Alex Groce (agroce@gmail.com), Northern Arizona University

Herbert A. Simon's *The Sciences of the Artificial (Third Edition)* is a possibly somewhat neglected classic text of computer science. It is not unread, but it is not as widely read as other "foundational" texts. This is a shame, for while the book (and Simon) are perhaps most associated with artificial intelligence research, *The Sciences of the Artificial* provides a new way to think about software engineering, and includes not only fundamental claims about human beings (and computers) that could shape our approach to software, but concrete proposals about how to teach software engineering as partly based on a science of design.

The eight chapters of *The Sciences of the Artificial* have common themes, and should ideally be read in order, but also work fairly well as isolated readings. This is not surprising: the chapters are re-workings (substantial ones, by the present, third edition) of lectures Simon gave, from coast to coast and over a period of more than a decade, and papers he wrote over the same long period. Simon, in the preface, calls the resulting essays "fugues," and the metaphor works well as a way to see the whole book: there are interlocking themes in these essays, repeated and re-shaped, and while listening in order is ideal, walking into the concert hall at any point it should still be possible to re-create key elements of the initial exposition.

Simon's initial exposition is an attempt to define and propose the "sciences of the artificial": that is, sciences in which, in addition to natural law, purpose or *goal* is central. Simon first rejects the pejorative connotations of the word "artificial." For Simon, the artificial is not superficial, false, or dishonest; the hallmark of the artificial is simply that artificial things are synthesized by human beings, "characterized in terms of functions, goals, adaptation," and discussed "in terms of imperatives as well as descriptives." That is to say, the artificial is engineered for to serve some purpose. The natural sciences are, essentially, descriptive: there is no value in saying that the law of gravity fails to optimally make things fall: it is what it is. But the sciences of the artificial concern how an entity, in conjunction with its environment, achieves its purposes. Where teleology is usually a mistake in natural sciences, it is unavoidable in the sciences of the artificial. One key upshot of this idea is that the artificial can often largely be considered in terms of its outside environment. This idea that "The behavior takes on the shape of the task environment," is, of course, central to the modern concept of software development as well. While a library API has an internal existence, we can consider it primarily in terms of its interface to an environment in which it provides certain functionality. In fact, without the ability to consider software elements without regard to their internal construction, but only in terms of their interface to an environment, and goal-directed behavior in that environment, it would be impossible for humans to produce software systems with significant scale.

Having defined the sciences of the artificial, Simon turns, perhaps surprisingly for a modern computer scientist or software engineer, to his first example: economic rationality, and the modeling of human behavior. But, of course, an economy is an artificial entity ordered by the goals of its participants, and the question of how to understand human behavior is central to any study of the artificial: we are the producers of artificial things, at present. One of Simon's big

ideas (the one that he is most famous for in economics, and received the “Nobel” in economics for) is that humans do not, because they *cannot*, rationally optimize their economic behavior. Instead they seek to satisfy minimal constraints, in most cases: the computational burden of optimality is far too large. Finite, limited minds act in ways that significantly differ from abstract optimality. Simon takes us on a quick tour of economic and administrative theory, from operations research to the concept of how organizations operate without using an internal market to make decisions.

The third essay explores how we can understand and perhaps even quantify the limits on human rationality via experiment. Simon proposes that, since it is possible to grasp the limitations of a system without knowing its internal construction, we can also hypothesize about how a system (in this case, a human being) generally goes about solving problems. Simon’s most famous contribution to AI is the General Problem Solver system. The fourth chapter continues this theme, with the focus changed to memory, rather than problem-solving. These chapters are interesting, but may seem to be rather remote from the interests of most software engineers. On the contrary, in practice, the limits of human abilities are among the most important, but hard to perceive, inputs to the process of engineering (whether software or some other kind of engineering). For example, Simon points out that plans that require effective prediction of the future often fail, because such prediction is beyond our bounded rationality (and, sometimes, not even possible in principle, as the later section on chaos points out). Feedback is a better mechanism than prediction, in many cases, for adapting to changes. To think that such discussions have nothing to contribute to the debates about software engineering methodologies (and the underlying motivations for, e.g., agile approaches) is to dismiss substance in favor of ephemera (including some current precise “best practices”).

That said, it is with the fifth chapter, “The Science of Design: Creating the Artificial,” that the most relevant part of *The Sciences of the Artificial* for software engineering starts. Simon begins by discussing the shift in engineering education from a practical, trade-school approach that lacked rigor towards a natural-sciences based approach that lacked consideration of the sciences of the artificial. Simon does not oversimplify: it is a mistake to have a chemical engineer who does not know chemistry, or a software engineering who does not know computer science; but it is also a mistake to not consider the actual design of distillation columns or working programs, as *artifacts*: things with goals, and constraints, and where solutions are heuristic and satisficing, rather than calculated from natural law. Simon therefore proposes an engineering design curriculum covering at least:

“THE EVALUATION OF DESIGNS 1. Theory of evaluation: utility theory, statistical decision theory 2. Computational methods: a. Algorithms for choosing optimal alternatives such as linear programming computations, control theory, dynamic programming b. Algorithms and heuristics for choosing satisfactory alternatives 3. THE FORMAL LOGIC OF DESIGN: imperative and declarative logics THE SEARCH FOR ALTERNATIVES 4. Heuristic search: factorization and means-ends analysis 5. Allocation of resources for search 6. THEORY OF STRUCTURE AND DESIGN ORGANIZATION: hierarchic systems 7. REPRESENTATION OF DESIGN

PROBLEMS”

After this, the entire final half of the book revolves around design and complexity (which design is often essentially about managing) as central to a science of the artificial. Chapter 6 examines the social considerations of design, including a broad overview of the ethical dimension of software engineering (or any other engineering or planning endeavor). Chapter 7 presents a few alternative views of complexity (holism vs. reductionism, cybernetics, catastrophe theory, and chaos and complexity analyses), and Chapter 8 presents Simon’s own proposal for an architecture of complexity, the understanding of hierarchic systems.

This summary leaves out much of what *The Sciences of the Artificial* discusses that is of interest to a software engineer: the usefulness of thinking about biological systems and evolution in considering software systems; the great importance of how complexity is described (state vs. process descriptions: in other words, blueprint vs. recipe), the nature of administrative decision-making; the shifts in time-perspectives introduced by 20th century technologies; the idea of symbol systems as a central organizing concept in consider information-processors such as computers and humans; the value of learning from example; and so on and so on… and so on. Simon’s Renaissance-man breadth leads to the production of long lists.

In fact, the variety of topics makes *The Sciences of the Artificial* irritatingly difficult to sell to the would-be reader. Most great books are less diffuse than this; this is partly due to the nature of a collection of talks and essays, and partly because, while the concept of “the artificial” does provide an underlying rationale for all that is within, Simon’s was not a mind, I think, that simply moved from one unifying vision to a series of applications. That metaphor that comes to mind is Berlin’s famous fox vs. hedgehog distinction: the fox knows many things, and the hedgehog only knows one (big) thing. Is Simon a fox? The claim is tempting. This book does not summarize well, and Simon’s ideas in computer science, political science, economics, and psychology really are varied and complex, like the topics themselves. Simon is the only Turing laureate to arguably be *more famous for other things he did*. In an essay celebrating Simon centenntial, (https://manwithoutqualities.com/2016/07/08/herbert-simon-a-hedgehog-and-a-fox/) Frantz and Marsh argue that Simon is *both* a fox and a hedgehog: quoting Simon, he was a “monomaniac about decision-making.” The central problem of the software engineer, or any engineer, is the problem of decision-making in the face of uncertainty. If Simon’s thoughts are complex and difficult to summarize, it is perhaps because decision-making reflects the complexity of the world in which decisions must be made, another of Simon’s central insights.