# SOFTWARE AND MIND

Andrei Sorin

EXTRACT

Chapter 3: *Pseudoscience*
Section *Consequences*

**This extract includes the book's front matter and part of chapter 3.**



This section examines the corruptive effect of the mechanistic ideology in universities, and shows how this ideology leads to fraudulent theories in the human sciences and in software development.

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.

# Consequences

## Academic Corruption

In the previous section we studied some of the more influential mechanistic delusions of our time – modern pseudosciences pursued in universities and accepted by large numbers of scientists. By discussing these pseudosciences here I am making a statement; namely, that I view our *software* delusions as a social phenomenon belonging to the same tradition. (The *language* delusions, as a matter of fact, have contributed *directly* to the software delusions. We will study this link in chapter 4.)

The theories of software engineering – the relational database model, structured programming, object-oriented programming, and the like – are in

the domain of programming what behaviourism, structuralism, or universal grammar are in the human sciences: mechanistic delusions, naive attempts to represent complex phenomena with exact models. They are the work of academic bureaucrats: individuals who cannot make a real contribution to their discipline or to society, and who hide their incompetence by imitating the methods of the exact sciences. Through this imitation they appear to be engaged in serious research, while pursuing in fact a pseudoscience.

One consequence of the mechanistic dogma, thus, is the intellectual corruption it fosters. These theories do not work, and they cannot possibly work; but because mechanism is taken as unquestionable truth, each falsification is seen as a challenge – the challenge to find ways to *deny* that it is a falsification. The theory becomes then unfalsifiable: it changes from a naive hypothesis to a full-scale system of belief, a pseudoscience.

This mechanistic culture is what allows now the *software* elites to deceive society, with *their* mechanistic concepts. For it is in universities and other research institutions that the software fantasies emerge: among individuals whose programming experience is limited to textbook examples, to trivial problems and neat solutions. To them, the possibility of finding exact models for complex, real-world software phenomena is as certain as is the possibility of finding exact models for complex psychological, social, or linguistic phenomena to their colleagues in the human sciences. The software fantasies are not so extraordinary once we recognize their grounding in the mechanistic ideology, and their similarity to the other academic fantasies; nor is extraordinary the dishonesty of their promoters and the evolution of the theories into pseudosciences.

It is impossible to assess the price we pay for these mechanistic obsessions and their ramifications. We cannot even imagine the progress we might have made in the human sciences, had the effort wasted on futile mechanistic theories been invested in other directions, more likely to increase our understanding of human minds and human relations; specifically, in theories that attempt to explain whole human phenomena, rather than break them down into simple and independent processes as if they were engineering projects.

Another consequence of our mechanistic obsessions is the prevention of expertise and responsibility. Workers in all fields are expected to follow blindly the principles of reductionism and atomism, rather than to search creatively for solutions and explanations. Instead of seeking to increase and broaden their knowledge, these two principles – which are taken as "the method of science" – allow them to equate expertise with narrow specialization: knowing as little as possible is perceived as a virtue, as a sign of professionalism. And if a narrow domain still requires too much knowledge, workers invoke these principles again and again, until they finally reach those low levels in the structure of

knowledge where even the most ignorant people can be experts – levels where they only need to deal with trivial and isolated problems.

This trend has affected all occupations that involve knowledge and skills, but is especially noticeable in research work. Science has been redefined: an individual is considered a great scientist simply for discovering a mechanistic theory, regardless of whether the theory works or not. Thus, a mechanistic culture rewards mediocrity and discourages creativity. To be successful in academia, an individual must think like a bureaucrat and must accept blindly the mechanistic doctrine. Moreover, creative individuals who could make an important contribution are ignored, or see their work branded as "unscientific," simply because they reject the mechanistic principles and try to deal holistically with complex phenomena.

And this trend is just as widespread in our software-related activities – in universities, in business, and now even in our personal affairs. An individual is considered knowledgeable simply for accepting the latest mechanistic software concepts, regardless of whether these concepts are valid or not. To be successful in a software-related career, an individual must have the temperament of a bureaucrat, must restrict himself to mechanistic practices, and must display an unwavering allegiance to whichever authority is supporting the doctrine of software mechanism.

Psychologist Abraham Maslow[1] suggests that mechanistic beliefs are a sign of immaturity and insecurity. Instead of seeking to understand the complex reality, the mechanists prefer the comfort of artificial, narrow domains, where it is easy to find theories: "Science, then, can be a defense. It can be primarily a safety philosophy, a security system, a complicated way of avoiding anxiety and upsetting problems. In the extreme instance it can be a way of avoiding life, a kind of self-cloistering. It can become – in the hands of some people, at least – a social institution with primarily defensive, conserving functions, ordering and stabilizing rather than discovering and renewing.… The greatest danger of such an extreme institutional position is that the enterprise may finally become functionally autonomous, like a kind of bureaucracy, forgetting its original purposes and goals and becoming a kind of Chinese Wall against innovation, creativeness, revolution, even against new truth itself if it is too upsetting."[2]

In many academic disciplines, and in our software pursuits, our culture increasingly resembles the culture of primitive societies, or of the West during the Dark Ages, or of totalitarian states. What characterizes these cultures is their dogmatic value system, grounded on belief instead of logic. In our culture the dogma is mechanism. This is a scientific rather than religious or

[1] Abraham H. Maslow, *The Psychology of Science: A Reconnaissance* (South Bend, IN: Gateway, 1966). [2] Ibid., p. 33.

political dogma, but its consequences are the same: intellectual stagnation; an ignorant population susceptible to irrational ideas, and hence to deception and propaganda; and, in the end, a society dominated by corrupt elites that exploit these weaknesses.

❖

In the following subsections, I want to discuss the social and political consequences of mechanistic thinking. Specifically, I want to show that the mechanistic theories promoted in our universities, even though invalid, are shaping the future of our society – by fostering totalitarianism.

The reason we must study the consequences of mechanism is that the belief in *software* mechanism has created the conditions for mechanistic theories to be actually implemented in society. The concepts promoted by mechanistic theories in sociology, psychology, and linguistics have undoubtedly influenced our world view, but they were never implemented on a large scale. These theories may pervade the academic world, but they have no direct application in business, or in politics, or in our social or personal affairs.[3] The mechanistic *software* theories, on the other hand, promise immediate benefits to everyone. They are appealing because they address the use and programming of computers, and we now depend on computers in practically every activity.

For example, corporate managers who have never heard of structuralism, and who would probably dismiss its fantastic claims, accept software ideas described as "solutions," without realizing that these ideas are based on the same mechanistic delusions as the structuralist theories. And liberal politicians who have never heard of behaviourism, and who would never endorse its totalitarian policies, accept the utopian promises of the "information revolution," without realizing that these promises are based on the same vision as the behaviourist theories.

So we accept mechanistic *software* theories, which are just as worthless as the traditional ones, not because we understand their principles better, but because their claims address immediate concerns. And we do not recognize their common totalitarian aspects any better than we do their common mechanistic principles. Thus, to recognize the totalitarian tendencies of the *software* theories, we must start by examining the totalitarian tendencies of the *traditional* mechanistic theories. (These two types of theories are the subject of the next two subsections.)

[3] We must remember, though, that totalitarian systems like Nazism and Communism *were* founded on mechanistic social and economic theories. And, as we will see in chapter 8, the democratic systems too are moving in this direction.

# The Traditional Theories

## 1

What do all mechanistic delusions have in common? If we represent as structures the phenomena they try to explain, then what they all claim is that it is possible to account for all the values of the top element from the values of the starting elements. For behaviourism, the starting elements are bits of behaviour, and the top element comprises all possible behaviour patterns and intelligent acts. For structuralism, the starting elements are bits of knowledge or logic, and the top element comprises all human knowledge, accomplishments, social customs, and institutions. For universal grammar, the starting elements are words and elementary sounds, and the top element comprises all valid sentences and some of the knowledge embodied in sentences. So these theories claim that we can explain precisely and completely, starting with some simple elements, all possible manifestations of the human phenomenon in question – mental acts, social behaviour, linguistic competence, etc.

The significance of the claims is evident, therefore, when the phenomena are seen as structures. We immediately notice that the theories describe *simple* structures. They may use diagrams or equations rather than a structure; but we know that if they attempt to provide a precise explanation, they are deterministic models, so they could also be represented with a simple structure. And we also know why these theories do not work: because the phenomena they try to model can be usefully represented only with *complex* structures.

The fact that they do not work, thus, is not surprising. It is important to note, however, the *claim*, or the *expectation*, that they work. The scientists who defend these theories *wish* them to work. Specifically, they *wish* the phenomena to be simple structures, and the top element to be precisely describable in terms of the low-level elements. But if the phenomena are in reality complex structures, if they are the result of interactions between the simple structures the scientists recognize and some *other* structures, then what these scientists do in effect is deny the importance of those other structures; that is, they deny their bearing on the value of the high-level elements. And what are those other structures? They are the phenomena created by human minds: the knowledge, the experience, the creativity, the intuition of individual human beings.

When behaviourists say that intelligent behaviour can be computed from elementary units of behaviour, or when structuralists say that knowledge and social customs can be computed from elementary bits of logic, or when linguists say that sentences can be computed from words, what they claim in effect is that there is nothing between the low levels and the high levels that is

unpredictable. They claim, thus, that we can describe the high levels in terms of the low ones just as we describe the operation of a machine in terms of its subassemblies and elementary parts.

But in the case of human beings and human societies, the high levels *are* unpredictable; and this unpredictability is what we understand as creativity, free will, and indeterminism. The indeterminism is caused by the complexity of interacting structures: the knowledge structures formed in individual minds, and the structures formed by many minds in a society. The structures studied by behaviourists, structuralists, and linguists are indeed among the structures that make up minds and societies. Taken alone, though, these structures cannot explain entire human phenomena; and this is why their theories do not work. In the end, the failure of the mechanistic theories constitutes corroborating evidence (to use Popper's principle) for *non-mechanistic* social and psychological theories: for theories that endow human beings with free will and unbounded creativity.

Mechanistic theories fail because they do not recognize the unique knowledge structures that can develop in a mind. Thus, the failure of these theories ought to enhance our respect for the potential of human beings, for the creativity of each individual. Instead, the scientists insist that these are not failures but merely setbacks, that they *will* eventually find mechanistic theories of mind and society.

Our mechanistic culture has given rise to this incredible spectacle: in a democratic society, in the name of science, renowned professors working in prestigious universities believe it is their duty to prove that human beings have no value. For, by denying the bearing that the knowledge structures present in our minds have on the structures studied by their theories, these scientists deny the unique contribution that each individual can make. By claiming that human phenomena can be explained with mechanistic theories, they claim in effect that these phenomena can be explained without taking into account the knowledge structures developed by individual minds.

And this is not all. Although these theories do *not* work, the mechanists use them to draw sweeping conclusions about man and society – the kind of conclusions that one would draw if the theories *did* work. Specifically, they maintain that human freedom and creativity are only illusions, prescientific notions, not unlike the ancient beliefs that the earth is the centre of the universe, or that Man was created in God's image. Hence, just as science has shown that the earth is merely another planet, and that Man is merely a higher animal, we must trust science again and resign ourselves to the fact that we

cannot be truly creative: everything we do is dictated by our genetic structure, or by our environment, or by other factors over which we have no control. Human beings are in reality nothing but machines – complicated ones perhaps, but machines nevertheless.

Thus, Skinner could only confirm his behaviourist theory in contrived laboratory experiments with rats and pigeons, but concluded that there is an urgent need to apply this science of behaviour to the shaping of human minds and societies. All human acts are the result of external influences, he says, and it is a mistake to believe that we are free and responsible agents. So, rather than allowing ourselves to be controlled by whoever has the power to influence us – parents, teachers, friends, advertisers – it is best to allow a hardheaded government and expert behaviourists do that. These elites would use objective principles and rigorous methods to shape the personality of each individual starting from birth, and thereby create a society of perfect citizens: "What we need is a technology of behavior.… But a behavioral technology comparable in power and precision to physical and biological technology is lacking."[1] This is where the science of behaviourism can help: the conditioning techniques that seem to work for the rats and pigeons trapped in a Skinner box in a laboratory must now be used for the people that make up modern society.

What prevents us from creating this progressive system is our democratic prejudices; that is, our naive belief in human freedom and dignity – notions that the science of behaviourism, according to Skinner, has shown to be illusory anyway: "The conception of the individual which emerges from a scientific analysis is distasteful to most of those who have been strongly affected by democratic philosophies."[2] Skinner was so confident in the potential of behaviourism to solve our social problems that he wrote a science-fiction novel to depict the kind of society we could create through "behavioural technology."[3]

It is significant that Skinner's ideas were very popular and became somewhat of a cult in America, especially among intellectuals. It is also significant that most of those who *rejected* Skinner's utopia did so because they found his behavioural technology objectionable on humanistic, not scientific, grounds: how outrageous that a professor from Harvard University is promoting totalitarianism. Few realized that the first objection to behaviourism must be that it is a pseudoscience, that it does not work, that it is founded on fallacious concepts of mind and society. And what ought to be outrageous is that our universities foster the corrupt environment where pseudoscientists like Skinner can peddle their theories.

[1] B. F. Skinner, *Beyond Freedom and Dignity* (New York: Knopf, 1972), p. 5.
[2] B. F. Skinner, *Science and Human Behaviour* (New York: Free Press, 1965), p. 449.
[3] B. F. Skinner, *Walden Two* (New York: Macmillan, 1948).

The structuralist theories work no better than the behaviourist ones, but their defenders do not hesitate to conclude that human freedom and creativity, in the sense in which we generally understand them, are mere illusions. When we acquire skills and knowledge, when we invent something or solve a problem, when we develop social customs and institutions, all we do in reality is select various acts from a predetermined range of alternatives – the range for which our brains are biologically wired.

Thus, Lévi-Strauss held that the set of possible social customs is analogous to the periodic table of chemical elements: all a society does when adopting a certain custom is select, perhaps randomly, one of the slots available in this table. And Piaget held that the mental development of children is analogous to an increase in the number of levels in a hierarchical structure of binary operations – the structure which is built into the human brain, and which, ultimately, determines our mental capabilities. As individuals or as societies, human beings can be no more creative or free than programmed computers. What we like to think of as creativity and free will is only an illusion caused by the large number of available alternatives.

❖

The case of Chomsky and his universal grammar is especially interesting, because Chomsky himself draws attention to the harmful influence that theories of mind can have on social and political ideologies. He stresses that the innateness hypothesis behind his linguistic theory postulates a view of human nature in the *rationalist* tradition. Rationalist philosophers, starting with Descartes, held that we possess certain mental capabilities simply by being born human; and, although we acquire much knowledge later, our innate capabilities restrict and structure forever what we can know. The rival philosophical school of *empiricism*, on the other hand, holds that human minds are empty at birth, that everything we know comes from interacting with our environment, and that there are no innate restrictions on the kind of knowledge we can acquire.

Chomsky points out that the rationalist view is conducive to ideologies that defend freedom, equality, and respect for the individual, whereas the empiricist view is conducive to ideologies that support authoritarianism, inequality, and exploitation. Specifically, if human beings are empty organisms at birth, as the empiricists say, this means that by their very nature they have no rights; so there is nothing wrong in moulding them to fit a certain policy. Thus, theories like behaviourism, ideologies like Nazism and Communism, and even democratic systems where various elites are permitted to control society, demonstrate the danger of the empiricist view: the leaders can invoke

the idea of human nature to justify the control of knowledge. Rationalists, on the contrary, respect the fundamental rights of the individual – the right to live free, to develop any personality, to pursue any lifestyle – simply by recognizing that human beings possess from birth some important and immutable faculties.[4]

Accordingly, says Chomsky, while rationalism appears to postulate a more limited view of human capabilities, it is in fact the one philosophy that defends individual freedom and creativity. He admits that the hypothesis of an innate language faculty restricts the types of languages that humans can acquire, and the types of sentences – and hence also the types of ideas – that they can create; and he admits that, if everything we can know is governed by innate faculties, similar restrictions apply to all other kinds of knowledge. But, he reassures us, we needn't worry that these restrictions limit our creativity or freedom, because we still have a very large number – indeed, an infinite number – of alternatives *within* the boundaries of innate capabilities. We can create an infinite number of sentences, for instance, despite the severe restrictions imposed by universal grammar.

Chomsky's thesis, however, is fallacious and dangerous. We can agree with him that the concept of empiricism has been distorted and abused by certain ideologies. And, even without accepting his hypothesis of innate faculties, we can agree that our mental capabilities are structured and restricted by certain biological characteristics. But these facts do not warrant his conclusions.

Chomsky's mistake is to assume that, if our mental capabilities lie within a certain range, we should be able to discover a deterministic model that accounts for all possible human acts (because these acts necessarily derive from that range of capabilities). His mechanistic theory of mind compels him to *degrade* the definition of creativity: from the capacity to perform *unpredictable* acts, to the capacity to *select* an act from a predetermined range of alternatives. The traditional view is that creativity gives rise to an infinity of alternatives, and, in particular, to alternatives that were not known in advance. Chomsky, though, believes that we can *account* for these alternatives – simply by inventing a deterministic model that generates an infinity of sentences, ideas, and types of knowledge. But the infinity displayed by a deterministic model is only a fraction of the *real* infinity of alternatives that human minds can develop. (This, obviously, is why his theory doesn't work.)

Chomsky speaks eloquently of human freedom and creativity, but at the same time he attempts to determine with precision all the manifestations of creativity. He seems oblivious to the self-contradiction. For, if there were ways

[4] See, for example, Chomsky's *Language and Politics* (Montréal: Black Rose Books, 1988), pp. 594–595, and *Reflections on Language* (New York: Pantheon Books, 1975), pp. 127–133.

to account for all possible human acts, it would be absurd to call the quality involved in performing these acts "creativity." No surprises would be possible – no exceptions, no novelty, no originality. Anything an individual would do could be shown to be derivable independently of that individual. To put this differently, if Chomsky's theory worked, we could implement it with software; a computer would then perform exactly the same acts as human beings (would generate, for example, the same sentences and ideas), but could the computer be said to be free or creative in the human sense? Determinism is the opposite of freedom, but Chomsky wants to have both: Chomsky the humanist is concerned with freedom, while Chomsky the scientist is searching for a theory that would make a mockery of freedom by showing that a programmed machine can be identical intellectually to a free human being.

Just like the mechanistic theories of mind in the field of artificial intelligence, Chomsky's theories are, in effect, an attempt to replace human minds with software. And, with microprocessors becoming more and more powerful, some of these theories can already be implemented with just one semiconductor chip. They may not state it, and they may not even realize it, but what all these researchers are claiming, essentially, is that we will soon be able to replace human beings with inexpensive devices. The important point, again, is that although these theories do *not* work, the researchers, and the lay people who trust them, are convinced that soon they *will* work, and are therefore developing a world view that reflects these theories. Nor should we forget that our society is already dominated by political and business elites who hold the same conviction, and who are planning our future accordingly. It is not difficult to imagine the kind of future these elites are preparing for us, if they believe that human beings are not very different from expendable semiconductor chips.

In conclusion, Chomsky's preference for a rationalist theory of mind, rather than an empiricist one, is irrelevant when rationalism is supplemented with a *mechanistic* theory of mind. It makes little difference which philosophy one starts with, if one ends by claiming that deterministic models of mind are possible.

Chomsky's case, then, is a good example of the corruptive effect of the mechanistic dogma. Through writings and lectures, he has become known throughout the world as a humanist. His efforts as a scientist working in the mechanistic tradition, however, are harming the humanistic cause more than his efforts as a humanist can help it.

Geoffrey Sampson[5] notes that Chomsky's impoverished definitions of freedom and creativity provide the common philosophical foundation for both

[5] Geoffrey Sampson, *Liberty and Language* (Oxford: Oxford University Press, 1979).

his linguistic and his political theories. As language users, Chomsky says, we are restricted genetically to certain types of grammatical constructions; hence, the creativity we display in speech is in reality only the capacity to select utterances from a certain range of alternatives. Similarly, as citizens, we are restricted genetically to certain types of achievements; hence, there is nothing wrong in defining freedom as merely the right to pursue any ideas within a prescribed range of alternatives: "Chomsky has misappropriated the term 'creative' as he misappropriated the term 'free.' In each case he uses the term in a sense that conflicts with its standard usage; but, by contrasting 'freedom,' or 'creativity,' in his impoverished sense with something that is even further removed from the notion usually associated with the respective word, he invites us to overlook the fact that what we usually mean by the word is something different from both his alternatives."[6]

Chomsky contrasts his theories to those of authoritarian ideologies, which deny freedom and creativity altogether, and which hold that human nature must be moulded to fit a general plan. When presented from this perspective, his own views appear enlightened and liberal. His theories are dangerous precisely because he appears to be defending freedom while defending, in fact, not the traditional concept of freedom, but an impoverished version of it: "The adverse consequences of scientism stem from its assumption that all human phenomena can be analysed by the scientific method; creativity is an exception, since acts which are truly creative cannot, by definition, be predicted. To the question 'Who, in the contemporary intellectual world, most stresses the importance of human creativity?', the answer must undoubtedly be Noam Chomsky.… Yet, when we ask what Chomsky means when he calls men creative, he turns out to refer to our ability to behave in conformity to certain fixed, rigorous rules."[7]

## 2

By way of summary, I want to show how the mistaken conclusions drawn by these scientists can be traced to the mechanistic fallacies. The scientists are evidently fascinated by the fact that simple structures, when used as mechanistic models, can generate a large number of different values for the top element. They can generate, in fact, an *infinite* number of values. Depending on the theory, these values represent the different alternatives displayed by individuals or societies in their knowledge, their behaviour, their traditions, their sentences, etc. So, the scientists conclude, mechanistic models can

[6] Ibid., p. 106. [7] Ibid., pp. 107–108.

account for *all* possible alternatives: all knowledge and behaviour of an individual, all customs and language uses of a society, and so forth.

But a large number of values – even an infinite number – does not necessarily mean that the model can account for *all* possible alternatives. We can demonstrate this with a simple example. It is easy to create a device (a piece of software, for instance) that generates numeric values according to certain rules: values limited to a given range, or values that are prime numbers, or integers. Consider now the set of all numeric values. Although the subset of integers is an infinite number, in practice there are infinitely more fractions than integers (there are, in fact, an infinite number of fractions between any two integers). Consequently, while accounting for an infinity of values, a device that generates integers accounts for an infinitely small subset of all possible numeric values. There are many similar examples, so it is important to bear in mind that an infinite number of alternatives may not mean *all* the alternatives.

What these scientists are seeking is a fully specifiable model that can account, nevertheless, for all the alternatives displayed by human minds. They hope to find, in other words, a deterministic model that can account for indeterministic phenomena – for the creativity and unpredictability of human acts. They are misled by the infinity of alternatives that their mechanistic systems can generate, and conclude that they have discovered such a model. They misinterpret this infinity of alternatives as equivalent to creativity and unpredictability – equivalent, that is, to all possible alternatives. As we just saw, it is easy for a mechanistic system to generate an infinity of values in a given domain while failing to account for all values possible in that domain. Thus, the infinity that the scientists notice in their models is *not* the infinity that gives rise to indeterminism, to creativity and unpredictability. Mechanistic theories fail because, even though explaining an infinity of acts, this infinity is a small subset of all possible human acts. (See also the related discussion in chapter 8, pp. 814–817.)

The scientists start with a reasonable hypothesis: the idea that human beings are restricted in the types of knowledge they can acquire, and in their behaviour patterns, by some innate capabilities. Now, no one can doubt that the basic human faculties *are* bounded by some low-level physiological processes occurring in the brain. The high-level phenomena of mind and society must then reflect these limitations, just as our physical characteristics and abilities reflect our genetic code.

But, to be consistent, the scientists ought to build their theories starting from those low-level physiological processes (from neurons, for instance). Instead, the starting elements in their structures are relatively high-level entities: for linguists they are phonemes and words; for behaviourists they are simple movements, reflexes, and the like; for structuralists they are the binary

opposites found in thought, or in stories and traditions. These are not the simplest entities from which human phenomena are made up, but merely the smallest entities that we notice or understand. The scientists believe that they can start their projects from any level of abstraction they like, so they choose high levels, which are more convenient. They base their theories on mechanistic processes assumed to occur at low physiological levels; but then they ignore the low levels, and the many intermediate levels, and build their models starting from some arbitrary, relatively high-level, entities.

Their mistake, thus, is to use as building blocks in their models, elements that are not independent. Elements like words, or limb movements, or pieces of mental logic, are related – both mutually and to other *types* of elements; so they give rise to complex, not simple, structures. They are related because they serve as elements in other structures too, besides the particular structure that each scientist is concerned with. All these structures are formed at the same time from the low-level physiological elements, so they reflect the countless interactions that take place at levels *lower* than the level where the scientists decided to start their projects.

When ignoring the interactions – when assuming that those starting elements are independent – the scientists separate the one structure that forms their particular project from the complex structure that is the human phenomenon. The top element in the complex structure represents all the alternatives that human beings can display; and the theory fails because it can account for only *some* of these alternatives – the ones that would occur if those starting elements were indeed independent. The theory may well account for an infinity of alternatives, but this is still a small fraction of the alternatives that constitute the actual phenomenon. When attempting to represent a complex phenomenon with a simple structure, and when starting from higher levels, the scientists are committing both fallacies, reification and abstraction. The dramatic reduction in alternatives is then the impoverishment that these fallacies inevitably cause.

We can understand now why these scientists are convinced that freedom and creativity are illusory, that what we perceive as free will is merely the freedom to select any act from a predetermined range of alternatives. They base this notion on the existence of low-level physiological elements, and on the assumption that, *in principle*, we can account for all the alternatives generated by these elements. At the same time, they admit that they *cannot* develop their models starting from these elements. So they continue to claim that freedom and creativity mean just a selection of alternatives, even while starting their models from much higher levels – as if the infinity of alternatives they *can* account for were the same as the infinity generated by the low-level elements. They are seeking mechanistic explanations, but they are *not* following the

mechanistic doctrine: a model cannot explain mechanistically a given phenomenon unless the starting elements are atomic and independent, and *their* starting elements are neither. (Even the low-level physiological elements, which they *fail* to reach, are only *assumed* to be atomic and independent.)

The scientists believe that it will soon be possible to explain all the alternatives and thereby account for the full range of possible human acts. But this optimism would be warranted only if their theories were completely mechanistic. They admit that they cannot find a true mechanistic explanation – a continuous series of reductions from the human phenomena to some atomic and independent entities – but they refuse to take this failure as evidence that human phenomena are indeterministic, and hence unexplainable with mechanistic theories.

Thus, no matter how we feel about mechanism and about mechanistic explanations of human phenomena, the conclusions drawn by these scientists are unjustified simply because their theories are fallacious even *within* the mechanistic doctrine. The indeterminism, the creativity and unpredictability of human minds, are precisely this impossibility of finding a set of atomic and independent entities for the starting elements.

## The Software Theories

### 1

The evil concepts of mind and society engendered by the traditional mechanistic theories are being implemented already, without anyone noticing, through the mechanistic *software* theories. While our democratic system has safeguards to protect society from the political consequences of the traditional mechanistic delusions, no safeguards exist to protect us from the ideas promoted by the software elites. And few of us recognize the similarity between the social systems envisaged by the traditional mechanists, which we reject as totalitarian, and those promoted by software experts and by the software companies, which we perceive as scientific and progressive.

Recall the distinguishing characteristic of the mechanistic delusions we studied in this chapter: the claim that, in a complex structure, it is possible to derive with precision the value of the top element from the values of the starting elements. The complex structure represents a complex human or social phenomenon, so what the mechanists are claiming is that a simple structure – a deterministic model – can account for all possible manifestations of the complex phenomenon. They are claiming, in other words, that the other structures that are part of the phenomenon are unimportant. But the other

structures are the knowledge structures present in human minds, or the social structures formed by the interaction of many minds. So, by making this claim, the mechanists are depreciating the role played by each mind in the complex phenomenon. The reason these theories are failing is that the assumption is wrong: the contribution made by individual minds is in reality an important part of the phenomenon.

In the case of software theories, the complex structures comprise the various activities performed by people when developing and using software applications. And what the software theories claim is that it is possible to account for the top element of these structures from a knowledge of their starting elements. Accordingly, we should be able to replace the knowledge involved in those activities with formal methods and, ultimately, with software devices based on these methods.

In programming activities, the starting elements are details like the definitions and statements used in an application, the fields in data files or display screens, and the software-related acts in the human environment where the application is used. The top element consists of the combinations of operations performed by these applications, and the business, social, and personal needs they serve. The knowledge and experience of programmers and users provide the structures that, together with the structures formed by the elements just mentioned, give rise to the complex phenomena observed when a social or business environment depends on software.

The software theories, though, claim that these are not complex, but mechanistic, phenomena: the top element can be described, precisely and completely, as a function of the starting elements. Whether they invoke mathematical principles or the similarity of programming to manufacturing, the appeal of these theories is understandable. Let us design our applications as hierarchical structures, we are told, as neat structures of independent modules. We should then be able to develop applications of any size and complexity simply by combining these software parts, one level at a time: starting with some atomic entities whose validity is established, we will create larger and larger software subassemblies – each one guaranteed to be correct because built from proven parts – until we reach the top element, the complete application.

All theories, and all methodologies and development environments, are grounded on this principle. But the principle is invalid, because even the smallest software elements share attributes, and are therefore interrelated. Thus, like the mechanistic theories of mind and society, the software theories praise reductionism and atomism, and at the same time they violate these principles by using starting elements that are not atomic and independent. Also like the traditional theories, the software theories are failing because

these elements give rise to multiple, interacting structures. Applications, and software-related activities generally, are not the simple hierarchical structures the mechanists assume them to be. (We will study the various types of software structures in chapter 4.)

## 2

Psychologists, sociologists, and linguists think that mechanistic theories can account for all the alternatives displayed by individuals and societies: all possible manifestations of knowledge, behaviour, customs, language, and so forth. Their theories account for only *some* of the alternatives, but despite these failures, the scientists conclude that what human minds do is simply select various acts from a predetermined range of alternatives: what we perceive as free will or creativity is an illusion caused by the large number of alternatives that minds can select from. Let us see how the mechanistic *software* theories lead to the same conclusion.

Like the mind mechanists, the software mechanists are searching for a deterministic model that can account for indeterministic phenomena – software-related phenomena, in this case. Specifically, they hope to find a mechanistic model that can account for all the alternatives displayed by human minds when engaged in software development and use. And they think that if a model can account for an infinite number of alternatives, this means that it can account for *all* the alternatives.

But we saw how easy it is for a mechanistic model to display an infinity of values while accounting, in fact, for only a small subset of the possible values. The alternatives in software-related phenomena are all the applications that can be implemented with software, and all the business, social, and personal aspects of programming and software use. These alternatives constitute an infinite number, of course. And, as is the case with the other theories, the infinity of alternatives that make up the real software phenomena is infinitely greater than the infinity of alternatives that a mechanistic model can account for. The difference is seen in the perpetual need for new programming theories – a need arising, obviously, from the failure of the previous ones.

While the mind mechanists attempt to represent with exact models *mental* phenomena, the software experts attempt to represent with exact models *software-related* phenomena. This must be possible, they argue, because the totality of software-related human acts can be described with precision as a function of some low-level elements. We should be able to build software devices, therefore, which incorporate these elements. These devices will then allow us to generate any software-related structure – that is, any alternative of

the phenomena of programming and software use – without ever again having to start from the low levels.

True to the mechanistic doctrine, the software experts attempt to explain complex software phenomena by repeatedly reducing them to simpler ones. And in so doing they commit both fallacies, reification and abstraction: they take into account only one structure, failing to see that software entities are related in several ways at the same time; and they stop the reduction long before reaching the lowest levels.

The experts notice that in processes like manufacturing we benefit from the mechanistic principles while starting our structures, nevertheless, from high-level elements (prefabricated subassemblies), and they conclude that software-related processes, too, can start from high levels. They also notice that physical entities can function as starting elements only if independent, and they conclude that software entities, too, can be independent: just as, in a physical structure, the internal properties of one subassembly are unrelated to those of the others (so that we can build them independently of one another), the internal operation of each software module can be unrelated to those of the others.

Both conclusions, however, are unwarranted. While it is true that we can build software structures starting from independent, high-level elements, if we limited ourselves to such structures we could represent with software only *a fraction* of our business, social, and personal affairs.

The experts treat software development as a manufacturing process because they don't appreciate how much richer are human phenomena than physical ones. The number of alternatives lost when we start from high levels and when we separate structures in *physical* phenomena is relatively small, and we gain important benefits in return; when we ignore the low levels and the links between structures in *human* phenomena, we lose an infinity of important alternatives.

Simple structures and high-level starting elements – that is, the use of standard parts and subassemblies – are acceptable in activities like manufacturing because there are only a few kinds of tools, appliances, vehicles, etc., that are useful, or convenient, or economical. The same is not true, however, in mental and social phenomena. There is an infinity of sentences, ideas, customs, cultures, forms of knowledge, and types of behaviour that are correct, or practical, or suitable. Unlike physical processes, therefore, our models of human phenomena cannot be restricted to simple structures, and cannot start from high levels. For, the loss of alternatives is then so severe that the benefits of simple structures and high starting levels become irrelevant: the number of human acts the model can explain, relative to those it cannot explain, is too small for it to be useful.

Software-related phenomena, in particular, despite their dependence on physical structures like computers, are largely *human* phenomena, because they entail various intellectual and social processes. Consequently, simple software structures and high-level starting elements can account for only *some* of the alternatives, just as the mechanistic models of mind and society can explain only *some* aspects of human phenomena.

## 3

The immediate benefit of software mechanism is thought to lie in explaining the phenomenon of programming itself. If we view as hierarchical structures not just the applications but also the activities and the mental acts involved in developing applications, we should be able to account for all the values displayed by the top element; that is, all the combinations of applications, requirements, software concepts, etc. We will then search for some *high-level* entities that can be used as starting elements in these structures. And, once we discover these entities, we will no longer have to develop applications starting from low levels.

Everyone agrees that it is possible to develop all conceivable software applications if starting with *low-level* elements. The promise of software mechanism, thus, is not to enable us to perform tasks that we could not perform otherwise, but to perform them more easily: less work and lower skills are needed to reach the top element – the application, the business system – when starting with higher-level elements. So what software mechanism promises us in effect is devices that would permit us, not only to attain everything that can be attained through software, but to attain it sooner, and with less knowledge. This is true, we are told, because the infinity of applications possible when starting from the higher levels is about the same as the infinity we could create by starting from the low levels.

It is not surprising that the scientists make such fantastic claims; after all, these ideas are nothing but the software counterpart of the theories of mind that mechanists have been proposing for centuries. What *is* surprising is that we all accept the claims now, and that we continue to accept them even as we see them refuted in practice.

A little thought will reveal the true nature of these claims. We are told that software devices will permit us to create *any* software applications, or to address *any* software-related matters, *without* the need to develop all the knowledge that human minds *can* develop. So, by stating this, the experts admit in effect that we *can* attain greater knowledge; we simply *do not need* that extra knowledge. But if we *are* capable of greater knowledge in software-

related matters, we necessarily could, through that knowledge, perform certain tasks, or develop certain ideas, or create certain applications, or make certain discoveries, that we cannot *without* it. To say that those additional alternatives are unimportant, that they represent an insignificant part of the potential achievements of human minds, and that they can be forsaken, is to make an astonishing statement about the future of humankind. For, we can only judge how important or unimportant those alternatives are by *having* them first; we cannot know in advance how human knowledge will evolve, or what our future needs and capabilities will be.

By making such statements, the software elites are claiming in effect the right to decide what knowledge and mental capabilities we, and future generations, are permitted to acquire. They see themselves as an enlightened vanguard: they are the select few who can appreciate the future of software, so it is their duty to guide us.

It is in the interest of the elites to maintain a general state of ignorance in all software-related matters, and the mechanistic software ideology is an essential part of this plan: we perceive the theories, methodologies, and devices as expressions of software science, even as they are preventing us from using our minds. By controlling the way we create and use software, the elites are restricting those aspects of our life that depend on software to a certain range of alternatives. But few of us realize just how narrow this range is, how many alternatives are lost when we are limited to mechanistic software thinking.

We would have no difficulty recognizing a similar limitation in areas in which we are already knowledgeable – other professions, or naturally acquired skills. Programming, however, was taken over by incompetents and charlatans before a body of responsible professionals could emerge; and as a result, we believe that what these impostors are doing is the utmost that a society can attain in the domain of programming. Since the programming we see is the only kind we have ever had, we cannot know how limited and inefficient our software-related affairs are. We cannot even imagine a society where programmers are as competent in their work as other professionals are now in theirs; that is, a society where programmers are permitted to attain the highest skill levels attainable by human minds.

The mechanistic software ideology, thus, fosters incompetence among programmers – but also among software *users*, because their performance is impaired by the inadequate applications. These conditions, then, impoverish our life by limiting our expectations in all software-related matters to only a fraction of the possible alternatives: those that can be accounted for through mechanistic concepts. Although our minds are capable of non-mechanistic knowledge, and we could therefore have infinitely more alternatives in our software-related affairs, as long as we limit ourselves to mechanistic thinking

we cannot know what these alternatives are. We trust the software devices provided by the elites, while the real purpose of these devices is to induce a state of ignorance and dependence. We like the many alternatives that we gain so easily through devices, and we are convinced that they constitute *all* the alternatives. We are unaware of the *missing* alternatives, because the only way to attain them is by starting our software-related structures from lower levels; that is, by *avoiding* the devices, and developing instead programming expertise.

Again, we can easily understand this for other types of knowledge, in areas in which we have had the time to become proficient. We all use *language*, for example, starting with low-level elements – with words. We also know that, when expressing wishes or ideas, we do more than just build linguistic structures; what we do is combine our linguistic knowledge with other types of knowledge. Hence, if we have difficulty expressing ourselves, we know that language devices would not solve the problem.

Imagine now a society where an elite suggested that we use language by starting with higher-level elements – with ready-made sentences and ideas. Thus, instead of *creating* sentences and ideas, we would have to express ourselves by combining the ready-made ones produced by certain devices; and instead of increasing our knowledge and our linguistic capability, we would learn only how to operate these devices. Even without recalling the mechanistic fallacies, or the difference between simple and complex structures, we sense intuitively that this could not work, that we would lose something. We sense that, even if we could still create an infinity of ideas, it would be impossible to create the same variety of ideas as we do now, starting with words. And we realize that, by restricting the use of language, the elite would impoverish all knowledge and all aspects of our life.

We accept in the domain of software, thus, mechanistic concepts and theories that we would immediately dismiss in the case of language. We accept them because we fail to see that they serve the same purpose: to restrict our knowledge, our values, and our expectations. (We will return to this subject in chapters 5 and 6.)

## 4

The mechanistic theories of mind, we saw, claim that we can account for all the alternatives in human knowledge, behaviour, language, customs, and so forth; and they conclude that no creativity is involved in mental acts, that what we perceive as creativity is merely the selection and combination of bits of knowledge leading to a particular alternative. In the mechanistic *software* theories, the counterpart of this belief is the belief that we can account for all

possible alternatives in software-related phenomena. If we can account for all the alternatives, the conclusion must be that we can account for all possible applications, and also for the processes that take place in the mind when creating applications. So what we perceive as creativity and originality in the work of an experienced programmer is only an illusion: what the programmer really does in his mind at every stage of software development is akin to selecting and combining bits of programming knowledge in order to generate one of the applications.

More accurately, what the software theorists say is that programmers *could* create applications in this manner, so it is wrong to depend on such inexact resources as personal knowledge. Professional programming entails nothing but rules, standards, principles, and methods. Only old-fashioned practitioners rely on unscientific notions like skill, experience, or intuition. In other words, the theorists say, programming phenomena are deterministic: practically all the work involved in developing a particular application, or in modifying it later, can be specified precisely and completely from a knowledge of the requirements – just as the steps needed to build a car or an appliance can be specified in advance from a knowledge of their physical characteristics.

So, if we can account for all software applications that human minds can create – if there is nothing indeterministic, or unpredictable, in what programmers do – we should be able to replace programming knowledge with devices. The devices known as development environments, for example, materialize this idea by providing high-level starting elements. They simplify programming by minimizing the number of levels between the starting elements and the final application. Programming expertise, clearly, is becoming redundant, since even novices can now create applications. It is also interesting to note the trend in these devices to reduce the programmer's involvement to a series of selections, and selections within selections. The process of programming – the definition of files and variables, the specification of operations and conditions – has given way to a process of *selections*: instead of a blank screen where we can enter our definitions and specifications freely, we find colourful screens replete with menus, lists, icons, buttons, and the like. This structure of selections within selections attempts to emulate, obviously, the hierarchical structure believed to constitute the mental part of programming: the selection and combination of bits of knowledge leading to one of the alternatives, to a particular application.

The devices, thus, are an embodiment of the software theories. And, since the software theories are the counterpart of the traditional mechanistic theories of mind, we are witnessing the actual implementation of the idea that human creativity is an illusion, that the mind works simply by selecting and combining bits of knowledge within a predetermined range of alternatives. The

substitutes for human intelligence – the models of mind that the mechanists were proposing all along in psychology, sociology, and linguistics – have finally found their practical expression in the world of programming. While the traditional mechanists were content with speculations and theories, the software mechanists are actually replacing human intelligence with devices.

❖

We have already discussed the consequences of mind mechanism: by claiming that all human acts can be explained – by claiming, therefore, that human beings are incapable of truly creative, or unpredictable, acts – the traditional mechanistic theories can lead to a society where freedom, expertise, and creativity are redefined to mean the selection of acts from the range of alternatives sanctioned by an elite. In the world of programming, clearly, the shift has *already* occurred: the software theories assume that programmers are capable of nothing more creative than using programming substitutes, and this has given rise to the belief that it is these substitutes that are important, not the individual minds. This, in turn, has led to the belief that we must depend on the companies which provide the substitutes, and that the only way to improve our programming practices is by constantly adopting new versions of the substitutes.

Both the traditional theories and the software theories, thus, lead to the belief that human intelligence can be replaced with deterministic models. But, whereas traditional mechanism has had so far no serious consequences outside academia, *software* mechanism is causing great harm – by distorting our conception of knowledge and skills. Like the traditional ones, the theories behind the programming substitutes have been repeatedly refuted. Yet, while the theories themselves are failing, our belief in substitutes has become a self-fulfilling idea: Because we assume that programmers cannot advance beyond their current level, we encourage them to depend on substitutes. Consequently, no matter how many years of experience they have, their programming skills remain at the same low level – the level needed to use substitutes. Given this state of affairs, the adoption of the next substitute always appears to be the only way to improve their performance. So they waste their time assimilating yet another theory, or methodology, or language, instead of simply programming and improving their skills. And so the fallacy feeds on itself.

Human beings, as a matter of fact, *can* attain higher skill levels. It is only because programmers are forced to stay at novice levels that their capabilities do not exceed those of the substitutes. Recall the process of skill acquisition we studied in "Tacit Knowledge" in chapter 2: the skills and knowledge of a novice programmer constitute *simple* structures, just like the structures on which the

mechanistic software theories are based; only when attaining expertise can the programmer's mind process the *complex* structures that make up software phenomena. Expert programmers outperform the substitutes because their minds can cope with whole software phenomena, while the substitutes can only cope with isolated aspects of them.

So it is the dependence on programming substitutes that is preventing the emergence of a true programming profession. Application programming has been so degraded that it now entails nothing more difficult than the performance of acts which almost anyone can master in a year or two. We are underrating the potential of our minds, and we have forgotten that there exist levels of knowledge and skills higher than those reached after a year or two of practice. In effect, we are degrading our conception of human intelligence to the level attainable by the mechanistic substitutes for intelligence.

## 5

We saw how the mechanistic ideology has affected the programming profession. The most serious consequence of software mechanism, however, is not the destruction of *programming* knowledge, but the destruction of the *other* kinds of knowledge – knowledge that took centuries to develop. The belief that intelligence can be replaced with devices is already spreading into other domains, especially in our business-related pursuits.

In one activity after another, we see the claim that it is possible to account for all the alternatives in that activity, discover the starting elements that give rise to these alternatives, and then incorporate the elements in a software device. In all activities, we are told, what the mind really does is combine some elementary bits of knowledge into knowledge structures, one level at a time, just as we build cars and appliances. By providing directly some high-level starting elements – prefabricated knowledge subassemblies, as it were – the devices eliminate the need for each one of us to develop in our minds the low-level elements and the combinations leading to the high levels. The lengthy and arduous period of learning and practice is a thing of the past: all we need to know now is how to operate software devices. In one occupation after another, we are told that it is unnecessary, even wrong, to rely on personal experience. Anyone can perform the same tasks as the most experienced worker simply by selecting and combining the ready-made, high-level elements available through these devices.

The devices known as office productivity systems, for example, which address office workers, are the counterpart of the development systems used by programmers. Instead of claiming that we can account for all the alternatives

in programming activities, the software companies claim now that we can account for all the alternatives in *business-related* activities. Instead of software concepts that programmers can use as substitutes for programming expertise, the office systems promise software concepts that replace the skills of office workers.

And here too we see the trend to provide these concepts in the form of selections, and selections within selections, rather than permitting the user to combine freely some low-level elements. Here too, therefore, the system attempts to emulate what are believed to be the mental acts of an experienced person: selecting and combining hierarchically, one level at a time, the bits of knowledge leading to one of the alternatives.

As in the case of programming systems, the fact that an office system can generate an infinity of alternatives in business-related matters is mistaken as evidence that these devices can generate *all* possible alternatives. But this infinity is only a fraction of the infinity that an experienced worker can implement by starting from low levels, through the complex knowledge structures developed in his mind. Software devices cannot replace business-related knowledge any more than they can programming knowledge.

The fallacy, again, lies in providing starting elements that are not atomic and independent. We are lured by these devices only because we forget that the knowledge required to perform a difficult task constitutes, not an isolated structure, but a system of interacting structures: it includes many knowledge structures besides the neat hierarchical structure of selections that can be embodied in a software device. The different values of the top element – values representing the various acts performed by an experienced mind – are determined largely by the *interactions* between structures. The most important interactions occur at levels lower than the starting elements provided by devices, because it is low-level elements like variables and individual operations that are shared by the software structures, business structures, and knowledge structures which together make up the phenomena of office work. These interactions are lost when replacing minds with devices, and this is why a device can display only *some* of the alternatives displayed by a mind – those alternatives that can be represented with isolated simple structures.

To conclude, I want to stress again the link between the mechanistic theories of mind and the delusion of software devices. If we believe that it is possible to account for all the alternatives displayed by human beings in their mental acts, we will necessarily conclude that it is possible to describe human intelligence as a function of some starting mental elements; that the creativity of human

minds is an illusion; and that everything the mind does can be explained as we explain the working of a machine. It should be possible to represent a person's mind, therefore, with a simple structure where the elements and levels correspond to the knowledge developed by the mind, and the values of the top element correspond to the various acts performed by that person.

What is left is to implement this structure by means of a software device. Then, any person operating the device will be able to perform the same acts as the person whose mind the device is emulating. All that human beings do, in reality, is operate devices. So, why depend on a device like the mind, which is inefficient and unreliable and, moreover, can only develop knowledge structures through painstaking learning and practice, when we can purchase modern software devices that already contain these structures?

In the domain of programming, we have already replaced minds with devices. Now, as our reliance on computers is growing, the software elites are degrading all activities where human knowledge and skills play a part, in the same way they have degraded the activity of programming. They are modifying our conception of knowledge and skills to mean simply a dependence on software devices. They are instilling in us the belief that our intelligence, our work, our initiative, our experience, can be reduced to the process of selecting and combining operations within the range of alternatives provided by these devices. They are shifting our definition of expertise, creativity, and responsibility from their traditional meaning – to do a good job, to solve an important problem, to make a real contribution – to merely knowing how to use the latest software devices.