to simple hierarchical structures, as we know. Mechanistic theories, therefore, depict human phenomena as simple structures. Depending on the theory, the starting elements are various bits of knowledge or behaviour, or various propensities, while the values of the top element are the many alternatives displayed by a mind or by a society: all possible knowledge, behaviour patterns, language uses, social customs, and so forth. The mechanistic *software* theories, for their part, depict as simple structures *software* phenomena – the development and use of software. The starting elements are various software entities and various bits of software-related knowledge, while the values of the top element are all the alternatives possible for software applications and their use.

Now, it is quite easy to design a mechanistic model that displays an infinite number of alternatives. And the scientists misinterpret this trivial quality of mechanistic models as indeterminism: they believe that these alternatives are

the same as the infinity of alternatives that make up the actual, complex phenomenon. So, they conclude, a model has been discovered that is both mechanistic and indeterministic.

Mechanistic theories, thus, assume that human minds and societies are akin to devices, in that they can be in a number of states, or can generate a number of alternatives – states and alternatives that we can account for. A device like a die, for example, behaves unpredictably when rolling; but we know that it will display, when stopped, one of six given symbols. Human phenomena are thought to be essentially the same, except for displaying many more alternatives, perhaps even an infinite number of alternatives. Let us use the analogy of the die to examine this fallacy.

A rolling die constitutes a complex phenomenon, the interaction of several phenomena; this interaction is what determines its movement and, consequently, which symbol will be displayed when it stops. The six alternatives form, in fact, only a small part of the actual phenomenon. To describe the complex phenomenon, we would have to take into account many details of the die itself, its movement when rolling, and its environment: its size, its material and weight, the shape of its edges and corners, its starting orientation in space, the form and texture of the surfaces it touches, how variables like the ambient temperature and humidity affect its movement, and the direction and magnitude of the force that sets it in motion. Also, the final state will be, not just the symbol displayed, but the exact position and orientation of the die in space when the rolling stops; that is, not one of six, but one of an infinite number of alternatives. The actual phenomenon, thus, consists of a large number of factors, which can display an infinite number of values, and lead to an infinite number of possible results.

It is not this infinity of alternatives that prevents us from predicting the final state, though, but the fact that the phenomenon is non-mechanistic: we cannot describe with precision how the final state depends on the various factors. Still, nothing stops us from representing the phenomenon with a mechanistic model: all we have to do is depict it with a simple structure where the starting elements are the six symbols, and the top element is the symbol displayed when the die stops rolling. The phenomenon can then be said to be both mechanistic (because we can account for all possible values of the top element from a knowledge of the starting elements) and indeterministic (because we don't know which symbol will actually be displayed).

What we did, obviously, is *simplify* the phenomenon: our model is an *approximation* of the actual die phenomenon. Rather than a complicated movement and a final state determined by the interaction of many factors, we described the phenomenon simply as the selection of one symbol out of six. And "indeterminism" is now, not the uncertainty of the actual phenomenon,

but the uncertainty of not knowing which symbol will be selected. In other words, if we degrade the notion of indeterminism to mean just the uncertainty of not knowing which alternative will be selected from a known range of alternatives, we can claim that the phenomenon of the die is both mechanistic and indeterministic.

Scientists attempt to represent minds and societies with mechanistic models because they believe the indeterminism of human phenomena to be like the indeterminism of the *simplified* die phenomenon. The only difference, they say, is the larger number of alternatives. The indeterminism of human phenomena, however, is like the indeterminism of the *actual* die phenomenon. Minds and societies give rise to complex phenomena, and hence to *true* indeterminism, not the weak indeterminism displayed by a mechanistic model.

We can perhaps delude ourselves that we understand the die phenomenon when we depict it as the selection of one symbol out of six; but this delusion comes to an end when we study the great complexity of the *actual* phenomenon. Clearly, the unpredictability of the final state of the die as a result of all those factors is of a higher level than the unpredictability of selecting one symbol out of six. Thus, the simplified, mechanistic model is useless if what we need is to describe the relationship between those factors and the die's final state.

Similarly, the scientists keep simplifying the complex structure that is a human phenomenon until they manage to depict it with a mechanistic model. But this approximation is too crude to be useful, because it can display only a fraction of the alternatives that make up the actual phenomenon. The scientists stop their simplification at a point where there is still enough indeterminism left to make the mechanistic model somewhat unpredictable; and they misinterpret this unpredictability – which is just the trivial process of selecting one alternative out of many – as the indeterminism of the original, complex phenomenon.

In our analogy, the actual die phenomenon stands for a human phenomenon, and the simplified model stands for the mechanistic model of the human phenomenon. But unlike the six alternatives of the die, the mechanistic model of the human phenomenon, even though a simplified version of the actual phenomenon, still displays an infinite number of alternatives. And this is the source of the confusion. For, the infinity of alternatives found in the mechanistic model is only a fraction of the infinity found in the actual, complex phenomenon; and consequently, the indeterminism of the mechanistic model is only a weak version of the indeterminism displayed by the actual phenomenon. The number is infinite in both cases, but we are witnessing in fact different *kinds* of phenomena: one can be represented mathematically, while the other cannot.

It is the infinity of alternatives displayed by their mechanistic models, therefore, that confuses the scientists: they mistakenly conclude that, being infinite, these must be all possible alternatives, so their models can represent human phenomena.

Mechanistic models fail because they are incapable of displaying the same indeterminism as the actual phenomena. True indeterminism is, of course, the highest form of indeterminism. It is only by choosing a weaker definition that the scientists manage to contrive models that are both mechanistic and indeterministic. But if their goal is to explain human phenomena, degrading the concept of indeterminism cannot help them: all they can discover then is theories that do not work, models that do not approximate the human phenomena closely enough to be useful.

In conclusion, the difference between true indeterminism and its weaker version is not as mysterious as it appears. When attempting to represent a complex human phenomenon with a simple structure, the scientists are committing the two mechanistic fallacies, abstraction and reification: they start from levels that are too high, and they ignore the links with the *other* structures that are part of the phenomenon. Their model, as a result, cannot account for all the alternatives: missing are those caused by interactions at levels lower than the level of the starting elements. The difference between the alternatives generated when starting from low-level physiological elements, and those generated by starting from the higher-level elements, is the difference between true indeterminism and its weaker version.

3

Just like the mechanistic theories of mind and society, the mechanistic *software* theories praise creativity while degrading this concept to match the weak version of indeterminism. Creativity in software-related matters, according to these theories, does not mean the utmost that human minds can accomplish, but merely the selection and combination of bits of knowledge within a predetermined range of alternatives.

Software mechanism holds that software-related phenomena can be represented as simple structures, and that it is possible to account for all the values of the top element in these structures. These values are, in effect, all the applications that can be implemented with software, and all aspects of software use. Software mechanism claims, thus, that it is possible to account for all the alternatives displayed by human minds when engaged in software-related activities. Accordingly, it should be possible to identify the elements that lie just a few levels below the top element. It is these elements, clearly, that give

rise to all the alternatives, so there is no need to deal with lower-level ones. We should treat *these* elements as starting elements, and incorporate them in software devices. Then, by operating such a device, anyone will be able to generate all the values of the top element – all conceivable alternatives in software-related phenomena – simply by selecting and combining built-in elements.

What this means in practice is that we should be able to replace the knowledge and experience needed to create software applications, with the simple skills needed to operate software devices; that is, the skills needed to make selections. Software devices, thus, materialize the belief that what an experienced mind does is merely select and combine alternatives, so the indeterminism we observe in software-related phenomena is just the uncertainty caused by the large number of alternatives.

This interpretation also explains why the software devices provide their operations in the form of selections (menus, buttons, lists, etc.), and selections within selections, instead of allowing us to use them freely – as we use natural languages, or traditional programming languages. Their designers, obviously, are attempting to implement a simple hierarchical structure: the structure of knowledge that, according to their theories, exists in the mind of an experienced person.

So there is no longer a need for each one of us to develop structures in our mind starting from low-level bits of knowledge. Since the operations provided by devices can directly replace high-level knowledge elements, by combining these operations we will be able to generate any one of the knowledge alternatives that human minds can display (or, at least, the important alternatives) without having to possess that knowledge ourselves. The easy skill of selecting and combining those operations, the software mechanists assure us, can be a substitute for the complex knowledge developed in a mind by starting from low levels. The only thing that human beings need to know from now on is how to select and combine operations within the range of alternatives provided by software devices.

Like the mind mechanists, the software mechanists commit the two fallacies, reification and abstraction: they take into account only one structure, ignoring the other relations that exist between the same elements; and they start from levels that are too high. As a result, they lose the alternatives arising from the interactions occurring at levels lower than their starting elements. They note the infinity of alternatives possible even when starting from higher levels, and conclude that, since their software devices can generate these alternatives, they have attained their goal: a mechanistic model that can emulate indeterministic phenomena. Their infinity, however, is only a fraction of the infinity of alternatives found in the actual, complex phenomenon.

By restricting ourselves to simple structures, we are becoming a closed, deterministic society, where only certain alternatives can exist. The danger posed by our software ideology, therefore, is not just the loss of alternatives in software-related matters, but the degradation of minds. Our non-mechanistic capabilities do not simply exist – they develop; and they can develop only when we are exposed to low-level elements, because this is the only way to create all possible alternatives in our minds. If we restrict ourselves to mechanistic knowledge – to simple knowledge structures and high-level starting elements – our minds cannot develop above the intellectual level of machines.

❖

Why is it so easy for the software elites to convince us that our minds are inferior to their devices? Our willingness to accept this notion, despite its obvious fallaciousness, may well be a symptom, an indication of how advanced our mental degradation *already* is. We must use our minds to judge the usefulness of software devices, but it is these very minds that remain undeveloped when we agree to depend on devices. So, if we trust the elites and get to depend on their devices, we are *bound* to lose our ability to recognize how useless these devices actually are. We will believe that the devices are superior to our minds, so we will continue to depend on them; our minds will then be further degraded, in a process that feeds on itself.

This collective mental degradation is an amazing spectacle. We are willingly renouncing our natural mental capabilities, and our responsibility as individuals, and replacing them with a dependence on devices and on the elites behind them. We are degrading our status from creative and responsible individuals to operators of devices. This process, which may well be irreversible, is the ultimate consequence of our mechanistic culture: when we agree to limit ourselves to mechanistic performance, and hence to a fraction of our mental capacity, what we do in effect is fulfil the wish of those scientists who have been telling us for a long time that our minds are nothing but complicated machines.

4

We must not be confused by words. No matter what the mechanists say, their theories necessarily lead to the view that there is no indeterminism in human phenomena. The scientists, and now also the software theorists, may use words like "indeterminism," "freedom," and "creativity," but they are degrading these concepts to mean the selection of acts from a known range of alternatives. This is how they manage to have mechanism and indeterminism at the same time.

Whether their theories describe minds and societies, or software development and use, if they claim that it is possible to account for all the alternatives in advance then what they are promoting is *deterministic* theories; and if they attempt to represent minds and societies with mechanistic models, or to replace knowledge with software devices, then what they are saying is that human beings are deterministic systems.

If the mechanists believe that we are deterministic systems, their conclusion *must* be that we are not free agents, and that we cannot be held responsible for our acts. The theories of mind and society manage to avoid this conclusion by maintaining that it is possible to have both mechanism and indeterminism. And, since these theories do not work anyway, few notice the absurdity of the claim. We didn't change the way we think or behave or speak, to match the mechanistic theories; so the fact that they are self-contradictory can be overlooked. The *software* theories make claims similar to the traditional mechanistic claims, but now we *are* modifying our conception of intelligence and creativity: we are mutating into deterministic beings.

I remarked at the beginning of this chapter that the incompetence and irresponsibility displayed by the software practitioners, and the tolerance and apathy of the rest of us – our belief that they need not be accountable for their deeds as other professionals are for theirs – constitute an extraordinary phenomenon. We are now in a position to explain this phenomenon.

These attitudes are a natural consequence of the mechanistic software culture. It is impossible to have at the same time a mechanistic, and hence deterministic, culture *and* responsibility. If people are seen as deterministic systems, we cannot expect them to possess non-mechanistic knowledge. So, if all they can do is follow theories and methodologies, or operate software devices, we must limit their accountability to the performance of these acts; we cannot hold them responsible for the *results*.

Increasingly, as programmers and as users, we depend on software devices that are based on mechanistic theories. And when we agree to depend on these devices, we agree in effect to modify our conception of human intelligence to correspond to the mechanistic dogma. We agree, in other words, to forgo our non-mechanistic capabilities: we restrict ourselves to mechanistic knowledge, and thereby keep our performance at the level of novices, or machines. This, clearly, is a new development, a new manifestation of the mechanistic myth.

The traditional mechanistic theories tried to explain and predict human acts; and they failed, because we ignored them and remained indeterministic systems. The software theories, when viewed as theories of human capabilities, appear to work; but this is because we have now agreed to restrict our performance to mechanistic levels. Thus, the reason why irresponsibility is especially noticeable in software-related activities is that, in our capacity as programmers

or software users, our transformation from indeterministic to deterministic systems is almost complete. In our software pursuits, we have renounced our non-mechanistic capabilities; so the world of software constitutes, in effect, a social system founded upon mechanistic principles. In this system, people have been absolved from accountability for the consequences of their acts, and the idea of responsibility has been degraded to mean simply adherence to the official ideology. Attitudes that used to exist only in fictional societies, or in totalitarian societies, can now be found in our software pursuits.

Today we can observe this phenomenon mainly in our software-related affairs – programming, in particular – because the mechanistic software theories affect our software pursuits directly. But we should expect to see the same irresponsibility and the same acceptance emerge in other domains, as these domains become dependent on mechanistic software notions. We cannot retain the concept of individual responsibility in a society where we consider individuals to be deterministic systems. Indeterminism is associated with free will, and hence responsibility; so determinism *logically* entails irresponsibility.

5

As our dependence on software is growing, our society is becoming increasingly totalitarian; and this progression can be observed in our attitude toward individual responsibility. Under totalitarianism, the responsibility of the individual is redefined to mean the responsibility of obeying the elite. Terms like "freedom," "creativity," "expertise," and "intelligence" are still used, but their meaning is degraded: individuals are expected to possess these qualities, but the qualities mean now merely the selection of certain acts from a predetermined range of alternatives. Whereas in their true sense the terms describe *absolute* values – the ability to select *any* alternative that can be conceived by human minds – under totalitarianism the terms describe *relative* values: the freedom to perform any activity sanctioned by the elite, the creativity to accomplish a task with the means supplied by the elite, expertise in the kind of knowledge taught by the elite, and the intelligence to appreciate the totalitarian ideology.

Note how similar these concepts are to the mechanistic *academic* ideology: the theories of mind claim that intelligence, creativity, and expertise mean the selection of acts from a predetermined range of alternatives – the range for which human minds are biologically wired; and the responsibility of scientists is redefined to mean the obligation of pursuing only this type of theories. Note also how similar these concepts are to the mechanistic *software* ideology: expertise in programming, and in software-related activities generally, has

been redefined to mean familiarity with the mechanistic software theories, and with the development tools provided by the elite; and responsibility means simply the challenge of keeping up with these theories and tools.

But this is no coincidence. Recall the inevitable progression – mechanism, scientism, utopianism, totalitarianism. Mechanistic beliefs lead to scientistic theories – mechanistic theories of mind and society – which then lead to the utopian vision of a perfect society. The perfect society must be founded upon strict mechanistic principles, but human beings are currently non-mechanistic systems, undisciplined and unpredictable. They must be coerced, therefore, in the name of progress, to abandon the old-fashioned ideas of freedom and creativity, to admit that they are merely small parts in a great deterministic system, and to restrict themselves to mechanistic performance. It *is* possible to create a perfect society, but, unfortunately, only through totalitarianism. This is a passing phase, though. The coercion is only necessary until everyone gets to appreciate the benefits of mechanistic thinking; that is, until every person becomes a deterministic system, an automaton.

Mechanism and determinism are logically related to irresponsibility, as we saw. Thus, one way of judging how close a society is to being "perfect" is by studying its conception of individual responsibility: how people think of themselves, of their rights and responsibilities, is an indication of the stage their society has reached in the progression from a human system to a mechanistic one. What we notice is a shift: from the idea that we are directly responsible for our acts as individuals and as professionals, to the idea that we are only responsible for conforming to a certain ideology, while its validity or morality need not concern us. If it is mechanistic, and hence "scientific," the ideology is believed to embody the absolute truth.

Under Nazism, for example, countless individuals were involved in atrocious crimes, or were watching passively as crimes were being committed all around them, convinced that they were serving their country. By modifying the idea of responsibility, perfectly normal people – people with family and friends, and an appreciation of life, art, and logic – were induced to do almost anything. For an entire society, the idea of responsibility was redefined in just a few years to mean, simply, adherence to the ideology established by their leaders. Pseudoscientific theories on social, racial, and political matters, invented by a small elite, became unquestionable principles. People no longer saw themselves as independent human beings, but as parts of a great mechanistic system, a great historical process that was as inevitable as evolution itself.

When the mechanistic dogma is accepted implicitly, the shift in the meaning of responsibility follows logically. If we believe in mechanistic social notions – if, in other words, we hold certain developments to be historically inevitable – we cannot at the same time consider ourselves to be free agents. All

our thoughts and acts must then conform to these notions, and any doubts seem as absurd as doubting the laws of nature. The way mechanism engenders irresponsibility, then, is by assuming that individuals and societies are parts of a great deterministic system, so everything can be explained precisely and completely – in principle, at least. Our common-sense conceptions of free will, intelligence, creativity, and responsibility are therefore mere illusions. In reality, we *cannot help* thinking, feeling, and acting as we do. Our knowledge, beliefs, and capabilities are determined largely by forces beyond our control. Although we do not yet understand that great system of which we are part, there can be no doubt that it is deterministic; so the conclusion that we are not really free, and hence cannot be held responsible for our acts, remains valid.

6

Nazism is only an extreme example of the progression of a society from liberalism to mechanistic thinking and irresponsibility. Because mechanistic theories of mind end up degrading the notion of responsibility, we can observe this progression, in a milder form, in our own society. Recall the mechanistic delusions we studied in chapter 3. While society has yet to reach the level of degradation depicted by these theories, we can already observe it in the attitude of the scientists themselves. They have degraded the definition of scientific responsibility, from discovering sound and useful theories, to discovering *mechanistic* theories.

Then, since what matters to them is not whether the theory is valid but whether it is mechanistic, the scientists draw from their *failing* theories the kind of conclusions that would be warranted only if the theories were successful. Specifically, they claim that human beings are nothing but complicated machines; that human acts are a function of innate propensities and can be explained with mathematical precision; and that, based on these facts, it is possible to design and implement a perfect society.

Thus, in addition to turning their disciplines into pseudosciences, the scientists promote theories that lead to totalitarianism. But they do it out of a sense of duty: their responsibility, they say, is to promote science, and science means mechanism. They perceive themselves as pioneers, as experts who understand what the rest of us cannot yet see, and who alone have the courage to accept the implications of these discoveries; namely, to accept the fact that human beings are in reality automatons. So it is their duty to enforce these ideas upon us, the ignorant masses. But there is nothing wrong in this, the scientists say; all they are doing is helping to speed up a process of social evolution that was going to take place anyway.

Since mechanism has distorted their own sense of responsibility, it is not surprising that their vision of the perfect society is a deterministic social system where everyone has been degraded just as they have degraded themselves. The only reason we have not yet been turned into automatons is that these scientists lack the power to implement their vision. But it is important to study their mechanistic obsessions and totalitarian attitudes, because the same obsessions and attitudes are now being displayed by our *software* elites; and these elites do have the power to put their ideas into practice – through software concepts and devices.

Recall the linguistic theories we studied in chapters 3 and 4. Discussing the same theories, Roy Harris[2] traces their evolution over the last three centuries, and notes that their foundation on the mechanistic dogma has turned the study of language and mind into a system of belief, a mythology. The current theories, which are merely the latest in our tradition of mechanistic language delusions, claim that human beings have an innate language faculty and that this faculty can be explained with deterministic models. What the mind does when we communicate through language, then, must be akin to what a machine does. So we can view the language faculty as a sort of language machine that runs in the mind.

Because they are founded on mechanistic delusions, our theories end up resembling the mystical conceptions of language held by earlier civilizations. "But no other civilization than ours," Harris observes, "has envisaged language as the product of mysterious inner machinery, run by programs over which human beings have no control. That, it will be said, is just the mythology one might expect of a computer-age society; and so it is.… What is significant is that the new view of language promoted is not a conceptual enrichment of what preceded, but a conceptual impoverishment. A society whose academic establishment accepts with alacrity and even with enthusiasm the prospect of being able to treat verbal communication as a complex form of data-processing is a society which proclaims its linguistic immaturity."[3]

This mechanistic model of mind, Harris continues, leads *unavoidably* to the conclusion that human beings are not free and responsible agents. The ultimate consequence of language mechanism, thus, is not the promotion of invalid linguistic theories, but the undermining of the idea of individual responsibility. Because language fulfils such an important function in society, if we believe

[2] Roy Harris, *The Language Machine* (Ithaca, NY: Cornell University Press, 1987).
[3] Ibid., pp. 171–172.

that we have little control over what we say – if we feel that all we do in reality is operate a language machine – we will necessarily conclude that we cannot be held responsible for any acts involving language. Our speech, our knowledge, our decisions, our social and professional performance, are then largely the result of factors beyond our control.

Such conclusions, like the language theories themselves, "are aberrations which the myth of the language machine unavoidably promotes. Unavoidably, because the workings of the machine are envisaged as totally independent of any criteria of values entertained by the machine's human operators. Thus a form of discourse about language is created which serves either to disengage language from human motives and intentions, or to disguise the extent and nature of that engagement.... The myth of the language machine is a convenient myth because it absolves us from our day-to-day duties as language-makers, and blankets out for us all awkward questions concerning the exercise of authority through language. We purchase this linguistic security at cost price: and the cost is the removal of language from the domain of social morality altogether."[4]

We have already determined the similarity of language and software, so the preceding arguments can also be read by substituting software for language. What we note then is that this analysis describes our software culture even more accurately than it does our conception of language. Our software theories are degrading the notion of individual responsibility just like the language theories. Like the myth of the language machine, the software myth "serves either to disengage [software] from human motives and intentions, or to disguise the extent and nature of that engagement." It "absolves us from our day-to-day duties as [software] makers, and blankets out for us all awkward questions concerning the exercise of authority through [software]." As in the case of language, human beings are perceived as mere operators of software machines; so our responsibility is limited to knowing how to operate these machines. The cost of this security is "the removal of [software] from the domain of social morality altogether."

Unlike the language machine believed to run in the mind, though, the software machines are real: these are the devices supplied by the software elites. Thus, while our degradation through language is a relatively slow process, the same degradation is taking place, at a much faster rate, through software. Since we have agreed to depend on software in everything we do, and have also agreed to restrict ourselves to the kind of applications that can be created with software devices, we are asserting in effect that our knowledge, our capabilities, and our performance are the result of factors beyond our control.

[4] Ibid., p. 162.

❖

Isaiah Berlin[5] shows that, despite their differences, most social and political theories are founded on the premise that social evolution is a deterministic process; that all historical developments are, in fact, inevitable and predictable; and that the only reason we cannot explain them is our limited knowledge.

Not surprisingly, these theories conclude that the ideas of free will and responsibility are illusions, reflections of our present ignorance. Science will allow us, one day, to control minds and societies as effectively as we control machines: "All these theories are, in one sense or another, forms of determinism, whether they be teleological, metaphysical, mechanistic, religious, aesthetic, or scientific. And one common characteristic of all such outlooks is the implication that the individual's freedom of choice … is ultimately an illusion, that the notion that human beings could have chosen otherwise than they did usually rests upon ignorance of facts.… The more we know, the farther the area of human freedom, and consequently of responsibility, is narrowed. For the omniscient being, who sees why nothing can be otherwise than as it is, the notions of responsibility or guilt, of right and wrong, are necessarily empty."[6]

To be as successful as are the theories of the natural sciences, the theories of mind and society would have to explain social phenomena through reductionism and atomism; specifically, they would have to show how social phenomena derive from the knowledge and acts of individual persons. Although they cannot discover such theories, the mechanists continue to believe in the possibility of exact explanations of social phenomena. So they claim that social evolution is brought about, not by people, but by *social forces*. While manifesting themselves through the acts of people, these forces have a character of their own, and are more important than the people themselves.

Depending on the theory, the forces may be described as social classes, political institutions, religions, cultures, economic conditions, or technological changes. All these concepts, however, serve in the end the same purpose: because the mechanists cannot explain social evolution in terms of individual persons, they attempt to explain it by replacing the persons with an abstract entity – a "whole." The theories appear then to explain the evolution, but this is an illusion, because the "whole" is merely an invention of the scientists. Those social forces remain unexplained, and their existence remains unproved. The forces are said to cause social evolution, but they are themselves described in terms of this evolution; so, being circular, the theories are fallacious.

[5] Isaiah Berlin, "Historical Inevitability," in *Four Essays on Liberty* (Oxford: Oxford University Press, 1969). [6] Ibid., pp. 58–59.

All deterministic theories, thus, are alike: "Different though the tone of these forms of determinism may be … they agree in this: that the world has a direction and is governed by laws, and that the direction and the laws can in some degree be discovered by employing the proper techniques of investigation: and moreover that the working of these laws can only be grasped by those who realize that the lives, characters, and acts of individuals, both mental and physical, are governed by the larger 'wholes' to which they belong."[7]

If we believe in the existence of some great social forces that control our life, the necessary conclusion is that it is not we, the individuals, but those forces, that are responsible for what happens in the world: "What the variants of either of these attitudes entail, like all forms of genuine determinism, is the elimination of the notion of individual responsibility.… If the history of the world is due to the operation of identifiable forces other than, and little affected by, free human wills and free choices (whether these occur or not), then the proper explanation of what happens must be given in terms of the evolution of such forces. And there is then a tendency to say that not individuals, but these larger entities, are ultimately 'responsible.'"[8]

These theories are popular because they are comforting. It is reassuring to hear that we must not blame ourselves for failures, that our knowledge and accomplishments are in reality part of a master plan, so whatever we do is necessarily right: "The growth of knowledge brings with it relief from moral burdens, for if powers beyond and above us are at work, it is wild presumption to claim responsibility for their activity or blame ourselves for failing in it.… And so individual responsibility and the perception of the difference between right and wrong choices, between avoidable evil and misfortune, are mere symptoms, evidences of vanity, of our imperfect adjustment, of human inability to face the truth. The more we know, the greater the relief from the burden of choice.… We escape moral dilemmas by denying their reality; and, by directing our gaze towards the greater wholes, we make them responsible in our place. All we lose is an illusion, and with it the painful and superfluous emotions of guilt and remorse. Freedom notoriously involves responsibility, and it is for many spirits a source of welcome relief to lose the burden of both."[9]

Until recently, the transition from determinism to irresponsibility could be observed only in totalitarian societies and in the mechanistic theories concocted by academics. In our time, however, a third possibility has emerged: the software culture. The basic tenets of the deterministic *social* theories can now be found in our *software* theories: the blind faith in mechanism; the belief in historical inevitability; the idea of the "whole" – the mighty forces of progress that are beyond our control even though we are parts of them; and the

[7] Ibid., p. 62. [8] Ibid., pp. 63–64. [9] Ibid., pp. 77–81.

comfort of knowing that, in the final analysis, we are not really responsible for our acts, nor for our failures.

The mechanistic software theories started by affecting only the software practitioners, but they are now spreading into all software-related activities. We have already discussed these theories, and how they have degraded our notions of knowledge, freedom, and creativity. What I want to show here is that, while concerned with software, these theories belong in fact to the same category as the deterministic social theories we have just examined.

The software counterpart of the social forces that control the history of humanity are the forces of *software evolution*. Although we take part in this evolution (by performing software-related activities), only the software elites actually understand it. It is their task, therefore, to help us assimilate software changes; and they do it by inventing concepts and devices that embody the power of software while remaining simple enough for us to use.

Our role in the process of software evolution, then, is to adopt the latest software devices provided by the elites – the latest theories, methodologies, development environments, and applications. And we must not attempt to judge these innovations with our naive standards of good and bad, of success and failure. If, for example, a software device does not provide the promised benefits, or if it creates several new problems for each problem it solves, or if we have to alter the way we run our affairs in order to use it, or if it demands more of our time than we had planned, or if it makes us unduly dependent on the elites, we must not think of these issues as shortcomings, but as a manifestation of software evolution. We must not blame the device, nor its maker, nor ourselves.

Each version of a device is a newer thing, so it constitutes software progress. This evolution is historically inevitable, so we must conform to it even if we do not understand it. Founded as they are on mechanistic principles, the devices are expressions of software science, and their perpetual changes reflect the never-ending process of software evolution. It is ignorance and vanity that prompt us to doubt the innovations, and tempt us to pursue personal ideas. It is presumptuous to think that we can accomplish more with our own minds than by conforming to the forces of progress. The belief in skills or creativity in software-related matters is evidence of immaturity, of imperfect adjustment. The more we learn about software and programming, the further the area of individual choice, and hence of responsibility, is narrowed. Whether programmers or users, we are parts of a "whole," of a great historical process that is beyond our control. All we can reasonably be expected to do is adapt to it, by accepting the devices provided by the elites.

This is what we are told, but the existence of the forces of software evolution, like the existence of the *social* forces of evolution, was never proved. The

generations of software devices are said to be a reflection of this evolution, but software evolution is explained in terms of generations of devices. So, just like the argument for social evolution, the argument for software evolution is circular, and hence fallacious.

Totalitarian systems like Nazism and Communism were said to be an expression of the great forces of social progress; so they represented an inevitable social evolution, and the elites were merely helping to bring about a social transformation that, historically, was *bound* to happen. Similarly, the software devices are said to be an expression of technological forces; so they represent an inevitable evolution of software principles, and the elites are merely helping to bring about a technological transformation that was *bound* to happen. Like the political elites, the software elites see themselves as an enlightened vanguard: they alone can understand this evolution, and it is their duty to make us conform to it.

In reality, the mechanistic software theories are pseudoscientific, as are the social theories behind totalitarianism. The software elites, therefore, are the same kind of charlatans as the totalitarian elites. Like them, the software elites managed to assume control of society by entrancing us with their utopian visions – visions that are in fact absurd, and are accepted only because of our mechanistic tradition. Moreover, by accepting their deterministic theories we are *all* participating in the creation of a totalitarian society, just as the millions of people living under Nazism and Communism were by accepting the deterministic *social* theories promoted by *their* elites.