Some Observations About the Nature of Computer Science

Juris Hartmanis

Computer Science Department
Cornell University
Ithaca, New York

Abstract. This paper discusses the nature of computer science as a science by analyzing computer science and comparing or contrasting it with other sciences. In particular, we compare and contrast various aspects of computer science with physics, astronomy, and mathematics.
Our analysis of computer science and comparison with other sciences is primarily in terms of:

(a) the roles played by experiments and theory,
(b) how research paradigms and problem areas are determined and change,
(c) the relation and interaction of the science and engineering aspects,
(d) comparing historical developments.

From this study and comparisons we conclude that computer science differs from the known sciences so deeply that it has to be viewed as a new species among the sciences. This view is justified by observing that theory and experiments in computer science play a different role and do not follow the classic pattern in physical sciences. The change of research paradigms in computer science is often technology driven and demos can play the role of experiments. Furthermore, the science and engineering aspects are deeply interwoven in computer science, where the distance from concepts to practical implementations is far shorter than in other disciplines.

Introduction

We all are witnessing an information revolution which is profoundly changing all aspects of our societies. This revolution is driven by unprecedented advances in computing power and communications capacity, and it is guided by computer science and engineering. It is quite likely that the changes caused by this revolution will rival those of the industrial revolution. Computing is already embedded in the fabric of our societies, but this is just the beginning of a complete penetration of all our activities as all information is digitalized and computing becomes ubiquitous in our physical and intellectual environment.

Personally, I am deeply convinced that computer science is a very important new science and that it is important that its nature as a science is understood better not only by the practitioners, but also by other scientists and the people who make decisions about science policy and support. It is important to understand where computer science fits in the constellation of sciences, how it relates, compares and differs from other sciences. It is particularly important to understand its research paradigms and methodology. One can cite many instances where lack of understanding of the nature of computer science has led to wrong expectations, to lack of appreciation of the achievements (since it does not behave like some other science) and incorrect policy decisions.

There is no doubt that computer science and engineering has played a very important role in guiding and facilitating the information revolution. At the same time, computer science is not very well understood as a science by other scientists and even by people close to computer science themselves.

This short paper will give only a few partial answers to the questions about the nature of computer science. A much more extensive comparison with paradigms, methodology and historical development of other sciences has to be made, and a much richer set of examples and case studies is needed for a proper understanding. At the same time, it is hoped that this paper will add to the insights about the nature of computer science and stimulate discussion and interest in further analysis. Particularly important is to collect more illustrations and case studies from various parts and periods of computer science.

In general, I believe that the historical development and the epistemological nature of computer science should be of considerable interest to historians and students of the philosophy of science. So far, there have been very few serious studies along these lines. I hope that will change soon, since this is indeed a rich new area of investigation of a very rapidly developing new species among the sciences.

The Other Sciences

To see the uniqueness of computer science and to justify our claim that it has to be viewed as a new species among the sciences, we discuss shortly the classic research paradigms and paradigm changes in other sciences. We take physics as our example of a very successful physical science with well-understood research methodology. In general, the interaction of theory and experiments is well understood and colorfully illustrated by the rich and impressive history of physics. At the same time, the history of physics and several other sciences reveals a larger structure in their evolution and how the research paradigms change. A penetrating discussion of this larger structure of scientific developments can be found in "The Structure of Scientific Revolutions" by Thomas S. Kuhn [Ku].

In a highly simplified form we can describe this view of the development of science as the practice of *normal science* which is broken by a series of successive revolutions, resulting in changed research paradigms. Each revolution is proceeded by an impasse or anomaly in which theory and observations clash irreconcilably.

The periods of normal science can be seen "as strenuous and devoted attempt to force nature into the conceptual boxes supplied by professional education". This is all done in terms of the current research paradigms, the "universally recognized scientific achievements that for a time provide model problems and solutions to a community of practitioners." The scientific revolutions force a change of the accepted research paradigms by introducing new theories, concepts and viewpoints, redefining what are accepted problems and solutions. Quite often, the new theories remove the troublesome anomalities but fail to fully explain all phenomena covered by the old theory. The old and new theories can exist side by side for some period, in some cases till the older generation of scientists leave the scene.

Indeed, the history of physics provides many examples for this view of scientific revolutions of various scales. In some of them a new experiment causes the change in research paradigms; in others the nagging anomalies of known experimental results which cannot be explained by the old theories force a paradigm change. To fix our ideas we will describe some developments in physics and contrast them with the situation in computer science.

In this short version of this paper we will illustrate the paradigm changes by a quick summary of some dramatic developments in physics during the early part of this century.

About the turn of the last century the atomic theory of matter was accepted and the electron and other elementary particles had been identified, but the nature of the atom itself was not at all clear. There was no explanation of the properties of various atoms nor for the light spectra emitted by them. Radioactivity was discovered in 1896, revealing a totally unexpected behavior of some atoms and, as we will see, providing later the probes to study the atom itself. The inability to properly explain (or derive a formula) for the black-body radiation energy distribution led Max Planck in 1900 to introduce the artifact of quantization (the radical idea that energy comes only in discrete quantities) to derive a formula consistent with physical evidence. Though Planck's new formula for the energy distribution of the black-body radiation agreed beautifully with experimental results, Planck himself for a while considered the quantization concept only as a mathematical artifact, not necessarily a natural phenomenon. The full impact of this revolutionary idea was better understood five years later when Einstein, in 1905, explained the photoelectric effect in terms of photons (light particles) whose energy was given in terms of their wavelength and Planck's constant. This explained why the velocity of electrons emitted from metals when light is shined on them depended on the color of the light and not the intensity of the light. It is interesting to recall that Einstein's Nobel Prize was awarded in 1921 for the explanation of the photoelectric effect and not for the far more fundamental work in Relativity Theory (1905 and 1915). Planck had received the Nobel Prize in 1919 for the work on the black body radiation.

Roughly at the same time, Ernest Rutherford, one of the best experimentalists ever, was exploring radioactivity, but more fortunately using it to explore matter. Rutherford and Geiger were using the alpha particles (helium atoms) emitted by radioactive materials to explore matter. They were shooting alpha particles at thin metal sheets and measuring the angles of deflection (from the straight path if there were no sheets), by observing scintillations on zinc sulfide sheets hit by alpha particles. During these experiments, Rutherford decided to test if any alpha particles are reflected back by a thin sheet of gold. Indeed, some alpha particles came straight back! Rutherford expressed his total surprise:

> It was quite the most incredible event that ever happened to me in my life. It was almost as incredible as if you fired a fifteen-inch shell at a piece of tissue paper and it came back and hit you.

What threw the alpha particles back, what could stop particles rushing at 10,000 miles per second? These startling experiments led Rutherford in 1911 to postulate and experimentally justify the "planetary" model of the atom: a very compact, positively charged nucleus containing most of the mass of the atom surrounded by orbiting electrons with lots of empty space between.

So far, we have seen how a dramatic experiment led to a new concept of matter. Next, theoretical considerations forced a refinement of this model which was inconsistent with theory. By electromagnetic theory, the orbiting electrons must radiate electromagnetic waves. The radiation would lose energy and eventually the electrons would crash into the nucleus, which clearly was not the case, since most matter was very stable. It was Neils Bohr who reconciled the new model with theory and also obtained new explanations for physical observations (to a considerably higher degree). Bohr, having grasped the real meaning of Planck's and Einstein's work, applied the quantization ideas to the new atomic model. The electrons orbit the nucleus in fixed orbits determined by the Planck-Einstein quantization. Light emission and absorption occurs only in fixed amounts of energy when electrons jump from one orbit to another. A course computation using Planck's constant h showed that this model roughly accounted for the observed hydrogen spectrum (of distinct lines). Subsequent work by Sommerfeld and others accounted for much of the fine structure of the hydrogen atom after further quantization of other aspects of the orbiting electrons.

It is interesting to observe that Rutherford received the Nobel Prize in Chemistry already in 1908 for his work on radioactivity, Neils Bohr received the Nobel Prize in Physics in 1922 for explaining the nature of atoms and molecules.

From this very sketchy account we can see the interaction of theory and experiments and the power of experiments leading to paradigm shifts. We also see very fundamental changes in the views of the physical world.

We will argue that research in computer science does not fit this classic pattern of the physical sciences. On the other hand, there should be no doubt that computer science, though not a physical science, is indeed a science with fundamental intellectual contributions and immense practical importance.

Theory, Experiments and Demos in Computer Science

We will not try to give a few-line definition of computer science since no such definition can capture the richness of this new and dynamic intellectual process, nor can this be done very well for any other science. The difference between this and other major sciences is that there is a much clearer public understanding of what physicists, chemists, and astronomers do than what computer science is all about (besides building computers and writing software which is hard to use).

At the same time, it is clear that the objects of study in computer science are information and the machines and systems which process and transmit information. From this alone, we can see that computer science is concerned with the abstract subject of information, which gains reality only when it has a physical representation, and the man-made devices which process the representations of information. The goal of computer science is to endow these information processing devices with as much intelligent behavior as possible.

One of the defining characteristics of computer science is the immense difference in scale of the phenomena computer science deals with. From the individual bits of programs and data in the computers to billions of operations per second to the highly complex operating systems and various languages in which the problems are described, the scale changes through many orders of magnitude. Donald Knuth [Kn] puts it nicely:

> CS&E is a field that attracts a different kind of thinker. I believe that one who is a natural computer scientist thinks algorithmically. Such people are especially good at dealing with situations where different rules apply in different cases; they are individuals who can rapidly change levels of abstraction, simultaneously seeing things "in the large" and "in the small."

The computer scientist has to create many levels of abstractions to deal with these problems. He has to create intellectual tools to conceive, design, control, program, and reason about the most complicated of human creations. Furthermore, this has to be done with unprecedented precision. The underlying hardware which executes the computations are univeral machines and therefore they are chaotic systems: the slightest change in their instructions can result in arbitrarily large differences in the results. This, as we well know, is an inherent property of universal computing devices (and theory makes clear that giving up universality imposes a very high price). Thus computer scientists are blessed with a universal device which can be instructed to perform any computation but which is therefore chaotic and must be controlled with unprecedented precision. This is achieved by the successive layers of implemented abstraction wrapped around the chaotic universal machines which help to bridge the many orders of magnitude in the scale of things.

Clearly, computer science is not a physical science; still, very often it is assumed that it will show strong similarities to physical sciences and may have similar research paradigms in regard to theory and experiments. The failure of computer science to conform to the expected paradigms of physical sciences is often interpreted as immaturity of computer science or that it is not developing as it should be.

We will argue that that is not the case and that theory and experiments in computer science play a considerably different role. In particular, from the short discussion of the paradigm changes in physics and the corresponding interaction between experiments and theory, we can see marked differences. It is clear that Planck tried to explain a nagging anomaly in the black-body radiation which did not agree with theory. We have no strong analogies of this kind of disagreement of theory and observations in computer science. Nor do we have anything coming even near the very dramatic experiment of the "reflected" alpha particles which led Rutherford to the new planetary model of the atom, which in turn violated classical laws of electrodynamics and had to be salvaged by Bohr's Theory, which inaugurated a new area in atomic physics and eventually led to quantum mechanics and a complete change in how we view (time), matter, determinism, and observations.

I submit to you that there are no analogies to this type of interaction between experiments and theory in computer science.

If one uses the self-referential definition that computer science is what computer scientists do, then the inspection of what computer scientists have done should let us deduce the characteristics of this science. Even a very quick look at the theoretical and experimental work in computer science shows us that theory and experiments play a different role than in physics. The difference in research paradigms between physics and computer science can be seen very clearly by comparing the work which led to Nobel Prizes in physics with the work recognized by the Turing Award (we will not pursue this analysis in this paper).

We mention just a few theoretical topics from computer science to fix our ideas.

For example, the design and analysis of algorithms is a central theme in theoretical computer science. Methods are developed for their design, measures defined for various computational resources, trade-offs between different resources are explored, and upper and lower resource bounds are proved for the solutions of various problems. Similarly, theory creates methodologies, logics and various semantic models to help design programs, to reason about programs, to prove their correctness, and to guide the design of new programming languages. Theories develop models, measures and methods to explore and optimize VLSI designs, to try to conceptualize techniques for computer and communications systems, etc., etc.

Thinking about the above-mentioned (and other) theoretical work in computer science, one is led to the very clear conclusion that theories do not compete with each other for which better explains the fundamental nature of information. Nor are new theories developed to reconcile theory with experimental results which reveal unexplained anomalies or new, unexpected phenomena. In computer science there is no history of single dramatic, critical experiments which decide between the validity of various theories, as there is in physical sciences.

The basic, underlying mathematical model of digital computing is not seriously challenged by theory or experiments. The ultimate limits of effective computing, imposed by the theory of computing, are well understood and accepted. There is a strong effort to define and prove the feasible limits of computation, but even here the basic model of computation is not questioned. The key effort is to prove that certain computations cannot be done in given resource bounds, well illustrated by the $P = NP$ question.

The results of theory are judged by the insights they reveal about the mathematical nature of various models of computing and/or by their utility to the practice of computing and their ease of applicability. Do the models conceptualize and capture the aspects computer scientists are interested in, do they yield insights in design problems, do they aid reasoning and communication about relevant problems? In the design and analysis of algorithms, which is a central theme in theoretical computer science, the measures of performance are well defined, and results can be compared quite easily in some of these measures (which may or may not fully reflect their performance on typical practical problems). Experiments with algorithms are used to test implementations and compare their "practical" performance on the subsets of problems deemed important.

Similarly, an inspection of the experimental work and systems building in computer science reveals a different pattern than in physical sciences. Such work deals with performance measurements, evaluation of design methodologies, testing of new architectures, and above all, testing feasibility by building systems to do what has never been done before.

Systems building, hardware and software, is the defining characteristic of applied work or experimental work in computer science (though experimental is not meant in the old sense). This has the consequence that computer science advances are often demonstrated and documented by a dramatic demonstration rather than a dramatic experiment as in physical sciences. It is the role of the *demo* to show the possibility or feasibility to do what was thought to be impossible or not feasible. It is often that the (ideas and concepts tested in the) dramatic demos influence the research agenda in computer science.

This is reflected in the battle cry of the young computer science faculties, "demo or die", which is starting to rival the older "publish or perish", which is still valid advice, but should be replaced by "publish in refereed journals or perish".

From the above observations we can see that theory and experiments in computer science are contributing to the design of algorithms and computing systems which execute them, that computer science is concentrating more on the *how* than the *what*, which is more the focal point of physical sciences. In general the *how* is associated with engineering, but computer science is not a subfield of engineering. Computer science is indeed an independent new science, but it is intertwined with engineering concerns and considerations. In many ways, the science and engineering aspects in computer science are much closer than in many other disciplines. To quote Fred Brooks [Br] about programming:

> The programmer, like the poet, works only slightly removed from pure thought-stuff. He builds his castles in the air, from air, creating by exertion of the imagination. Few media of creation are so flexible, so easy to polish and re-work, so readily capable of realizing grand conceptual structures. (. . . later, this very tractibility has its own problems.)
>
> Yet the program construct, unlike the poet's words, is real in the sense that it moves and works, producing visible outputs separate from the construct itself. It prints results, draws pictures, produces sounds, moves arms. The magic of myth and legend has come true in our time. One types the correct incantation on a keyboard, and a display screen comes to life, showing things that never were nor could be.

Webster's dictionary defines engineering as "the application of scientific principles to practical ends as the design, construction, and operation of efficient and economical structures, equipment and systems". By this definition, much of computer science activity can be viewed as engineering or at least the search for those scientific principles which can be applied "to practical ends, design, construction, . . ." But again, keeping in mind Brooks' quote and reflecting on the scope of computer science and engineering activities, we see that the engineering in our field has different characteristics than the more classical practice of engineering.

As observed earlier, computer science work is permeated by concepts of efficiency and search for optimality. The "how" motivation of computer science brings engineering concepts into the science, and we should take pride in this nearness of our science to applicability.

Somewhat facetiously, but with a grain of truth in it, we can say that computer science is the engineering of mathematics (or mathematical processes). In these terms we see very strongly that it is a new form of engineering.

I am deeply convinced that we should not try to draw a sharp line between computer science and engineering and that any attempt to separate them is counterproductive.

Research Paradigms in Computer Science

We have argued that research paradigms in computer science do not show the same pattern as in physics. Still, we can observe interesting paradigm changes in computer science. Even in the short history of computer science as an organized academic discipline, which can be dated from the early 1960's, we can detect periods of "normal science" (in various areas of computer science) interspersed with marked paradigm changes. These paradigm changes do not rival the number and magnitude of such revolutionary changes in physics, but they still are detectable and have played an important role in the development of computer science. As a matter of fact, we believe that the understanding of computer science development can be helped by a careful study of paradigm changes. Also the history of computer science could better be viewed from this perspective.

In the following we will review very shortly some of these changes. A much more thorough study is needed.

A fundamental paradigm change in mathematics started in 1936 with Turing's successful capture of the concept of computability. Mathematics has had a longstanding interest in numerical computations, but the search for the mathematical formulation of effective computability intensified only after Gödel's incompleteness results. Turing's work culminated this effort. Gödel viewed Turing's success as a "minor miracle" that effective computability could be so sharply and elegantly defined. The effect on mathematics was profound. From this concept emerged the beautiful area of recursive function theory, logic was changed completely, and Kolmogorov (Solomonoff and Chaitin) were led to the beautiful definition of (algorithmic) randomness, which is playing an increasingly important role in theoretical computer science [LP]. As a matter of fact, the computing paradigm and computing itself have started a profound transformation of mathematics which is still accelerating.

A mathematical imperialist could say, somewhat facetiously, that computer science is nothing but a paradigm change of (or in) mathematics.

The real interest in computer science started only after the development of the first digital computers, of which ENIAC was one of the most influential. ENIAC was an engineering tour de force and a technological demonstration (demo) of the feasibility of electronic digital computing. We can see here a very clear example of the forces that influence and change the research paradigms, technological advances and demos. We are separating these two causes since, as we will see later, technological advances can influence the research agenda by their continuous steady progress, without a sharply delineated technological demo, and many influential demos are not hardware or technology related. The essential observation is that technology and demos play an important role in setting the research agenda in computer science. At the same time, we observe that demos are results of hardware and/or software implementations and therefore have an engineering component, which demonstrates again how in computer science the science and engineering aspects are intertwined in a new way.

To see this we just have to recall that Turing was originally motivated by a need to understand deeper Gödel's incompleteness results and was led to the definition of the Turing machine. Only later did Turing, in connection with code breaking, consider practical construction of computing devices and then start raising fundamental questions about artificial intelligence. von Neumann got interested in electronic computing when he learned about the ENIAC project, and only after consulting on this project did he help abstract and separate the logical design of computers from their implementation and conceptualize the architecture of the internally-stored program machine (a concept basically due to Turing). Konrad Zuse's work also contained hardware development with work on a universal programming language and reflections on machine chess playing.

Finally, we can view Babbage's effort in computing (difference and analytic engines) as demos that failed and missed the possibility to initiate the emergence of computer science.

In short, we can see the three main forces on research agenda in computer science: technology, demos, and new conceptualizations.

In more recent developments, compiler construction and operating systems work provide nice examples of normal periods of science, conceptual changes and technology-driven changes. FORTRAN and ALGOL, their conception and implementations, mark successful demos and new conceptualizations, respectively. After the articulation of syntax-directed compiling in the 1960's, the work on formal languages, parsing and compiler construction entered a normal development with impressive maturing in compiler design and code optimization (also a diminishing interest as it matured). This area of research attracted renewed interest and gained vitality with the technology-driven transition to research on compilers for parallel machines.

The development of operating systems provides another interesting sequence of conceptual changes facilitated by invention of new concepts and technology-driven changes. One of the major paradigm shifts occurred with time-sharing and interactive computing which, after some early work (demos), was epitomized by Project MAC in the USA. The appearance of mini-computers provided another shift. The emergence of personal computers and work stations led to the integration of communication into operating systems and the general interest in distributed computing and fault tolerance, which are today active areas of research.

In theoretical computer science, work in automata theory and formal languages shows an extended period of normal science development. This was followed in the early 1960's by computational complexity considerations, and this work experienced a noticeable paradigm shift in 1971 (Cook and Karp) with the definition of P and NP and the study of complete problems and reduction between classes of problems. Though we do not yet know if $P \neq NP \neq PSPACE \neq EXTIME \neq NEXTIME \neq EXSPACE$, the proof that a problem is complete or hard for NP (or a higher class) is accepted as very strong evidence that the problem is intractable. One of the profound intellectual changes in all of computer science is the growing understanding of the limits of tractability.

An interesting competition can be observed currently between two views (philosophies) of processor architecture. Turing's work clearly showed the extensive interchangeability of hardware and software in computing. Still, where to draw this dividing line for various technologies is not clear. For a long time the CISC architecture dominated the commercial computer field. The rival RISC architecture was influenced by a deeper understanding of what should be in hardware and software in processor design for chips. Today, both architectures enjoy commercial success, and there is no single experiment which will decide which is the right one, though I suspect that most academic computer scientists would bet on the RISC model and even industry seems to be going in that direction.

Many recent and future changes in computer science research agenda have and will be driven by the ever increasing power (and price performance) of microprocessors and the immense fiber optics communication capabity. The merger of these two technologies and the digitalization of all information are opening vast new areas for computer science research with new challenges for conceptualizations, trend-setting demos, and methodologies for system design. The impact of these technologies on computer science and on our societies is not yet well understood, but we already can sense that the effect is going to be profound.

Conclusion

We have identified some characteristics of computer science and discussed how research paradigms change. Though we have argued that computer science is a new species among the known sciences, a haunting question remains about analogies with the development of physics. Towards the end of the last century, there was a conviction among scientists that the basic nature of the physical world was understood and that further elaboration of Newtonian mechanics, electromagnetic theory and thermodynamics would give a complete description of the natural phenomena. Max Planck was advised not to become a physicist for lack of challenges in this field. History can record this as one of the greatest misjudgments in science. Can we expect eventually some comparably fundamental changes in the paradigms of computer science, or have we already established the framework in which this science will develop?

Acknowledgments

The views expressed in this paper have been deeply influenced by the author's participation in the National Research Council study resulting in the report, Computing the Future: A Broader Agenda for Computer Science and Engineering [CF] and by discussions with colleagues at Cornell University. Particularly influential have been Robert Constable, Fred Schneider, and Richard Zippel. The importance of demos in computer science was emphasized by Constable and the importance of the immense differences in the scale of phenomena in computer science was eloquently explained by Zippel and compared to the still badly understood phenomena of turbulence in fluid dynamics, where the wide range of scales is contributing to the difficulty of the problem.

References

[Br] Frederick P. Brooks, Jr., *The Mythical Man-Month: Essays on Software Engineering.* Addison-Wesley Publishing Co., Reading, Massachusetts, 1975.

[CF] J. Hartmanis and H. Lin, editors. *Computing the Future: A Broader Agenda for Computer Science and Engineering.* National Academy Press, Washington, D.C., 1992.

[Kn] D. Knuth, personal communication, March 10, 1992 letter.

[Ku] Thomas S. Kuhn, *The Structure of Scientific Revolutions.* University of Chicago Press, Chicago, 1962.

[LP] M. Li and P. M. B. Vitanyi, *An Introduction to Kolmogorov Complexity and Its Applications.* Springer-Verlag, Heidelberg, Germany, 1993.

[Mo] Ruth Moore, *Niels Bohr, The Man, His Science, and the World They Changed.* Alfred A. Knopf, New York, 1966.

[Wi] David Wilson, *Rutherford, A Simple Genius.* MIT Press, Cambridge, Massachusetts, 1983.