Report on "Software and Mind" by A. Sorin
Reviewed by G. Snelting
April 2019

Summary. In Eco's famous book "The Name of the Rose", inquisition victim Salvatore claims "If one's true enemies are too powerful, one must search for weaker enemies". This is exactly what Sorin is doing in his 900 page book. His ultimate goal is to defend human freedom and creativity. Unaware of today's true enemies of freedom & creativity, he constructs a weaker enemy: the mechanistic world view, as found in software engineering, mathematics, physics, behaviourism, structuralism, and Chomsky's generative grammar. Sorin fights this enemy on 900 pages, but his enemy is like Don Quichottes windmill.

I can understand him, though. Sorin was a software developer in the 90's, and is traumatized by software techniques such as "Jackson structured programming" which, together with questionable CASE tools, back then were agressively hyped and forced upon application developers. Indeed, the "million monkey approach" was quite popular at IBM and elsewhere. Naturally, Sorin didn't want to be a monkey. Later, he discovered that similar mechanistic, hierarchical schemes as in 90's software can be found everywhere in modern science. But Sorin claims that mechanistic world views and techniques can deliver only hierarchical "entities within entities" models, and cannot handle complex interactions between entities. Mechanism, says Sorin, thus provides a modern, popular science myth, but is in fact designed to dehumanize people, transform them into dumb assembly line workers, and exercise totalitarian control.

2/3 of the book deal with general aspects of mechanistic views. The book has well-written, interesting sections on modern myths, Popper, behaviourism (I liked the behaviourism chapter a lot), totalitarism and more. But it is very redundant. The book's structure reminds me of the TVangelist who explained: "First I tell them what I will tell them; then I tell them what I have to tell them, then I tell them what I've told them." And the windmill fight by and by transforms into a true conspiracy theory, such as in the claim (p. 196) "prevention of true expertise is the final goal of software elites". Worse, Sorin, obsessed with his "entity within entity" furor, is unaware of the deep non-hierarchical results in e.g. general relativity, quantum entanglement or modern software technology.

I will provide below detailed comments on specific claims in the book, in particular claims about software engineering and mathematics. Here in my summary, I present just one example why Sorin's fight in fact *favours* modern totalitarian positions. Sorin claims that Chomsky's generative grammar - which Chomsky supposed to have a biological basis -- is an example of simplistic, mechanistic thinking, which limits creativity and freedom; and that "there is no innate language faculty" (p. 273). The latter argument has also been put forward by some contemporary linguists: they claim that language structures do not have a common, biologically based fundament, but it's all learning (like car driving). And it is certainly true that Chomsky's original proposal was simplistic. Chomsky later held a much weaker position, however Sorin does not tolerate "improvements" of mechanistic theories (p. 284ff).

Sorin -- accidentially -- thus takes position in the nature vs nurture debate. In fact he supports those who condemn behaviour-influencing structures developed in biological evolution as "biologistic". These modern "constructivists" create a new totalitarian world view: "it's all cultural, and those who deny this are biologistical". Today, the irrational claims by both left and right wing radicals, namely that there is no objective truth, and everything is just a "construct" (or "fake news") is much more dangerous for freedom than Chomsky or any mechanistic philosophy ever could be. Sorin however is obsessed with his weak enemy from the 90's, "dolls within dolls", and misses todays epistemological challenges. Like his ancestor Salvatore, he will one day wake up and face his true enemies.

The Software Chapter. The 300-page software chapter was the starting point of the book, and one might think of publishing it separately (with perhaps one introductory chapter). But the chapter is not "provocative" by today's standards, it is just out of date. It was begun in the 90's, and as mentioned, the author was victim of the back-then hype of some back-then software methods. Let us consider his explanations in some detail.

Concerning structured programming, Sorin rattles pages on pages about presumed inconsistencies, namely that some authors recommend 6 basic constructs, others only 2 etc. He did not really understand that the essence of structured programming is „one entry, one exit" per construct. The actual number of constructs is a matter of taste (or hype). But it is essential that all flowgraphs remain *reducible*. GOTOs can be tolerated as long as reducible flowgraphs result. Sorin has not heard about reducible flowgraphs, so his ranting about the Böhm-Jacopini theorem is just ridiculous. Even when I was a student (80's), the Böhm-Jacobini theorem was clearly presented as "theoretically interesting, practically useless". This was and is the academic position, regardless of all the hype Sorin has been suffering from. Looking back, structured programming was an important step away from the horrible spaghetti coding of the time. Hence Dijkstra is another of Sorin's "weak enemies".

I would also like to note again that Sorin's notorious denial of "improvement" (or "evolution") of theories is not Popperian, but Un-Popperian. Consider Einsteins "falsification" of Newton: In fact Newton's gravity was not abandoned by Einstein, in fact general relativity includes Newton's theory as a limit case (small masses & speeds). Even today, Newton's theory is of great importance in practical applications, and Einstein cannot be understood without Newton. In a strict ("Sorin") sense however Einstein falsified Newton. But such a formal view is disconnected from historical and practical reality, and that is why Popper argued for "evolution of knowledge". Likewise, today's extremely successful software verification techniques such as model checking cannot be understood without structured programming etc, and the corresponding (historical) fight against software entropy. Evolution of knowledge is not pseudo-scientific, but very scientific.

Concerning relational databases, it is true that for some (or many) applications the 3rd normal form is too restricted (and new concepts have been developed since the 90's). But Sorin fails to see that the main task of databases is not to store & access data, but to keep them consistent in a distributed multi-user environment of perhaps thousands of simultaneous accesses per second. Thus relational databases were a huge advantage over Cobol files not due to the 3rd normal form Sorin despises so much, but due to the „ACID (atomicity, consistency, isolation, durability)" properties.

Concerning object-oriented programming, again I can understand Soron's disgust about all the 90's hype. Indeed, inheritance hierarchies are often too unexpressive (this problem became known in 1992 as the "tyranny of the dominant decomposition"). Today smart answers are available such as Traits and Mixins in Scala, or Virtual Classes. Sorin has not heard about these, but might find them appealing for competent programmers. And even for original OO, dynamic dispatch (perhaps in combination with multiple inheritance as in C++) was a big step. Dynamic dispatch is definitely better than explicit case-switching, and modern books explain why. But it seems Sorin does not know about dynamic dispatch, he only talks about single inheritance - another "weak enemy".

Some Detailed Remarks.

- PLUS: well-written history of mechanics & corresponding epistemology
- PLUS: criticism of meachanistic psychology, sociology (behaviourism)
- PLUS: it is true that often expertise is replaced by software tools, destroying expertise & creativity. Another example: music composition / production!
- PLUS: praising Popper
- but every succesful use case is a failed falsification, so Sorins completely negative attitude towards positive examples is distorted
- often theories are probabilistic, and then a single counterexample cannot falsify a theory. For example a new medical drug might work for 99 patients, but not work for 1. Sorin would call the latter case a falsification. By statistical standards, it is a positive use case.
- It is unfair and distorted to describe the evolution of a theory as a trick to integrate or neutralize falsifications. Initial theories are often too limited, but still provide important breakthroughs. Thus theory evolution is not escaping refutation! Popper himself clearly identified refutation as a tool to stimulate evolution of knowledge, where evolution of knowledge is NOT "escaping refutation".
- Sorin only knows trees and "entities within entities". Has he ever heard about graphs? Modern science is full of graphs.
- Sorin claims that "mechanistic thinking cannot handle interactions between entities" and that "physic is mechanistic thinking". Has he ever heard about quantum mechanics, entanglement, and the EPR paradox? It seems not, which explains his ridiculous simplifications.
- Sorin (like Wittgenstein in his Tractatus!) thinks that mathematics is tautologic, because every theorem is -- presumably -- already born in the axioms. But this view is very distorted. It is true that mathematical proofs are hierarchically structured, but that is just the surface. Think e.g. of the deep and beautiful properties of holomorphic functions in complex analysis. These took centuries to find, and still there are questions open for more than 150 years (e.g. Riemann conjecture). Advanced mathematics is full of deep, nonhierarchical properties. The deep insights in many theorems can only be obtained by creative experts. Thus mathematics is not mechanistic. And since complex analysis is so important for eg quantum physics, many scientists believe that complex numbers are much more than just a formal construction or an „approximation", namely that they are part of reality.
- Chomsky: it's true that phrase structure grammars are too restricted, and that Chomsky ignored semantic aspects. But today's successful ML based machine translators still use simple grammars, and replace semantics simply by statistical coincidence (aka deep learning)! That's mechanistic, and very successful!
- Wirth + Dijkstra, who promoted structured programming, were not "academic bureaucrats, who cannot make a real contribution" (p. 274). Both made eminent contributions to e.g. algorithmics and compilers. Both received a Turing award for this.