120th ASEE Annual Conference & Exposition
FRANKLY, WE DO GIVE A D*MN
June 23-26, 2013

Paper ID #6570

Applying a knowledge-generation epistemological approach to computer science and software engineering education

Dr. Stephen T Frezza, Gannon University

Dr. Stephen T. Frezza, C.S.D.P. is a Professor of Software Engineering at Gannon University in Erie, PA (USA). Dr. Frezza is a Certified Software Development Professional (CSDP), and at Gannon pursues research in Program Assessment, Software Engineering Pedagogy, and Engineering Philosopy. His teaching interests include Software Process, Requirements, Design, Testing and Quality Assurance. He is the past chair of the Computer and Information Science Department in delivering accredited undergraduate Computer Science and Management Information Systems programs, as well as an undergraduate Software Engineering and graduate Computer and Information Science program.

Dr. Richard W. Moodey, Gannon University

Dr. Moodey received his PhD in sociology from the University of Chicago in 1971. He has taught at Loyola University of Chicago, Allegheny College, and is currently chair of the department of Criminal Justice and Social Work at Gannon University.

Dr. David Arthur Nordquest, Gannon University

David A. Nordquest is Assistant Professor of Philosophy at Gannon University.

Mr. Krishnakishore Pilla P.E., Gannon University

Applying a knowledge-generation epistemological approach to computer science and software engineering pedagogy

TLC Topic Area: Concepts and Philosophy of Engineering Literacy

Abstract This paper proposes a brief exploration of the epistemology of knowledge, specifically distinguishing the development of scientific knowledge from the development of engineering knowledge. Based on a pragmatic theory approach (1), the paper proposes a pattern for distinguishing the 'science' of computer science from its 'engineering' aspects. The paper then applies these distinctions to traditional Computer Science knowledge, and explores its relationship to 'engineering science'. The implications of this knowledge-generation approach to discipline exploration are then applied to Software Engineering as an engineering discipline. This application aims at distinguishing Software Engineering from the scientific and engineering aspects of Computer Science.

A cursory introduction to the literature of the philosophy of engineering reveals competing viewpoints on what distinguishes scientific from engineering knowledge, including engineering (and technology) as applied science (2 p. 42) and the influence of knowledge generation as a means to distinguish between 'scientific' and 'engineering' knowledge. When seen through the lens of a pragmatic theory of knowledge, the crucial characteristics of scientific knowledge include that scientific knowledge is theory bound, and scientific knowledge is developed to explain the way the world works. (1 p. 2) Engineering knowledge can be considered a distinct form of knowledge since scientific and engineering knowledge aim at different ends. In short, science aims to explain and technology/engineering aims to create artifices. "Technology, though it may *apply* science, is not the same as or entirely *applied* science" (3 p. 4).

Introduction

Technology literacy covers a broad spectrum of topics and constituents, not the least of which are the users, producers, and those becoming producers of technology. The types of knowledge produced, as well as the processes by which knowledge assimilated and utilized are different among these groups of persons. Central to these sociological distinctions and their respective knowledge generation/use is the relationship of different technological disciplines to one another, and how the technological disciplines effect knowledge generation.

This paper examines a particular aspect of technology, computing, and its relationship to related technological disciplines, specifically to 'science' and 'engineering.' Here science and engineering are treated broadly, with the recognition that these disciplines each have long

histories and significant nuances among them. The viewpoints of science and engineering are idealistic and pragmatic respectively (4).

The nature, or goal of science as a whole, is to explain how the world works, as determined through experiments and their artifacts. Knowledge generation, from the perspective of scientists is approached via creating a hypothesis first, and testing it second. These scientific findings and related technologies are often codified, and managed by a body of scientists, and transmitted through published works and education.

The nature of engineering is for addressing human needs/problems, and generally applies domain-specific heuristics for building a system, or components intended for system deployment. The lessons learned and identified technologies are often (though not always) codified, and may be transmitted informally, or formally to other engineers in the form of best practices, or where applicable, via a hypothesis that generalizes the best practice and testing of that hypothesis. In either case, the more formalized engineering findings are often codified, and managed by a body of engineers, and transmitted through published works and education.

Both science and engineering involve the identification of and development of technology, and both include significant relationship to and roots in mathematics. The tested formalizations of theory, identified technology, and best practice can lead to the creation of new branches of engineering and/or science.

Epistemology and knowledge

Briefly stated epistemology is the study of knowledge as justified belief. Development in epistemology since the work of David Hume has focused on identifying the justified true belief about what a single person knows. When applied to disciplines or communities, it also includes the requirement that to qualify as knowledge a proposition or set of propositions must be endorsed by an appropriate community. Hence, there is both an individual, and a significant sociological aspect to knowledge and justified belief. Individuals produce candidate claims for knowledge, and these candidates become knowledge once they are endorsed by the appropriate community using agreed upon standards. (1 p. 2) Components of the *method of the human mind* (5) (Lonergan, 1992) and of Polanyi's *Personal Knowledge* are outlined as foundations for individual understanding, followed by an outline centered on Polanyi, Sorokin and Vanderburg for the sociological approaches to epistemology.

Individual and Communal Knowing Lonergan argues a “method of the human mind,” consisting of *experiencing, understanding,* and *judging*, underlies all specialized forms of knowing. This basic heuristic method or structure is tailored in various ways to meet the particular requirements of specialized contexts such as everyday practical life and the fields of mathematical and empirical science. (5)

In the "method of the human mind," experiencing provides us with data and depends upon adherence to a norm of attentiveness. Our striving to make sense of the data (e.g., “What is it?),

promotes us from experiencing to attempts to understand; to finding the form, pattern, meaning, or significance of what we have experienced. Inquiry and imagination yield insights, which are expressed in concepts and definitions to provide a formulation of the understanding we have attained. Inquiry, insight and formulation embody a norm of intelligence. (Citation?) Because understandings may be misunderstandings, we cannot stop with them but must go on to ask the critical question, "is it really so?" The process of answering this question thematizes our desire to move through critical reflection to judgment. Judging marshals and weighs the evidence to assess the adequacy of our understanding. The evidence is adequate if it shows that the conditions necessary for something's being so are all met. If they are met, within the context our knowing reaches a "virtually unconditioned" state, whose conditions for justified belief are all fulfilled, so that it no longer is conditional and must be so. The norm embodied in these operations of judging is that of reasonableness. The overall method is adjusted based on the perceived need for timeliness, precision, comprehensiveness, universality, and/or completeness. (5)

Thus knowing is both a process, involving both covert and overt human actions, which involve the application of personal and social criteria. These knowledge-generation actions apply to the production of scientific and technological things. Polanyi describes covert actions to include operations such as sensing, perceiving, imagining, understanding, judging, and deciding. The overt actions include operations such as speaking, writing, drawing, calculating, grasping, shaping materials, and using tools. (6)

Individual knowing is a precursor to the community endorsement necessary for knowledge and justified belief to become part of the body of accepted knowledge. Both covert and overt actions have an internal structure that Polanyi calls a *from-to* relation. A skillful achievement, whether practical or theoretical, is the *to-* term of this relation, and the subordinated particulars constitute the from term. He suggests a movement metaphor for this relation when, in discussing the *from-to* relation in acts of knowing. (6 p. 10) Sociological models of the epistemological process by which we create scientific and technological products also embody a movement metaphor. As presented in Table 1, Sorokin breaks the process into three stages, which are used here as an outline of the individual + communal knowledge development process. (7) For technical development, Vanderburg proposes a "technological cycle" with five phases. (8) In both models, the movements are from each earlier stage or phase to the next one.

Table 1. Correspondences between knowledge development stages (7), and phases of technological development (8)

Sorokin 3 Stage Model	Vanderburg 5 Phase Technological Cycle
1. Mental Integration	1. Invention
2. Empirical Objectification	2. Innovation and Development 3. Application
3. Socialization	4. Diffusion
	5. Displacement

This parallels Lonergan's proposals. According to Morelli and Morelli, Lonergan, similarly, "identifies the scheme of recurrence which constitutes technological, economic, and political advance: *situation – insight – communication – persuasion – agreement – decision – action – new situation – insight,*" etc.

Mental Integration: In defining "mental integration," Sorokin states that the "integration of two or more meanings into one system is an act of creation occurring in the human mind." (7 p. 63) This treats it as a covert act "in the human mind." Vanderburg's description of "invention" includes both covert acts, covert states, and overt acts (8 pp. 135-6), although the acts of exploring and working out details are usually overt actions, including actions such as writing, calculating, sketching, building physical models, and conversing with others.

Empirical Objectification: Sorokin's characterization of empirical objectification emphasizes the need for "empirical vehicles through which [new knowledge propositions] can be conveyed to others." (7 p. 64)These can be incarnated through documents, examples, products, or other means. They key point among these processes is the development and presentation to a wider (more than individual) audience.

Socialization implies that the "empirical vehicles" are utilized to share knowledge among individuals, led by agents of socialization. He adds that most of the thousands of texts and artifacts produced never succeed in socialization, in being accepted and used by anybody but their authors. (7 pp. 63-4). The key here is that people are the agents of socialization, and socialization efforts begin early in the process and continue throughout. When the original inventor or discoverer attempts to explain his idea to another person, he is an agent of socialization. When members of an original group attempt to persuade controllers of resources to support their project, market products, etc., they are agents of socialization.

The point of these phases is to recognize that individuals produce candidate claims for knowledge, and these candidates become knowledge once they are endorsed by the appropriate community using agreed upon standards. (9) The importance of the different stages is two-fold: first to recognize the importance of the inner mental state of a single individual, and to understand the difficulties this presents with respect to the certainty with which one can assert that someone actually 'knows' something. Among philosophers, this has led to "devising doomed criteria by which we can determine whether an individual uttering a proposition with X, Y, and Z properties can be said to know something. The criteria are doomed because they ignore contingency, historical and otherwise." (1 p. 1)

A pragmatic view of knowledge, on the other hand, shifts the emphasis to the criteria that the community has devised. But, even here, the criteria must meet some bottom line condition. For the pragmatist the bottom line is successful action. C. I. Lewis put it this way: "the utility of knowledge lies in the control it gives us, through appropriate action, over the quality of our future experience" (10).

Understanding scientific knowledge

Moving from these more generalized forms of knowledge-generation to study those used in engineering and science is not straightforward. Common sense knowing is a form of the general method of the mind (5) where the process is tailored to the requirement for timely action and has less need for great precision, comprehensiveness, universality, or completeness, especially where attempting to meet such standards would only delay action unnecessarily. Common sense is content to find out only what we need to know right now, in this particular context, and to formulate itself allusively and incompletely in rules of thumb, examples, proverbs, and parables. Because its concern is our practical living, it relates things to ourselves rather than to each other, as the mathematical equations of science would do. In this sense, scientific knowing (sciences) develop from common sense and may be more or less distant from it. This distance is critical, and plays a difficult role in the development of a 'new' science. For example, botany, relying on descriptions of plants as they appear to us, is much closer to a common sense method than the current study of plant genetics. Yet, botany is a well-recognized subfield within biology, and both common sense and theoretical approaches are used in its pedagogy.

Lonergan distinguishes mathematical and empirical scientific heuristic structures. He subdivides empirical approaches into classical and statistical types. Classical approaches yield an intrinsic intelligibility and pertain to systematic aspects of reality, marked by schemes of recurrence. Statistical approaches pertain to non-systematic aspects of reality and yield no intrinsic intelligibility but, rather, probabilities, ideal frequencies around which actual events tend to cluster randomly.

The central point is that scientific claims derive their meaning from the theories within which they are associated, hence, *scientific knowledge is theory-bound.* The dynamic process in which scientists continuously revise what they are willing to endorse – and by which they examine their assumptions and their methods – is at the very heart of the strength of the sciences. Thus, despite the theory-bound nature of scientific knowledge, the self-critical process of scientific inquiry insures that the knowledge it claims is the best available at that time insofar as it is judged "best" according to community standards. The ultimate aim of scientific inquiry is explanation. Thus, in the context of a pragmatic account, the ultimate success of the use of scientific knowledge is explanation. (1 pp. 2-3)

Understanding engineering knowledge

Both science and technology may borrow from or rely on each other in various ways – they constitute two distinct forms of knowledge since they aim at different ends. Science aims to explain and technology/engineering aims to create artifices. Vincenti puts it this way, "technology, though it may *apply* science, is not the same as or entirely *applied* science" (3 p. 4)

Engineering refers to the practice of organizing the design and construction and operation of any artifice which transforms the physical or social world around us to meet some recognized need. Engineering – like science – is an *activity* with specific objectives. Consequently *engineering knowledge concerns the design, construction, and operation of artifices for the purpose of*

manipulating the human environment. (1 p. 5) One can reasonably narrow the focus of engineering knowledge to the topic of "design knowledge," by concentrating on design.

"Design" in this context denotes both the content of a set of plans (as in "the design for a new airplane") and the process by which those plans are produced. In the latter meaning, it typically involves tentative layout (or layouts) of the arrangement and dimensions of the artifice, checking of the candidate device by mathematical analysis or experimental test to see if it does the required job, and modification when (as commonly happens at first) it does not. This is a process of refinement, subject to a perceived need for timeliness, precision, comprehensiveness, and/or completeness. As such, this procedure usually requires several iterations before finally dimensioned plans/specifications/requirements can be released for production, and the effort is judged by its results – the usefulness of the design and/or product. (3 p. 7), (1)

Comparing the knowledge generation processes involved, engineering and science appear related, but significantly different, differentiated in both purpose and manner. This follows the distinctions suggested by Lonergan in his observations on empirical science. One of the distinctions between mathematical and empirical science/research is the empirical researcher's return to overlooked or neglected data that "forces the revision of initial viewpoints."

> "The circuit, then, of mathematical development may be named immanent; [as] it moves from images through insights and conceptions to the production of symbolic images whence higher insights arise. But the circuit of [empirical] scientific development includes action upon external things; it moves from observation and experiment to tabulations and graphs, from these to insights and formulations, from formulations to forecasts, from forecasts to operations, in which it obtains fresh evidence either for confirmation or for the revision of existing views." (5 p. 35)

One of the key pragmatic distinctions between this iterative scientific approach and the engineering approach is its goal: The scientist aims at explanation: universal, reliable, comprehensive and sufficiently precise formulation of knowledge; the engineer, aims at timeliness, completeness, with sufficient precision and comprehension.

Pitt provides a similar comparison of scientific and engineering knowledge

> "First, the characterization of scientific knowledge as theory-bound and aiming at explanation appears to be in sharp contrast to the …task-specific knowledge of engineering that aims at the production of an artifact to serve a predetermined purpose.
>
> "The second important difference between the two forms of knowledge [is]…the manner in which engineers solve their problems does have a distinctive aspect. The solution to specific *kinds* of problems ends up cataloged and recorded in the form of reference works which can be employed across engineering areas. For example, measuring material stress has been systematized to a great extent. Depending on the material, how to do it can be found in an appropriate book. This gives rise to the idea that much engineering is "cookbook engineering," but what is forgotten in this caricature is that another part of the necessary knowledge is knowing what book to look for. This a unique form of knowledge that engineers bring to problem solving. (1 pp. 5-6)

The viewpoints of science and engineering are pragmatic and idealistic respectively (4). The nature of science explains how the world works and this can be learned though experiments (producing artifacts). Knowledge generation, from the perspective of scientists is approached via creating a hypothesis first, and testing it second. The nature of engineering is for addressing human needs/problems, and generally applies domain-specific heuristics for building a system. Both involve the identification of and development of technology.

The awkward case of computer science

Computing as a discipline (11 p. 13) is one of the more epistemologically complex disciplines to emerge. Computing is applied in nearly all scientific and engineering disciplines, with significant historical roots in mathematics (not science). The study and development of computing technology historically began in Mathematics and Electrical Engineering, emerging as *Computer Science*. It has been openly argued about the 'scientific' nature of Computer Science - is the discipline a science (12), bad science (13), or not science at all (14).

Computing as… science

The process of identifying a discipline for computer science as a science or natural science has been a challenge and has engendered significant debate. Dijkstra argued that computer scientists do theoretical work like mathematicians, and therefore computing cannot be called as natural science as mathematics. On the flip side, Peter Denning argued that computer science is a natural science in the sense that it studies naturally occurring information processes. (15) Often the pro-science argument is that although computer science might not be a natural science, it is still an empirical or experimental science. Paul Rosenbloom argued that computer science is a new domain which differs from physical science, life science and social science. (4 pp. 363-364) (16)

Colburn presented the analogy relating the scientific method as used in computer science, noting that what is being tested in the scientific method is not the experiment, but the hypothesis. The experiment is a tool for testing the hypothesis. (17) Similarly, what is being tested in problem-solving in computer science is not the program, but the algorithm. This idea finds its roots in an earlier analogy by Kahlil and Levy: "programming is to computer science what the laboratory is to the physical sciences". (18)

Computing as *bad* science

In this area researchers argue that computer scientists publish relatively few papers with experimentally validated results also research reports in computing disciplines rarely include an explanation of research approach in the abstract, keyword, or research report itself, which makes it difficult to analyze how computer scientists arrived at their results

Fletcher criticized the strong adherence to experimental procedure and disagreed with Denning's, Glass's and others' preoccupation with experimentation. He argued that most research in computing is not of the experimental sort "to find best solution to the previous problem," rather that computer scientists work with problems that are poorly understood, and with one major goal

is to understand the problem and delimit it more precisely. (19) Here, the argument suggests that computing is pragmatically oriented, and more like engineering, hence ‘bad’ science.

Computing as... *not* science

The scientific nature of computer science was significantly criticized in the 1990s. McKee argued that computer scientists are not honest about their work and they are “just acting like scientists and not actually doing science”. For instance, Brooks (1996) wrote that computer science is a synthetic, engineering discipline. He also argued about the misnaming of computing as a science. Firstly, it leads computer scientists to accept a pecking order where theory is respected to more than practice. Secondly, it leads them to regard the invention [INCOMPLETE thought]. Thirdly, it leads them to forget the users and their real problems. Fourthly, it directs young and brilliant minds towards theoretical subjects. Among the arguments against the scientific aspects of computing were several calls to rename the discipline. (4 pp. 366-367)

There are an abundance of viewpoints about valid activities, methods, and research approaches in computing fields. While the foundations laid by Lonergan suggest that Computer Science is most closely related to empirical science, the current gamut of topics, subjects, and approaches in computing fields does not fit under any single epistemological or methodological system. That variety is a source of strength and progress (Tedre and Sutinen 2009), but it also exposes the field to critique from all directions, both external and internal. The heated debates about the soul of computing have not ceased. Among all the debates about definition or characterization of computing as a scientific discipline, there has not yet been an argument that would have brought the discussion to closure. (4)

However, these tensions about the soul of computing as a scientific discipline do not come without a cost. “Throughout its history, computing as a discipline has been overshadowed by an identity crisis.” (4 p. 382). The identity crisis for this significant area of technology has led to ambiguity about the proper topic and proper methods of computer science, as well as how the term ‘computer science’ should be operationalized. (4 p. 363). The present need for this operationalization is even more critical, as the operational definition of ‘computer science’ remains central to the definition of Software Engineering as a discipline. (20 p. 82).

Computing Pedagogy: Scientific, Engineering, or Other?

In computer science, problem solving skills and hypothesis testing is different and is more complicated due to its intangible nature. Thus the traditional “analysis-design-technology” trichotomy (21) useful for achieving technology literacy applies differently in learning computer science. The focus of this presentation is how technology literacy (understanding) is differentiated in science and engineering. Technology awareness in computer science is further complicated, as computing is different than other technology fields due to its abstract nature.

In science, experiments based on an analysis of the world embody a design calculated to produce an effect in reality. However, the resulting alteration of reality is not for its own sake but for the

sake of testing and confirming the analysis. In the case of engineering technology, goal of developing and deploying technology is the alteration in reality accomplished – changing reality with the intent of addressing some person's needs (problem solving). For example, engineering draws on a foundation in physics and chemistry to solve design problems, but those sciences will not help *directly* in developing the needed novel artifacts. Stated more formally, engineering knowledge is transformational in that laws and theories are used to solve real problems. Engineering knowledge always aims at success in the product.

Knowledge generation in engineering explains how problems can be solved by performing some series of actions leading to new functionalities. These functionalities are achieved by practicing, designing and solving a set of concrete problems such as those answered by Vincentti's air foil design. (3) If the functional rules are satisfied, then engineers jump into structural design of the artifact where the fundamental laws of physics are applied in order to connect the technical objects with the natural environment.

This process is significantly different in computer science. Like mathematics and logic, it produces little or no alteration of the world, even for the sake of testing. (We may leave to the side Sorokin's empirical objectification discussed above). The 'problem' that computer science deals with is internal to the discipline and usually involves the development of an algorithm - a reusable, mathematical structure that is sufficiently comprehensive and precise. While the algorithm itself may yet be applied to some real problem, and the process of developing the algorithm may also be applied to some problem, the science focuses on the development of the algorithm. Its' application, while motivational, is neither the initial intent nor a necessary part of the process by which it is developed.

The patterns and operations distinguished and studied by computer science often have great heuristic value for solving both scientific and engineering problems. The algorithms become, like differential equations, part of the tool-kit of the scientist seeking to understand and of the engineer intending to analyze, design and transform reality. However, computer science is closer to science than to engineering because its aim is a comprehensive understanding of certain logical relations rather than a transformation of reality. Indeed, it is even farther from engineering than the physical sciences are, because its concern is less with the concrete world than with a world of symbolic structures and constructs. In this regard, it is closer to mathematics and symbolic logic. On the other hand, computer science is, and in practice computer science education (CSE) is often more closely linked to technology, to computers, than either logic or mathematics or other empirical sciences are. The aim of the science, and of learning the technology is about algorithms and the symbolic structures and constructs that manifest them.

In contrast, in software engineering, the aim is entirely different. There we seek the application of the algorithm in a sufficiently timely, sufficiently complete (satisficing) manner to produce an artifact and a desired change in reality. Given this fundamental difference in goals, 'computer science', at least as taught, will look more like science than engineering, and only in its more

pragmatic areas will its methods be more engineering-like. This distinction suggests that computer science, and especially CSE is more like mathematics and science - a foundation for engineering, rather than branch of engineering.

Devon and Ollis have proposed seven dimensions to the effective learning of technology. Among these dimensions, three are particularly useful for distinguishing knowledge generation in engineering from that in science: “for whom it works”, “its deployment: market value and other measure of value”, “the tradeoffs: strongest and weakest features (what the critiques say)” (21). Because such concerns are not typical of CSE, they help us to differentiate computer science from computer engineering (11). Computer science is more a pure than an applied science and so works for itself more than for external purposes or clients. Computer engineering, however, serves other masters: it determines how to meet needs and purposes that are external to it and set by others. Similarly, the deployment of computer science occurs in journal articles, books, conferences and classes, in which it is the focus. Computer engineering, on the other hand, is deployed in response to external needs and its value is generally considered instrumental, derived from the value of its products. Even in its teaching, real world problems are constantly posed and solved. In regard to trade-offs, typically such problems arise in engineering in addressing real world concerns involving time and economy in the use of materials. These are, of course, external factors, whereas, in computer science, while trade-offs of simplicity and precision may be possible, they are internal to the process of doing computer science.

The pragmatic theory of knowledge provides a means by which, particularly in a learning environment, the critical terms of ‘science’, and ‘engineering’ can be reliably distinguished. This paper has outlined some of these distinctions, and applied them briefly to the challenges of

Bibliography

1. *What Engineers Know*. **Pitt, Joseph C.** 3, Fall 2007, Techné: Research in Philosophy and Technology, Vol. 5.

2. **Durbin, Paul T.** Multiple Facets of Philosophy and Engineering. [ed.] Ibo and Goldberg, David E. Ibo van de Poel. *Philosophy and Engineering, An Emerging Agenda.* s.l. : Springer, 2010, 4, pp. 41-47.

3. **Vincente, Walter.** *What Engineers Know and How They Know It.* Baltimore : Johns Hopkins University Press, 1990.

4. *Survey of Viewpoints.* **Tedre, Matti.** 21, s.l. : Springer, 2011, Minds and Mission, pp. 361-387.

5. **Lonergan, Bernard.** *Insight: A Study of Human Understanding. Collected Works.* [ed.] Robert M. Doran, S.J. and Frederick E. Crowe, S.J. 5th. Toronto : University of Toronto Press, 1992. p. 810. Vol. 3.

6. **Polanyi, M.** *Personal Knowledge: Toward a Post-Critical Philosophy.* Chicago : University of Chicago Press, 1958.

7. **Sorokin, P.** *Social and Cultural Dynamics.* New York : American Book Company, 1941. Vol. IV.

8. **Vanderburg, W.** *Living in the Labyrinth of Technology.* Toronto : University of Toronto Press, 2005.

9. **Pitt, Joseph C.** *Thinking about Techonlogy: Foundations of the Philosophy of Technology.* s.l. : Seven Bridges Pr Llc, 1999.

10. **Lewis, C. I.** *Analysis of knowledge and valuation.* La Salle : Open Court, 1946.

11. **Force, Interim Review Task.** *Computer Science Curriculum 2008, An Interim Revision of CS 2001.* s.l. : Association for Computing Machinery and IEEE Computer Society, 2008.

12. *Computer Scvience Revisited.* **Cerf, Vinton G.** 12, December 2012, Communications of the ACM, Vol. 55, p. 7.

13. *Research in Information Systems: An empirical study of diversity in the discipline and its journals.* **Vessey, I., Ramesh, G., and Glass, R. L.** 2, 2002, Journal of Management Information Systems, Vol. 19, pp. 129-174. As quoted in "Computing as Science, a survey of Competing Viewpoints".

14. *The computer scientist as toolsmith II.* **Brooks, F. P. Jr.** 3, March 1996, Communications of the ACM, Vol. 39, pp. 61-68.

15. *Computing is a Natural Science.* **Denning, Peter.** 7, July 2007, Communications of the ACM, Vol. 50, pp. 13-18.

16. *A New Framework for Computer Science and Engineering.* **Rosenbloom, Paul.** 2004, IEEE Computer, pp. 31-36.

17. **Colburn, T. R.** *Philosophy and Computer Science.* Armonk, NY : M. E. Sharpe, 2000.

18. **Kahlil, H. Levy and L. S.** The Academic Image of Computer Science. *ACM SIGCSE Bulliten.* 1978, Vol. 10, 2, pp. 31-33.

19. **Fletcher, P.** The role of experiments in computer science. *Journal of Systems and Software.* 1995, Vol. 30, 1-2, pp. 161-163.

20. *Computer Science: Is It Really the Scientific Foundation for Software Engineering?* **Frezza, Stephen T.** 2010, IEEE Computer Society , pp. 82-85.

21. *Technology Literacy for the Technologically Literate.* **Devon, Richard and Ollis, David.** 2007. Proceedings of the ASEE.

22. *Computer Science:Is it Really the Scientific Foundation for Software Engineering?* **Frezza, Stephen.** s.l. : IEEE Computer Society, August 2010, Computer, pp. 82-85.

23. *Know Your Discipline: Teaching the Philosophy of Computer Science.* **Tedre, Matti.** 2007, Journal of Information Technology Education, Vol. 6, pp. 105-122.

24. *Survey of Viewpoints.* **Tedre, Matti.** 21, s.l. : Springer, 2011, Minds and Mission, pp. 361-387.