SOFTWARE AND MIND

The Mechanistic Myth and Its Consequences

Andrei Sorin
The mechanistic myth is the belief that everything can be described as a neat hierarchical structure of things within things. And few of us realize that our entire culture is based on this fallacy. While the world consists of complex, interacting structures, we prefer to treat every phenomenon as a simple, isolated structure.

Through our software pursuits, the mechanistic myth has spread beyond its academic origins, and is now affecting every aspect of human existence. In just one generation, it has expanded from worthless theories of mind and society (behaviourism, structuralism, universal grammar, etc.) to worthless concepts in the field of programming (structured programming, object-oriented programming, the relational database model, etc.) to worthless software-related activities that we all have to perform.

What is worse, our mechanistic beliefs have permitted powerful software elites to arise. While appearing to help us enjoy the benefits of software, the elites are in fact preventing us from creating and using software effectively. By invoking mechanistic software principles, they are fostering ignorance in software-related matters and inducing dependence on their systems. Increasingly, in one occupation after another, all we need to know is how to operate some software systems that are based on mechanistic principles. But our minds are capable of non-mechanistic knowledge. So, when the elites force us to depend on their software, they exploit us in two ways: by preventing us from creating better, non-mechanistic software; and by preventing us from using the superior, non-mechanistic capabilities of our minds.

The ultimate consequence of our mechanistic culture, then, is the degradation of minds. If we restrict ourselves to mechanistic performance, our non-mechanistic capabilities remain undeveloped. The world is becoming more and more complex, yet we see only its simple, mechanistic aspects. So we cope perhaps with the mechanistic problems, but the complex, non-mechanistic ones remain unsolved, and may eventually destroy us.
SOFTWARE AND MIND

The Mechanistic Myth and Its Consequences

Andrei Sorin

ANDSOR BOOKS
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  xiii

**Introduction**  Belief and Software  1
- Modern Myths  2
- The Mechanistic Myth  8
- The Software Myth  26
- Anthropology and Software  42
  - Software Magic  42
  - Software Power  57

**Chapter 1**  Mechanism and Mechanistic Delusions  68
- The Mechanistic Philosophy  68
- Reductionism and Atomism  73
- Simple Structures  92
- Complex Structures  98
- Abstraction and Reification  113
- Scientism  127

**Chapter 2**  The Mind  142
- Mind Mechanism  143
- Models of Mind  147
Chapter 3  **Pseudoscience**  202
The Problem of Pseudoscience  203
Popper's Principles of Demarcation  208
The New Pseudosciences  233
  The Mechanistic Roots  233
  Behaviourism  235
  Structuralism  242
  Universal Grammar  251
Consequences  273
  Academic Corruption  273
  The Traditional Theories  277
  The Software Theories  286

Chapter 4  **Language and Software**  298
The Common Fallacies  299
The Search for the Perfect Language  306
Wittgenstein and Software  328
Software Structures  347

Chapter 5  **Language as Weapon**  368
Mechanistic Communication  368
The Practice of Deceit  371
The Slogan “Technology”  385
Orwell's Newspeak  398

Chapter 6  **Software as Weapon**  408
A New Form of Domination  409
  The Risks of Software Dependence  409
  The Prevention of Expertise  413
  The Lure of Software Expedients  421
Software Charlatanism  440
  The Delusion of High Levels  440
  The Delusion of Methodologies  470
The Spread of Software Mechanism  483

Chapter 7  **Software Engineering**  492
Introduction  492
The Fallacy of Software Engineering  494
Software Engineering as Pseudoscience  508
Structured Programming 515
  The Theory 517
  The Promise 529
  The Contradictions 537
  The First Delusion 550
  The Second Delusion 552
  The Third Delusion 562
  The Fourth Delusion 580
  The GOTO Delusion 600
  The Legacy 625
Object-Oriented Programming 628
  The Quest for Higher Levels 628
  The Promise 630
  The Theory 636
  The Contradictions 640
  The First Delusion 651
  The Second Delusion 653
  The Third Delusion 655
  The Fourth Delusion 657
  The Fifth Delusion 662
  The Final Degradation 669
The Relational Database Model 676
  The Promise 677
  The Basic File Operations 686
  The Lost Integration 701
  The Theory 707
  The Contradictions 721
  The First Delusion 728
  The Second Delusion 742
  The Third Delusion 783
  The Verdict 815

Chapter 8 From Mechanism to Totalitarianism 818
  The End of Responsibility 818
    Software Irresponsibility 818
    Determinism versus Responsibility 823
  Totalitarian Democracy 843
    The Totalitarian Elites 843
    Talmon’s Model of Totalitarianism 848
    Orwell’s Model of Totalitarianism 858
    Software Totalitarianism 866

Index 877
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. This process started three centuries ago, is increasingly corrupting us, and may well destroy us in the future. The book discusses all stages of this degradation.

The book's epigraph, about Newspeak, will become clear when we discuss the similarity of language and software (see, for example, pp. 411–413).

Throughout the book, the software-related arguments are also supported with ideas from other disciplines – from philosophy, 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. Often, the connection between the traditional issues and the software issues is immediately apparent; but sometimes its full extent can be appreciated only in the following sections or chapters. If tempted to skip these discussions, remember that our software delusions can be recognized only when investigating the software practices from this broader perspective.

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. 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.

There is some repetitiveness in the book, deliberately introduced in order to make the individual chapters, and even the individual sections, reasonably independent. Thus, while the book is intended to be read from the beginning, you can select almost any portion and still follow the discussion. An additional benefit of the repetitions is that they help to explain the more complex issues, by presenting the same ideas from different perspectives or in different contexts.

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 usually 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); and the plural, “elites,” is used when referring to several entities, or groups of entities, within such a body. Thus, although both forms refer to the same entities, the singular is employed when it is important to stress the existence of the whole body, and the plural when it is the existence of the individual entities that must be stressed. The plural is also employed, occasionally, in its normal sense—a group of several different bodies. Again, the meaning is clear from the context.

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
Thank You for previewing this eBook

You can read the full version of this eBook in different formats:

- HTML (Free / Available to everyone)
- PDF / TXT (Available to V.I.P. members. Free Standard members can access up to 5 PDF/TXT eBooks per month each month)
- Epub & Mobipocket (Exclusive to V.I.P. members)

To download this full book, simply select the format you desire below

Free-Ebooks.net