The Computer Science Education Crisis: Fact or Illusion?

Hyacinth S. Nwana

University of Cambridge
Institute of Education
Shaftesbury Road
Cambridge
CB2 2BX, U.K.

e-mail: hyacinth@info.bt.co.uk

Abstract

Some contemporary or relatively recent articles, studied together, seem to suggest that computer science education is in a state of crisis. But, is it? This article examines the arguments. My preliminary findings is that if you examine the arguments from a *positivist* stance, you may well infer that there is a crisis, but if you view them from a *constructivist* perspective, the crisis becomes an illusion. Either way, the article concludes by setting a research question whose outcome may establish, more conclusively, if the crisis is indeed true or illusory.

1 Prolegomena

Computer science is a very young discipline and, *a fortiori*, a controversial, much-maligned and misunderstood one - even amongst computer scientists. It is truly absurd that a discipline which has been so evidently successful still faces such a crisis of identity. Computers/computing have irrevocably changed society in ways totally unpredicted, even by some of its inventors and other illustrious commentators, as is evidenced by these quotes from *The Sunday Times*, London, of the 12th November 1995 (section 6 - **12**):

> "Computers in the future will weigh no more than 1.5 tons.", *Popular Mechanics, forecasting advance of science, 1949*
>
> "I think there's a world market for maybe five computers.", *Thomas Watson, chairman of IBM, 1943*
>
> "I have travelled the length and breadth of this country and talked with the best people, and I can assure you that data processing is a fad that won't last a year.", *Editor in charge of business books for Prentice Hall, 1957*
>
> "There is no reason why anyone would want to have a computer in their home.", *Ken Olson, president, chairman and founder of Digital Equipment Corporation, 1977.*

The famous physicist, Neils Bohr, once wrote: "prediction is difficult, especially of the future". These quotes make this statement of Bohr's read like one of the understatements of the century.

"Computing is inextricably and ubiquitously interwoven into the fabric of modern life" (Hartmantis & Lin, 1992, 13). You have to look at the ubiquity of how computer software and hardware facilitate the delivery of much more efficient and high quality goods and services to an ever increasing number of people. Dijkstra (1989) argues that computers represent a radical novelty in our history. Their increasing power/speed and memories at ever decreasing costs have enabled them be the driving force behind much of modern life. Consider their roles in offices (e.g. spreadsheets, word-processors and databases), large businesses, telecommunications, transportation, medicine, science and engineering, finance, aeronautics, etc. Furthermore, "for every field X, it sometimes seems that someone creates a subfield, computational X" (Hartmantis & Lin, 1992, 61), e.g. computational medicine or computational physics. These authors also highlight, correctly, the fact that computing is becoming a third paradigm of scientific enquiry, on a par with theory and observation. This is because they often enable unexpected or unanticipated insights which elude the purely theoretical and experimental paradigms. The roles played by computer scientists and computer science education in making all these happen has been enormous.

Yet, despite these successes, computer science finds itself in the throes of conflict. Freeman (1995) notes correctly that, because of the centrality of computing to so much of modern life, computer science is heavily involved in many debates which extend beyond computer scientists, and rightly so too. Non-computer scientists notwithstanding, there is raging

debate amongst computer scientists as to what computer science is or is not, e.g. see the March 1995 issue of *ACM Computing Surveys* for a series of short articles which examine both technical questions in theoretical computer science and other broader issues. These articles are in response to the 1993 Turing lecture by Professor Juris Hartmantis, in which, *inter alia*, he discusses the nature of computer science.

Even the name "computer science" is *prima-facie* problematic. In a delightfully whimsical paragraph of his 1995 article, Mark Priestley notes why:

> "...it implies the existence of a discipline centred around a piece of technology rather than on any fundamental intellectual principles. Cynics might argue that equally compelling subjects could be invented, such as "refrigerator science": this would presumably cover all aspects of the construction and use of refrigerators, including electronics and cookery. Clearly the coherence of computer science as a discipline has to be argued for. (Priestley, 1995, 184).

Hence, the discipline's problems appears to begin with its *name*, let alone its *definition*. This fact has not just come to light: as far back as 30 years ago, the founding fathers of artificial intelligence, Allen Newell and Herbert Simon, had to defend the name 'computer science' against other contenders such as 'information science' and 'computer and information sciences' (Newell *et al.*, 1967, as quoted in Denning *et al.*, 1989, 11). They offered a definition (the study of computers and the major phenomena that surround them), which like many others since, fails to garner support from many other computer scientists. Some argue that computer science is engineering, others that it is an experimental science, and others that it is neither. Some will like to make it more mathematical while others just debunk current university computer science curricula. Such authors are cited later in this discussion.

Furthermore, others have questioned the *raison d'être* for computer science and some employers mistrust or dislike computer science education. Some (in many cases, themselves computer scientists) even go as far as questioning its very existence! Examples of these are also provided later. Thus despite computer science's successes, the debate on what it is continues. *Prima facie*, computer science education is in crisis - but, is it? I will argue in this paper that it depends on your 'paradigm lens' of choice, through which you decide to view the debate.

1.1 Breakdown of the rest of the paper

The breakdown of the rest of the paper is as follows. Section 2 reports on the motivations of this paper. Section 3 discusses the distinction between the positivist and the naturalist paradigms. Section 4 provides a historical review of computer science education and its curricula; it also overviews many definitions of the discipline. Section 5 overviews criticisms of the discipline culling from its many critics. It covers arguments that, together with others mentioned in Section 4, make computer science (CS), and hence CS education, appear to have a crisis of identity. Section 6 uses the distinction between the positivist and naturalist

paradigms of Section 3, to argue the case that the apparent crisis becomes an illusion if perceived from the viewpoint of an alternative paradigm. Section 6 also attempts to explain some of the issues of Section 5 in light of the new paradigm. Having established the case for a paradigm shift, Section 7 concludes the paper and raises a research question which could serve as the basis for an empirical investigation.

2 Motivations

There are two reasons which motivated me to examine computer science education: computer science-specific and one that is more general.

2.1 Computer Science-Specific Reasons

Back in 1974, D. L. Fisher, then head of the Computer Laboratory at the University of Leicester wrote a paper entitled 'Computer Science: a suitable case for treatment?' In this inspirational paper, Fisher expressed some, erstwhile, incandescent views. For example, he wrote lucidly:

> Basically, whether one likes it or not, whether it is true or not, the majority of employers mistrust (perhaps even dislike) computer science education. That education is often considered irrelevant and impractical, and the attitude of the computer science graduate is frequently arrogant and disoriented... Far better results are obtained, particularly in the commercial world, if graduates are taken from any other subject but computer science, for these graduates have an open mind and have none of the preconceptions that cause the computer scientist such terrible confusion. They can accept their new job at its apparent face value, and unlike the computer scientist, they do not have to go through that painful stage, whilst they discover and realise that their university education is of limited value" (Fisher, 1974, 1).

What made this quote memorable was the fact that I had heard similar comments from a couple of other UK industrialists: in fact, I had had to defend my profession against such a challenge at an interview. I was asked "what can you do which one of our in-house non-computer science graduates can't after having been on a short training course...?" - I did not get the job!

Not much later after I had read Fisher's paper, I stumbled across Professor Parnas' paper. In what may be described by some as a wholesale debunking of current computer science (CS) education, he writes:

> "In recent years, I have talked to a number of top industry researchers and implementors who are reluctant to hire CS graduates at any level. They prefer to take engineers or mathematicians, even history majors, and teach them programming" (Parnas, 1990, 19).

Fisher's and Parnas' quotes are almost two decades apart but convey the same message which other computer science lecturers may/must have heard too. Elsewhere in the paper, Parnas is even more cryptic:

> "As I look at CS departments around the world, I am appalled at what my younger colleagues - those with their education in computing science - don't know" (*ibid*, 18).

And the eminent computer scientist and Turing award winner, Edsger Dijkstra, also writes:

> "If I look into my foggy crystal ball at the future of computing science education, I overwhelmingly see the depressing picture of "business as usual". The universities will continue to lack the courage to teach hard science; they will continue to misguide the students, and each next stage of infantilization of the curriculum will be hailed as educational progress" (Dijkstra, 1989, 1402).

This list of quotes is by no means exhaustive, but they suffice to make my point that there appears to be a crisis in computer science education. Such viewpoints chiefly sparked my interest in investigating this area.

2.2 General Reasons

In addition to the domain-specific reasons, two further quotes capture the more general reasons that motivate such work. Firstly, if we look no further beyond current computer science educational practices, we are exposed

> "...to the charge that its sole function was to increase the efficiency of the existing system in terms of accepted criteria and deny it the opportunity to explore potentially more effective alternatives" (Eggleston, 1979, 5, as quoted in Bell, 1993, 16).

This refers to what Dijkstra refers to as "business as usual". Hence, periodic re-examinations and an understanding of the range of different views are needed for computer science education, as for any other discipline. Secondly, such studies

> "are a 'practical science' in the sense that we do not only want to know facts and to understand relations for the sake of knowledge, we want to know and understand *in order to be able to act and act 'better than we did before*[1]" (Langeveld, 1965, 4, as quoted in Bell, 1993, 16).

In the context of this paper, by understanding the range on viewpoints in computer science education, we can act in order to enhance it.

In the next section, we introduce briefly two educational paradigms which we draw from later in this discussion.

3 Positivism and Constructivism: A Brief Introduction

There are many paradigms to choose from in the educational literature. A paradigm is the basic set of beliefs that guide action, either in every day situations or in a disciplined enquiry (Guba, 1990). As Patton (1988) notes, paradigms tell practitioners what is important, legitimate and reasonable. Therefore, Patton continues, they are *normative* and largely implicit since they tell researchers what to do without the necessity of long existential or epistemological considerations. Paradigms also have their associated methodologies and

[1]The italics represent my emphasis

methods; they are enabling, but also constraining. In any case, a powerful paradigm grew up in the nineteenth century as an account of the rise of science and technology. So powerful was it that by the late nineteenth century, *positivism* had become the dominant philosophy, though its epistemology only took beguiling clarity in the early twentieth century, thanks to the Vienna circle of positivists (Schon, 1983).

In brief, the paradigm of positivism holds that there is a reality 'out there' governed by immutable laws. Thus it is argued that the purpose of science is to discover how this reality works, in order to predict and control phenomena. In the context of this paper, there is a single 'computer science domain' reality out there which is distinctly separate from those doing the seeing, i.e. the computer science authors. This reality is time-free as well as context-free (i.e. *nomothetic*). Furthermore, positivism assumes the inquiry process is value-free since it uses an 'objective' methodology. In my context, the computer science education researcher is supposedly not influenced by any values since the enquiry process is context-free. A key derivation of this view is that there is a shared reality as perceived by all researchers, and if you do not share this reality, you must be either *wrong* or *unobjective*.

In contrast, there is another paradigm school of choice which could not be more different from positivism. *Constructivism* holds that positivism is flawed and must be replaced. It holds that there is no reality out there. In my context, there is no single tangible reality out there called 'computer science' which is fragmentable as a computer is. It maintains that there are multiple *constructed* realities and they are best studied holistically. It also refutes most of the essential tenets of positivism: constructivism is about *understanding*, not about prediction or control; the inquirer or researcher and the domain are not independent, i.e. the computer science knower and the known are inseparable; computer science is time and context-bound (i.e. *idiographic*) in contrast to it being nomothetic; and inquiry is value-bound, not value-free. The resulting effect of these is that researchers working within the constructivist paradigm see reality as a social construct, and so do not necessarily expect other researchers to have the same perceptions or understanding of shared phenomena. In this view, the domain of computer science is a construction unique to the individual, department, university or nation.

Despite the fact that positivism has been discredited by vanguard thinkers in many disciplines, it continues to this day (for understandable reasons) to guide the efforts of many researchers. However, since I hold a constructivist viewpoint, I do not necessarily expect the reader to subscribe to this paradigm as well. But, I would advise against what I call *paradigm enslavement*. Patton writes in his chapter on 'Paradigms and Pragmatism':

> "Pragmatism can overcome seeming logical contradictions. I believe that the flexible, responsive evaluator can make mind-shifts back-and-forth between paradigms within a single evaluation setting. In so doing, this evaluation can view the same data from the perspective of each paradigm, and can help adherents of either paradigm interpret data in more than one way" (Patton, 1988, 127).

4 Computer Science Education: A Review

Computer Science did not have an easy birth and, as can be alluded from the previous section, it has not had an easy 'growing up' ever since. For instance, Fisher, in his 1974 article already quoted from, argued that Computer Science was not yet a suitable subject for a dedicated first degree. He was in good company: Cambridge University did not offer a 3-year Computer Science Tripos or honours degree until 1988! Why?

> "Cambridge was deliberately slow to offer undergraduate courses, waiting for the discipline to mature" (Bacon, 1991, 174).

There are those who argue that this is still the case, e.g. Hirose (1990).

4.1 A Brief History of Computer Science

The stored program computer was invented in the 1940s and and the newest engineering profession, Computer Science, was subsequently born; it has a short but rapidly evolving history (Bacon, 1991). Universities began offering computer science diplomas or degrees in the early 1960s, i.e. just over 30 years ago. Parnas (1990) notes that computer science departments were formed by multi-disciplinary teams comprising mathematicians interested in computing, electrical engineers who had constructed and used computers, and physicists who had principally been computer users. At first, computing was part of electrical engineering or mathematics departments, and they offered courses with titles like Cambridge's then Diploma in 'Numerical Analysis and Automatic Computation', which began in 1953. Later, the academics in this new and exciting venture grew restive under the yoke of their parent departments, and in the early 1960s, there was a move to set up Departments of Computer Science (Barron, 1989). Naturally, there was opposition from some mathematicians and electrical engineers who felt that computing was an integral part of their disciplines, and that a discipline of computing on its own will be too shallow (Parnas, 1990). Their fear was that the subsequent computer science graduates would neither be mathematicians nor engineers. Barron (*ibid*) was one among many who think, perhaps with the benefit of hindsight, that the name 'computer science' was rather grandiose and unmerited for the early 1960s. He writes:

> "...the high sounding 'Computer Science' really meant 'all about computers and programs', but you cannot use that as the name of an academic department" (page 27).

In the event, the opposing voices were drowned out and the subject was born. Today, just about every university worldwide offers a computer science qualification of some sort - such has been the 'success' of the discipline. But, then again, what is computer science?

4.2 What is Computer Science?

Naturally, such a question evokes positivist and varied responses, and hence is problematic. However, before I tackle this question, I make several relevant comments. Firstly, as noted earlier, I believe it is what the researcher, department or institution defines is to be, preferably after a thorough understanding and acceptance of other views. Secondly, computer science is a rapidly changing field which makes it less amenable to precise definition. Thirdly, in the light of the current vexatious debate on what computer science is, no new definition would satisfy all. So, in keeping with the constructivist tradition, I offer a *summary* of significant features of a range of definitions/viewpoints proffered by other authors. Due to the 6000 word limitation of this paper, the full quotations are offered in the appendix and I would encourage the reader to read them fully. This summary begins to convey a flavour of the apparent identity crisis that computer science is in. Views on 'what is computer science' include:

- 'The study of computers';
- 'The impact of computers on society';
- 'Deals with information, its creation and processing, and with the system that performs it';
- 'A new species among the sciences';
- 'A new species of engineering';
- 'Any attempt to separate them (computer science and engineering) is counter productive';
- 'Clearly, computer science is not a physical science';
- 'Computer science is an experimental science';
- 'The systematic study of algorithmic processes that describe and transform information'
- 'Is there a computer science?';
- 'Computer science may become irrelevant';
- 'Computer science is a collection of techniques';
- 'Computer science is *not* a collection of techniques';
- 'Computer science is a craft, an art, and to some extent a science'.

These viewpoints are by no means exhaustive; space limitations prevents me from providing more. Clearly, from these, seeking a consensus definition is an exercise in futility though there is some overlap between a few of them. Some of these viewpoints (in addition to others noted earlier on in this paper) give the impression of a subject descending into an astonishing bout of internecine warfare over what it is, its future direction, and that the fate of computer science remains in question. But then again, perhaps this range of viewpoints befits a discipline still in an inchoate state. Next, I proceed to review what lies beyond these definitions/viewpoints - the curriculum.

4.3 From Curriculum 65 to Curriculum 91: an Overview

Curricular work in computer science dates as far back as the 1962. Then, in the United States, a committee was formed called the Curriculum Committee on Computer Science (C^3S). It was constituted initially as a sub-committee of the Education Committee of the Association for Computing Machinery (ACM). It must be noted at this juncture that the ACM's contribution, since, to computer science education has been immeasurable, and is now, arguably, the leading and most reputable computing society in the world. However, C^3S (1968) explain that C^3S functioned informally for a couple of years until it became an independent committee of the ACM in 1964, when it began to formulate, actively, detailed recommendations for curricula in computer science. It went on to produce the first computer science curriculum in 1965 in a report entitled "An Undergraduate Program in Computer Science - Preliminary Recommendations" (C^3S, 1965). Though only a preliminary report, it defined *the* framework for later reports. Naturally, being the first, it devoted considerable attention to justifying computing as a new and distinct discipline (rather than one which was part of others), and in promulgating the name 'computer science' against other rival names including "Computer and Information Sciences" (Gorn, 1963) and "Information Science". The report also proposed sixteen courses which defined the scope of the discipline and went on to provide detailed outlines and bibliographies for these courses, many of which survived the later 1968 curriculum. In any case, C^3S received much valuable feedback on this preliminary report in terms of comments, criticisms and suggestions. Drawing from these, and from the advice of numerous consultants, C^3S went on to produce its first proper curriculum recommendations - Curriculum 68 (C^3S, 1968).

Curriculum 68 identified and described twenty two courses which made up computer science, and like Curriculum 65, it presented the prerequisites, catalogue descriptions, detailed outlines and annotated bibliographies for these courses. It grouped the subject areas of computer science into three major areas: 'information structures and processing', 'information processing' and 'methodologies'. The courses were drawn mainly from computing and the mathematical sciences with some contributions from the physical and engineering sciences. However, in contrast to Curriculum 65, it also went on to discuss and

provide recommendations for graduate programs, though some specific recommendations were made for master's degree programs only: none were made for doctoral programs though some some guidelines were provided. It also paid some attention to staff requirements, computer facilities and other resources required to implement computer science programmes.

Ten years later came Curriculum 78 (C3S, 1979) which once again was the result of numerous suggestions and criticisms from varied interests within computer science. In addition to the consolidation of the discipline, it had also been evolving since Curriculum 68. The format of Curriculum 78 is almost identical to its predecessor but it introduced the notion of a 'core curriculum' - one that should be common to all computer science undergraduate programs. It presented these in terms of a set of elementary and intermediate level courses. It also identified elementary and advanced level elective courses that may be used to round out an undergraduate program. The structure of most current computer science curricula at universities worldwide emanates, clearly, from Curriculum 78.

The latest of the curricula is that which was published in 1991 (ACM/IEEE-CS, 1991), some thirteen years after the previous one when the discipline had further matured considerably and it was time the curriculum was updated. What also makes this more authoritative than any others that had preceded it stems from the fact that it represents the efforts of not only the ACM's C3S committee, as had traditionally been the case, but also of members of the Institute of Electronic and Electrical Engineers (IEEE) Computer Society's Education Board. The report is based largely on the paper 'Computing as a Discipline' (Denning *et al.*, 1989) which provides a clear definition of the discipline and goals for its various sub-programs. Indeed, it fleshes-out and uses the subject matter classification presented in the Denning paper. The report is also unique in that it provides a uniform set of recommendations to a broad church of programmes including computer science, computer science and engineering and liberal arts programmes with a major in computing.

The previous reports had also been more like design documents, i.e. there was a clear recommendation on *how* to group courses together to form a programme (Engel, 1991). Computing Curricula 1991 set out clearly to be more of a specification (i.e. it proposes *what* should go into these programmes without any clear direction as to how they should be put together). However, it proceeds to providing eleven sample implementations for various programmes, but the idea is to allow flexibility in terms of how programmes are assembled.

The report splits computer science into nine areas of study, namely algorithms and data structures, architecture, artificial intelligence and robotics, databases and information retrieval, human-computer interaction, operating systems, programming languages, numeric and symbolic computation, and software methodology and engineering. In addition to these, it recommends the teaching of social, ethical and professional issues associated with

computers. These areas are split further into modular knowledge units and the report also places a stronger emphasis on laboratories than any of its predecessors. Curricula 91 appears to have been endorsed by most computer science departments judging from the near-zero critical reviews I have come across thus far; indeed, McGettrick (1991, page 30) claims that it represents an important landmark in the development of computing. The fact that it was the result of the joint efforts of two internationally-respected computing bodies also gives it much well-merited credence.

5 Computer Science Education: Criticisms

Despite the evolution of CS curricula, computer science education still has many critics. Indeed, its critics are as vehement as they are eminent. However, it must be emphasised that the viewpoints included in this section, as with most others in the rest of the paper (e.g. see Sections 2, 4 and the Appendix), are open to the criticism that they are *assertions* only. Rarely do the authors ground these assertions on sound educational studies. This said, I believe their long and rich personal experiences cannot or, rather, should not be ignored.

5.1 Key Criticisms

Culling from a distributed literature, the charges against computer science education are, at least, fourfold.

5.1.1 Computer Science is neither mathematics, science nor engineering

> "Computer science graduates "are very weak on fundamental science; their knowledge of technology is focussed on the very narrow areas of programming, programming languages, compilers and operating systems. They confuse existence proofs with products, toys with useful tools." (Parnas, 1990, 18).

Parnas proceeds to discuss a litany of other issues which he claims computer scientists either do not understand, confuse, do not know or have a very shallow understanding of. He argues that computer science programmes do not prepare students to apply mathematical and engineering fundamentals to the design of computing systems - in short, computer science is neither mathematics nor engineering, and as the last quote states, it is weak on fundamental science. To Parnas, we have the worst possible scenario that could have befallen the discipline, and he maintains that the dire predictions of mathematicians and electrical engineers back in the 1960s have turned out to be correct. Indeed, further to his earlier quote in which he lamented what computer scientists do not know (see Section 2.1), he writes:

> "My criticism of the education we now provide is unavoidably a criticism of the preparation of my younger colleagues. A university's primary responsibilities are to its students and society at large. It is unfortunate that they are usually run for the comfort and happiness of the teachers and administrators. In this matter, the interests of our students and society coincide. It is not in students' interest to make them perform engineering without being prepared for that responsibility. Nor is it in their interest to give them an education that prepares them only to be technicians. Too many graduates

> end up "maintaining" commercial software products, which is analogous to electrical engineers climbing poles to replace cables on microwave towers" (Parnas, 1990, 22).

Though the key target of this incisive criticism are university teachers and administrators, his claim that computer scientists are no better than technicians certainly hits a raw nerve with computer scientists. He concedes that such comments will not endear him to computer science faculty, especially as he asserts that public safety is seriously affected by the fact that computer science graduates program parts of systems that run nuclear plants or fly aircraft. But his comments carry much currency, not least because of their clarity and his considerable experience. He presents an alternative programme which "more closely resembles a heavily packed engineering program than the liberal arts program to which CS educators have become accustomed" (page 21). He would also welcome five-year undergraduate degrees!

5.1.2 Computer science education is infantile and immature

Computer science education is 'infantile' (Dijkstra, 1989). Essentially, Parnas (1990) is making this charge too. Dijkstra's paper is titled "On the Cruelty of Really Teaching Computer Science". Though he shares similar sentiments on the state of computing education to Parnas, his proposal is to make the discipline more mathematical/formal, rather than more engineering-like. He argues that the refusal to exploit the power of mathematics by computer scientists amounts to intellectual and technological suicide. Arguing from the viewpoint that computers represent a "radical novelty in our history", he notes that the discipline has been "unfathomedly misunderstood" due to the inappropriate analogy to engineering - hence, software engineering, software maintenance, quality control, software productivity, etc. He would like to see computing science transcend its parent disciplines, mathematics and logic[3], "by effectively realizing a significant part of Liebniz's Dream of providing symbolic calculation as an alternative to human reasoning" (page 1402). But Dijkstra himself is less optimistic. He opines:

> "I now have had my foggy crystal ball for quite a long time. Its predictions are invariably gloomy and usually correct. However, I am quite used to that, and they will not keep me from giving you a few suggestions, even if it is merely an exercise in futility whose only effect is to make you feel guilty" (Dijkstra, 1989, 1402).

He then goes on to make some useful suggestions. In summary, Dijkstra's main charge is that computer science education is infantile as it lacks the courage to teach hard science. As earlier noted in this paper, he argues that computer science students are being misguided. The successive curricula are of limited improvement: in his words, "each stage of infantilization of the curriculum will be hailed as educational progress".

---

[3] Note that Dijkstra believes that computer science's parent disciplines are mathematics and logic, in contrast to the more established view of mathematics and electrical engineering.

Several also still level the charge that computer science is still immature to be a discipline in its own right despite the fact that it is now more than 30 years old. However, this should not be confused with Dijkstra's 'infantile' argument which, stated bluntly, translates to 'computer science education does not want to grow up'. Hirose (1990), for example, maintains that there are many "theories" in computer science, but there is still little foundation to them. He also argues that current computer science curricula are still at the stage of trial and error, and that many in the discipline are not aware of this fact. Unfortunately, Hirose does not expand on his assertions. Another form that this charge manifests itself is via those who argue that computer science departments were formed too soon (e.g. Fisher, 1974; Parnas, 1990; Bacon, 1991). They argue that very little computing science is fundamental enough to be worth teaching to undergraduates, which is why they would rather see more mathematics and/or engineering courses taught to computer scientists. Recall that Cambridge did not feel the subject had matured enough to merit a dedicated first degree until 1988 - only eight years ago. Indeed, though written in 1974, astonishingly, Fisher captures succinctly the essence of the immaturity argument which still thrives as per 1996:

> "Perhaps computer science is still not worth teaching at undergraduate level as the total substance of a degree course. Computing is still in its infancy...Computing still lacks real depth.." (Fisher, 1974, 1).

5.1.3 Is there a computer science?

Henderson (1991) accuses computer science of having fallen prey to some superficially very attractive ideas (of which he counts parallel processing, formal methods, artificial intelligence and expert systems) and that there is nothing scientific about computer science. He asserts:

> "To describe what is taught at universities and polytechnics as computer science distorts our understanding of what is involved in the production of systems using computers. There are two important adverse consequences. Large amounts of government money have been poured into university research, and the opportunity to educate undergraduates appropriately has been lost" (*ibid*, 12).

In the same article in which he begins by openly questioning if there is a computer science, he proceeds to assert that computing (as it stands currently) is no more than a collection of techniques. Certainly, this assertion did hit, at least, one raw nerve: it stung Simpson (1991) into issuing a rebuttal claiming that Henderson's article "seemed nihilistic and depressing" (page 24).

5.1.4 Computer science graduates are not suitable for many employers

This is a more ubiquitous charge levelled at computer science education and it takes me back to the key motivation for this work (see Section 2.1). Like it or loathe it, Fisher's (1974) viewpoint that some employers mistrust computer science education still thrives on, as is evidenced in Parnas (1990). As I have personally experienced in the past, there are some employers who would prefer to hire engineers or mathematicians (even history majors as

Parnas claims), and teach them programming. Certainly, this view is held by some in my company - British Telecommunications PLC (BT). For example BT's Prof. Robin Smith, who has been hiring graduates for the best part of 20 years to do mainly *computing-based* jobs, hold similar sentiments. The pecking order for his preferred graduates leaves computer scientists trailing a poor third after electrical engineers and physicists (Personal communication).

5.2 Is there a Crisis in Computer Science Education?

This question must be a logical inevitability from the arguments expressed so far. Let me review the evidence. In Section 4, I demonstrated that computer scientists themselves seem to be in some confusion over what computer science is. Indeed, I argued there that some of the viewpoints give the impression of a subject descending into a bout of internicine warfare over what it is, and its future direction. The debates are so polarised that it seems inconceivable that they could be reconciled. I then proceeded, also in Section 4, to review successive computer science curricula, but in Section 5, some very eminent computer science scholars debunked them as infantile and immature. Computer science is accused of neither being mathematics, science nor engineering. More worrying still are those who question whether there is indeed a science of computing. Many more question the *raison d'être* for computer science education. Perhaps most worrying of all, are the persistent views that computer science graduates are not suitable for some employers, who appear to distrust computing qualifications.

Prima facie, computer science education *must* be in crisis. But, is it really? Astonishingly, perhaps to the reader, I think not, though I do acknowledge that, using positivism as a paradigm lens of choice, it would/may be interpreted that there is indeed a crisis. I must come clean of my own predisposition and note that I have eschewed positivism in favour of constructivism as my paradigm lens of choice, and the result of this move is that I see no crisis, *yet*, in computer science education.

6 The Computer Science Education Crisis is an Illusion

Using the positivist lens, the charges levelled against computer science education (see Section 5) and/or the disagreements amongst computer scientists as to what computer science is or is not, are arguably irreconcilable. For example, how do you reconcile the following: computer science is an experimental science (Plaice, 1995); computer science is engineering (Loui, 1995); computer science is currently neither science nor engineering (Parnas, 1990); and computer science should be more mathematical (Dijkstra, 1989)? Positivism illuminates such areas of strong disagreement. A key point of this paper is that such positivist positions tend to lead to polarised debates. Even the language adopted in the debate is very positivist. For example, Dijkstra (1989) writes:

> "...it was such as obvious case for curriculum infantalization that its author should be cited for "contempt of the student body", but this was only a minor offence compared with..." (page 1403).

The point here is that such language is, arguably, too strong and dismissive of other opinions. Though, I have only given an example from one side of the debate, similar language is also used on the other side too. For instance, I have heard Dijkstra's views referred to as 'eccentric'. Even a couple of the Dijkstra's quotes seem to suggest that he is resigned to such labelling. Such language (on all sides of the argument), which trades in such certainties, is anathema to constructivists, and ensures that no consensus is reachable. It also gives currency to both sides of the argument dismissing, and hence not understanding, each others' viewpoints. Therefore, though Dijkstra and Parnas write very erudite articles on this subject, they tend, at best, to be ignored and, at worst, considered eccentrics as their viewpoints do not agree with the majority view. Follow-up articles only further entrench their positions which exacerbate the problem. Therefore, a crisis is perceived.

However, constructivism accepts as axiomatic that such differences in viewpoints occur and seeks to locate areas of consensus. A cardinal rule of this paradigm is that no viewpoint is 'eccentric'. Afterall, the majority view is not necessarily the more trusted one; all views must be understood.

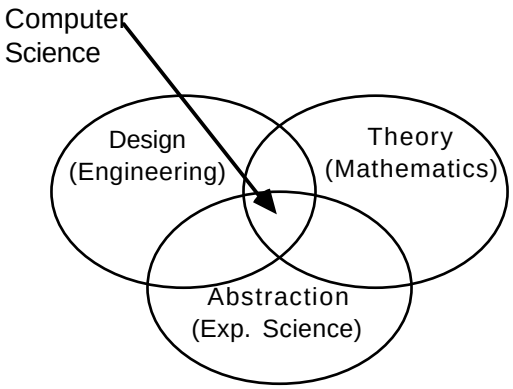

Figure 1 - A consensus 'situation' of computer science

Figure 1 shows how areas of consensus can be arrived at from the different viewpoint expressed in this paper. It draws from Denning *et al.*'s (1989) work on which Curricula '91 is based. In this paper, the authors argue that computer science derives from three major paradigms: *theory*, rooted in mathematics; *abstraction* (modelling), rooted in experimental science; and *design*, rooted in engineering. Hence, computer science falls at the intersection of all three as shown in Figure 1.

If we now return to Section 2.1 (also refer to the Appendix) and using such a framework as Figure 1, we can now reconcile computer science being both an experimental science and engineering. Hartmantis' (1995) point, that drawing a sharp line between computer science

and engineering is counter-productive, now makes much sense. Henderson's (1991) question 'is there a computer science?' is no longer that problematic, because it is clearly *not* a science in the exact Popperian sense; rather computer science is a discipline which inherits some, but not all, of the attributes of a traditional science discipline, along with other attributes from engineering and mathematics. Likewise Loui's (1995) and Hartmantis' (1995) views, that computer science is a new species among the sciences and engineering respectively, are accommodated easily in this framework. Even Simpson's (1991) idea that computing is a craft and an art also derives from the abstraction paradigm. Turning to Section 5.1, Parnas' (1990) argument that computer science is neither mathematics, science nor engineering is *logically* explicable by Figure 1. Indeed, it is all three.

Whether the reader agrees with the accuracy of the preceding analysis or not is not as important; rather, I am emphasising here how a constructivist approach can lead to a consensus from seemingly irreconcilable viewpoints. One by one, the issues are being reconciled, though not all necessarily are. Admittedly, the framework has not addressed other criticisms raised in Section 5.1 (e.g. computer science education is infantile), but these remaining criticisms cannot be addressed by any other framework; rather, they can only be addressed by further *research.*

In effect, the apparent positivist crisis becomes an *illusion* when viewed from a constructivist lens! I now provide a viewpoint, arguably written from a positivist stance, but which captures succinctly the constructivist notion of avoiding splits in favour of seeking consensus:

> "Are we scientists or engineers? ...I reject the title question. I call my discipline Computer Science because that's the common name in the US - but my discipline spans a multidimensional spectrum from deep and elegant mathematics to crafty programming, from abstraction to solder joints, from deep truth to elusive human factors, from scholars motivated purely by the desire for knowledge to practitioners making my everyday life better. It embraces the ethos of the scholar as well as that of the professional. To answer the question would be to exclude some portion of this spectrum, and I would be poorer for that" (Wulf, 1995, 55-57).

So what is the main point of this paper? Basically, there are several approaches to viewing and dealing with the conflicting viewpoints on computer science education. First, there are those who may be prepared to ignore them but only up to a point, due to the insightful nature of some of the criticisms and because of the eminence of some of the critics themselves. Alternatively and simply, they may be explained away by noting that the birth of any new discipline is usually fraught with similar difficulties, which die off as the discipline matures. This is arguably right but such a response reads as a *mea culpa*. In any case, the criticisms have not died off; indeed, three decades after the birth of the discipline, they show no signs of abating. A more positive way of looking at them is by noting that the debates depict a healthy discipline. However, they tend to adopt very *positivist* stances, with the inevitable

result that the debates become very polarised and never seem to converge: each researcher adopts an 'I am right - you are wrong' position.

In this paper, I have entered the fray with a different perspective: I eschew positivism in favour of *social constructivism*. Constructivists accept as axiomatic that authors will construct and hold different and *subjective* viewpoints, which derive from their past experiences, education, opinions, values, morals, culture and individual contexts. Indeed, as Guba (1990) points out,

> "constructivists not only abjure objectivity, but celebrate subjectivity", page 17.

I consider it important to *understand* 'the range' of individual views held by different authors on computer science education, to enable me *construct* my own individual viewpoint. Using this stance, no viewpoint is 'deviant' or 'eccentric' as positivists tend to describe those who do not share the 'representative' view. Essentially, I believe that computer science is what the researcher, department or institution *constructs* and defines it to be - *preferably, after a thorough understanding and acceptance of the range of viewpoints proffered by other authors*. My adoption of this stance is not only philosophically-driven, but pragmatic: it seems a certainty that some of the different poles of the computer science debate are totally irreconcilable within the positivist framework. Finally, the approach I have adopted in this paper coincides with Rorty (1982); being a constructivist, he also believes pragmatism offers the possibility of harmonising different research traditions.

7 Conclusion and a Research Question

I have argued the case that computer science has been a successful discipline, but that it is also controversial and much misunderstood. I have identified the misunderstandings as stemming, in part, from seemingly conflicting viewpoints proffered by respected computer science authors on the definition and future of the discipline. After a critical review of its history and curriculum, I have argued the case that these conflicting viewpoints and severe criticisms depict a crisis when viewed from a positivist lens, but they become illusionary when viewed from a constructivist one. I also conclude that faster progress would occur in the discipline if CS education authors adopt more constructivist positions and language. As a summary, I hope I have contributed to enhancing CS education by, at least, pointing out another paradigm lens which ameliorates, but probably not eliminates, some of the polarised debates which tends to characterise our discipline.

However, I am still aware of the outstanding criticisms of computer science education: it is shallow; it is infantile; and its graduates are not suitable for many employers. I argue that the evidence provided by the accusers in support of these criticisms is largely anecdotal. Therefore, I propose an umbrella research question which could resolve these criticisms: does computer science education meet the needs of its stakeholders? However, since the

curriculum is at the core of computer science education and Curricula '91 is the most recent curriculum, the question may be reposed as follows: t*o what extent does ACM/IEEE-CS's Computing Curricula 1991 meet the needs of stakeholders in computer science education*? Stakeholders include computer science academics, industrialists, students, professional computing bodies, politicians and society at large. Only when the results of such a study yields, conclusively, negative results can we possibly suggest that computer science education is in crisis. Until then, we conclude, preliminarily, that whatever crisis is perceived is illusionary.

8 Acknowledgements

I am indebted to Nader Azarmi who allowed me the time to pursue this work. I acknowledge useful discussions and comments from late Brent Robinson and Dr. Colin Conner of the School of Education, University of Cambridge.

9 References

ACM Computing Surveys, 27, 1, March 1995

ACM/IEEE-CS (1991), 'Computing Curricula 1991' *Communications of the ACM*, 34, 6, June, 69-84

Bacon, J (1991) 'Computer Science and Computing Education' *The Cambridge Review*, December, 173-177

Barron, D (1989) 'Two cultures: computer science and personal computers' *Endeavour, New Series*, 13, 1, 25-28

Bell, J (1993) *Doing Your Research Project* Milton Keynes: Open University Press

C3S (1965) 'An Undergraduate Program in Computer Science - Preliminary Recommendations' *Communications of the ACM*, 8, 9, September, 543-552

C3S (1968) 'Curriculum 68: Recommendations for Academic Programs in Computer Science' *Communications of the ACM*, 11, 3, March, 151-197

C3S (1979) 'Curriculum 78: Recommendations for the Undergraduate Program in Computer Science' *Communications of the ACM*, 22, 3, March, 147-167

Denning, P J (1995) 'Can there be a Science of Information?' *ACM Computing Surveys*, 27, 1, March 1995, 23-25

Denning, P J, Comer, D E, Gries, D, Mulder, M C, Tucker, A B, Turner, A J & Young, P R (1989) 'Computing as a Discipline' *Communications of the ACM*, 32, 1, January, 9-23

Dijkstra, E W (1989) 'On the Cruelty of Really Teaching Computer Science' *Communications of the ACM*, 32, 12, December, 1398-1404

Engel, G (1991) 'Computing Curricula 1991 - a Résumé' *The Computer Bulletin*, June, 29

Fisher, D L (1974) 'Computer Science Education - a suitable case for treatment?' *Paper presented at the IUCL Computer Science Colloquium*, University College, Swansea, September

Freeman, P A (1995) 'Effective Computer Science' *ACM Computing Surveys*, 27, 1, March, 27-29

Goldschlager, L & Lister, A (1982) *Computer Science: A Modern Introduction*, London: Prentice Hall

Gorn, S (1963) 'The Computer and Information Sciences: a new basic discipline' *SIAM Review* 5, 2, April, 150-155

Guba, E G (1990) 'The Alternative Paradigm Dialog' in Guba, E G (ed) T*he Paradigm Dialogue*, London: Sage, 17-27

Hartmantis, J (1995) 'Turing Award Lecture: On Computational Complexity and the Nature of Computer Science' *ACM Computing Surveys*, 27, 1, March 1995, 7-16

Hartmantis, J & Lin, H (Eds.) (1992) *Computing the Future: A Broader Agenda for Computing Science and Engineering*, Washington, D. C.: National Academy Press

Henderson, J (1991) 'Is there a Computer Science?' *The Computer Bulletin*, February, 12-14

Hirose, K (1990) 'On Computer Science Curricula at Universities' *New Generation Computing*, 8, 183-184

Loui, M C (1995), 'Computer Science is a New Engineering Discipline' *ACM Computing Surveys*, 27, 1, March, 31-32

McGettrick, A (1991) 'Computing Curricula 1991 - a review' *The Computer Bulletin*, June, 30-32

Parnas, D L (1990) 'Education for Computing Professionals' *IEEE Computer*, 23, 1, 17-22

Patton, M Q (1988) 'Paradigms and Pragmatism' in Fetterman, D M (ed) *Qualitative Approaches to Evaluation in Education: The Silent Revolution*, New York: Praeger, 116-137

Plaice, J (1995), 'Computer Science is an Experimental Science' *ACM Computing Surveys*, 27, 1, March, 33

Priestley, M (1995) 'Discipline or Punish: The Cruelty of Not Teaching Software Engineering', In Myers C (Ed.) *Professional Awareness in Software Engineering*, London: McGraw Hill, 176-192

Rorty, R (1982), *Consequences of Pragmatism* Minneapolis, MN: University of Minnesota Press

Savage, J E (1995) 'Will Computer Science become Irrelevant?' *ACM Computing Surveys*, 27, 1, March, 35-37

Schon, D (1983) *The Reflective Practitioner. How Professionals Think in Action*, New York: Basic Books

Simpson, J (1991), 'Computer what? Science? Craft? Art? Techniques?' *The Computer Bulletin*, September, 24-25

Tucker, A B, Barnes B H, Aiken, R M, Barker, K, Bruce, K B, Cain, J T, Conry, S E, Engel, G L, Epstein, R G, Lidtke, D K, Mulder, M C, Rogers, J B, Spafford, E H & Turner, A J (1991) 'Computing Curricula 1991' *Communications of the ACM*, 34, 6, 69-84

Wulf, W A 'Are we Scientists or Engineers?' *ACM Computing Surveys*, 27, 1, March 1995, 55-57

Appendix

(What is Computer Science: Some Viewpoints from Computer Scientists)

This appendix provides the full quotations of various authors' viewpoints on the 'What is Computer Science' debate summarised in Section 4.2. I confess to posing this question to enable me to provide the following views which, hopefully, begins to convey a flavour of the apparent identity crisis that computer science is in.

> "Computer science is the study of computers and the major phenomena that surrounds them" (Newell *et al*., 1967, as quoted in Denning *et al.*, 1989, 11).
>
> "Computer science is a discipline with many facets, ranging from the impact of computers on society through to the technological details involved in computer design" (Goldschlager & Lister, 1982, xi).
>
> "Computer science differs so basically from the other sciences that it *has to be viewed as a new species among the sciences*[2], and it must be understood. Computer science deals with information, its creation and processing, and with the system that performs it, much of which is not restrained and governed by physical laws" (Hartmantis, 1995, 10).
>
> "Hartmantis calls computer science "a new species among the sciences", whose science and engineering aspects are particularly close. I have argued that it will be *more accurate to call computer science a new species of engineering*[2]" (Loui, 1995, 31).
>
> "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 counter productive" (Hartmantis, 1995, 14).
>
> "Clearly, *computer science is not a physical science*[2]; 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" (Hartmantis, 1995, 12-13).
>
> "C*omputer science is an experimental science*[2] ... We *are* in fact experimental scientists" (Plaice, 1995, 33).
>
> "The discipline of computer science is the systematic study of algorithmic processes that describe and transform information: their theory, analysis, design, efficiency, implementation, and application. The fundamental question underlying all of computing is, "what can be (efficiently) automated?" (Denning *et al.*, 1989, 12).
>
> "Will computer science become irrelevant?" (Savage, 1995, 35).
>
> "The challenge facing us as computer scientists is to make sure that computer science as a discipline of education and research does not, in fact, become irrelevant. How can this be? How can we be so important in so many ways and at the same time in danger of

---

[2] The italics represent my emphasis.

becoming irrelevant? ... Computing is going to be used more and more *whether or not computer scientists are involved*. We must find ways to make progress on the intellectual questions and educational agendas that motivate us, while earning enough of society's resources to support those endeavors. If we don't, then we may well find that as a field we have become irrelevant" (Freeman, 1995, 27).

"Computing science is - and will always be - concerned with the interplay between mechanized and human symbol manipulation usually referred to as "computing" and "programming" respectively" (Dijkstra, 1989, 1401).

"Is there a computer science?... I propose a simple definition of what computing is about. Computers should be used to do that which could not be done before and/or to do more efficiently that which was previously possible. All useful computing to date can be seen to fit into one or the other of these categories... If this definition is accepted perhaps some of the mystique can be dropped. It already has in many areas...computing is a collection of techniques..." (Henderson, 1991, 12-14).

In response to Henderson (preceding), Simpson writes:

"Computing is a craft, an art, and to some extent, a science. Its complexity and diversity defies simplistic definitions. The name 'computing science' is a hostage to fortune and we should perhaps change it to 'informatics' or simply 'computing' to avoid all the carping. What is really hard to stomach is the idea that computing is just a 'collection of techniques'. I completely reject the idea" (Simpson, 1991, 25).