Philosophies **2016**, *1*, 28–39; doi:10.3390/philosophies1010028

OPEN ACCESS

philosophies

ISSN 2409-9287

www.mdpi.com/journal/philosophies

Article

Machine Code and Metaphysics: A Perspective on Software Engineering

Lindsay Smith *, Vito Veneziano and Paul Wernick

School of Computer Science, University of Hertfordshire, Hatfield, AL10 9AB, UK;
E-Mails: v.veneziano@herts.ac.uk (V.V.); p.d.wernick@herts.ac.uk (P.W.)

* Author to whom correspondence should be addressed; E-Mail: l.1.Smith@herts.ac.uk;
Tel.: +44-1707-284-367.

Academic Editor: Vincent C. Müller

Received: 21 August 2015 / Accepted: 30 October 2015 / Published: 10 November 2015

Abstract: A major, but too-little-considered problem for Software Engineering (SE) is a lack of consensus concerning Computer Science (CS) and how this relates to developing unpredictable computing technology. We consider some implications for SE of computer systems differing scientific basis, exemplified with the International Standard Organisations Open Systems Interconnection (ISO-OSI) layered architectural model. An architectural view allows comparison of computing technology components facilitating a view of computing as a continuum. For example, at one layer of computer architecture, components written in Turing-complete machine language can be seen as deterministic and consistent with a theoretical paradigm of CS. At another layer, components (applications) closer to the human sphere have been seen as non-deterministic and inconsistent with theoretical CS. We compare unpredictable development of computing technology against the cyclic legacy of technological advance and scientific discovery, and suggest that SE indicates an enabling cycle, discernible in previous scientific revolution(s), is stalled or possibly hidden. The CS consequence of divorcing technological advance from scientific consensus is particularly concerning. For example human/computing events could be seen as unpredictable virtual phenomena that *somehow* extend the ontology of CS. Our approach challenges practical and philosophical boundaries by investigating if applying scientific method (SM) resolves any SE/Science dichotomy.

Keywords: scientific method; computer science; software engineering; computing; continuum; determinist; non-deterministic

1. Introduction

In this paper, we try to address three initial fundamental "technical" ("Technical" as in programming and, more in general, as in all techniques software engineers adopt in their practice) issues in software engineering (SE) that occur when computing technology is developed.

The first issue is about establishing **what** SE problems are. Then a second issue we aim at exploring is **how** critical, when building computing technology, problem solving is for SE. This, in turn, leads into a third issue: **why** it is possible for SE to be building less predictable technology than other engineering disciplines. These questions lead into a discussion on the epistemological and metaphysical status of SE in relation to the artefacts built, e.g. software.

In Section 2, we provide some more substantial grounding to the analysis of the above issues.

In Section 3, we introduce our ideas for a more empirically based approach to computer science (CS). We outline computing as having a "nature" which is occurring within SE processes and in the "architecture" of the technological outcomes. This discussion explores a connection we see between SE the "nature" of computing and a lack of consensus in what CS is (Eden, 2007) [1]. Some constraints on adopting an empirical approach, such as the viability of applying scientific method (SM) and hypothesis testing in CS, are outlined, put into a wider context and further discussed in the fourth and last section of this paper.

2. A "Technical" Problem for Software Engineering

In general, implementing computer systems can change human environments unpredictably, e.g., the role of social media in the so called "Arab spring". In particular for software development projects end-user-related problems can be a major failure factor.

What SE problems look and feel like for stakeholders in software development is evidenced by an example taken from a large UK publicly funded project the "NHS (National Health Service) National Programme for Information technology (IT) called NPfIT, subsequently re-named Connecting for Health" [2]. Anecdotal and research evidence on the failures of this particular project are widely reported on in the computing industry press. Although reports tend to concentrate on commercial concerns, e.g. project management issues, one particular factor in the NHS project failure concerned end-users of a computerised medical records system. Specifically these end-users, who had the power to refuse to use the resulting technology, became a factor in the NHS project failure. End-users such as individual doctors in General Practice (GPs) "who struggled to use a computer screen" [2] to access medical records found this interfered with patient consultations. It was acknowledged this resulted in GP surgeries "de-railing" "plans for online patient records by 2015" [2].

Similar examples are plentiful in SE [3,4] and give some indication of the likelihood of SE projects, producing unusable technology. As Maughan states, "few projects can succeed over the outright opposition of the proposed users" [2]. This state of affairs in SE has raised some "fundamental questions" concerning what SE is [5–7]. We are raising an "elephant in the room" type problem

(Figure 1) [8] for SE because, as in the NHS example, computing technology problems can become indeterminate (A computing industry press campaign to stop the NHS project was raised due to the impossibility of achieving a successful project completion.). Typical end-user problems with computer systems not being technically robust, reliable, usable, cannot not be anticipated if it is not possible for SE practitioners to determine what is deliverable in cases such as the NHS project.

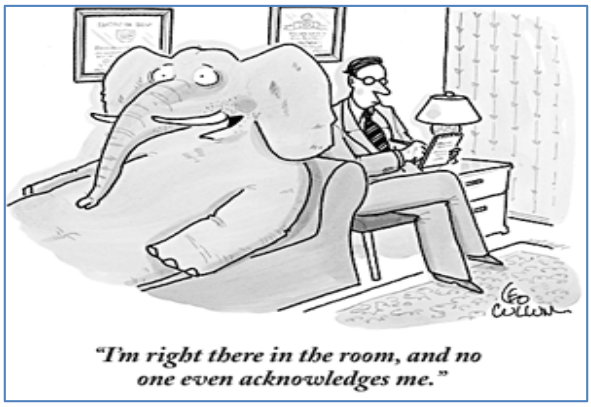

Figure 1. Elephant in the room [8].

Certainly this type of scenario for SE projects runs alongside a continuing lack of consensus on how to define CS. For example (Eden, 2007) [1] sets out three definitions of CS, "as a branch of mathematics (e.g., [9]), as an engineering or 'technological' discipline and as a natural science" [1]. **How** problem solving in SE extends beyond a theoretical (deterministic) CS paradigm (An original starting point for computing.) or **how** compatible that is with a natural science paradigm are 'elephant in the room' questions we are asking about SE.

The example (Figure 2) [10] of computer machine architecture is used to consider a deterministic/non-deterministic dichotomy in computing. Differing claims, to a scientific base for computing are illustrated using the model's architectural "layers". Developing computing technology such as machine code which is deterministic is consistent with a theoretical paradigm underlying CS, e.g., machine code can be proven to be Turing-complete [1]. However, layers closer to the human sphere of operation, e.g., user programs, are not consistent with a purely theoretical CS paradigm [11]. This is because interaction within human social contexts, e.g., an NHS GP needing to access medical records, is non-deterministic.

It is by combining the above NHS example with (Figure 2) we are placing SE within a specific technical deterministic/non-deterministic dichotomy, e.g., when building computing systems. Acknowledging computing has a distinct 'nature' explains **how** problems in SE present compared to other type of engineering, e.g., civil engineering. For instance a computer (or Turing machine) can process information which at the arrow labelled A in (Figure 2) between layers 5–6 is unusable. Such as a computerised medical records system [2] interfering with a GP's patient consultations. However at the arrow labelled B between layers 0–1 a GP's machine is deterministic and is functioning correctly even with an instance of higher level failures. The implications are lower layers in the technology are analogous to a Wittgensteinian "calculus" [TLP 5.53–5.532] [12]. At this level the architecture can be proven *a priori* to compute correctly [13], and be insensitive to any "contingent" [12] context of the technology, e.g., in a human social environment. In effect SE practitioners are working in an

"epistemological gap" [14] building artefacts not completely understood, e.g., not fully underpinned by either a theoretical or empirical science of computing. The above discussion illustrates **how** deterministic solutions to non-deterministic human problems have a high risk probability in SE. We consider next **how** this makes SE more unpredictable than other types of engineering.

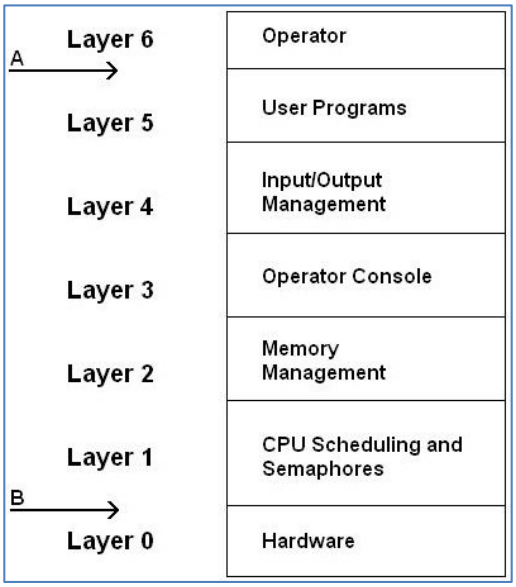

Figure 2. Computer architecture (based on Zimmermann, 1980 [15]).

The above computerising medical records scenario is similar to building a technically functioning bridge that nevertheless, people couldn't use to cross over because it was too difficult. The London millennium bridge is a well-understood example of a civil engineering project that went wrong. The bridge became impossible cross once it was opened to pedestrians as the momentum of people walking caused the bridge to sway (wobble) from side to side (like a ship in heavy seas) [16].

The technical perfection of a (millennium) bridge is as irrelevant to "ordinary" people as the correctness of the internal state of a computer system. The contingent, or "everyday", aspects of technology are much more significant to people when deciding if it is worth using a bridge to cross a river or a computer to access medical records. The intentions of computer, or bridge, users are not compatible with formal proof(s) of correctness. Maybe this could explain why computing included technological advances such as supplementing 0–1 layer machine architecture with the additional layers 2–6 in Figure 2?

One difference between the NHS project failure and the millennium bridge was the later was fixed now the bridge is used successfully. Highlighting a particular failure factor belies the fact the NHS project failure was due to multiple factors. The point is SE questions concerning **why** building computing technology can be so much more unpredictable, than bridge building, are likely to be complex and need multi factorial answers. Building computer systems spans a wide knowledge domain from end-user context to software developer understanding of the architecture, in Figure 2 or the many other versions [17] of the technology. The question of **why** SE projects fail we are locating in the need to bridge the gap between lower, deterministic, layers and higher, non-deterministic, layers in computing technology.

Our proposal includes testing such claims by undertaking some practical work. This includes attempting to identify some possible boundaries for SE to clarify the situation. Envisaging computing technology and its development as occupying a continuum (Figure 3) is a starting point for this work. It a strategic move to avoid extreme positions such as defining CS *exclusively* as theoretical and concerned with abstract objects and SE as *purely* "technological" [1]. Our analysis adopts a minimal position so makes no claims as to what exists, e.g., where science stops and technological development begins

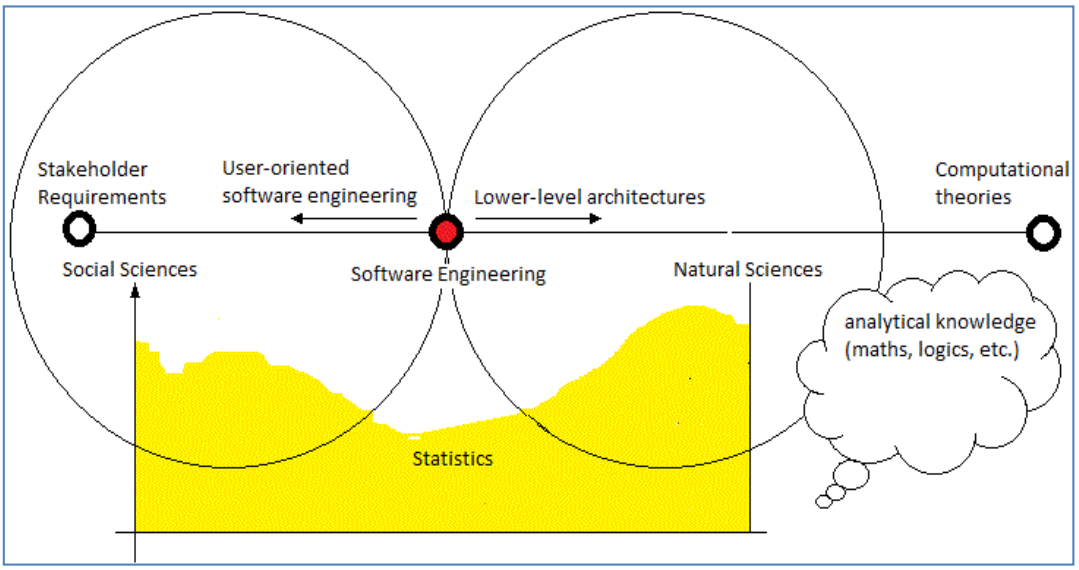

Figure 3. Computing occurs as continuum.

We are looking at a position for SE in a computing continuum (Figure 3), which in turn is facilitating a wider scientific perspective on CS. In Figure 3 it is possible to depict the findings of social science(s) as more relevant for SE, than the natural science(s), when eliciting stakeholder requirements. However where to place the boundaries for such SE activities across this spectrum cannot be clearly defined scientifically. For example Figure 3 is depicting the lower availability of SE statistical results in comparison to the social or natural science, which serves as an indication of a lack of scientific evidence [18] in computing.

What Figure 3 is illustrating is a lack of clarity as to what computing is and the lack of consensus in CS reflects this too. The development of computing technology is like a "puzzle" built into the architecture of the technology see Figure 2. One consequence of this "puzzle" is SE occurring in an "epistemological gap" [14] a claim introduced in the above discussion and expanded upon further in the next section.

To conclude we assert the concept of a universal Turing machine supplies information processing power, but provides no insight into how that information is perceived in a human context. What occurs on the computing continuum Figure 3 is often veiled in (metaphysical?) mystery.

3. Classical Scientific Revolution & Technological Advance

In this section, we further consider the epistemological status of computing whilst establishing some general constraints for our approach (as introduced in Section 2). In order to do so, we would

consider the compatibility of science (in general) with computing (in particular) by referring to natural and social science examples for guidance.

The analysis we are reporting here was originally triggered by practical work in SE undertaken previously. The aim of the analysis was to establish if applying scientific method (SM) in computing is compromised due to an 'epistemological gap' [14]. We want to expand upon a particular result of this analysis which is CS may have an exceptional relationship with SE. The relationship is exceptional in the sense of being removed from the classical cyclical relationship between scientific discovery in nature and human technological advance.

Defining what is scientific in general introduces a series of problems. What we are concerned with is whether CS proves any exception to a chain of dependencies in scientific discovery which established previous scientific revolutions. For example scientific discovery can be a prerequisite for further technological change, which in turn enables other scientific advances. Scientific knowledge is however provisional and not often complete as seen in development of modern theories of electricity. This occurred, typically piecemeal, via experiments with magnetism [19] whilst the connection between lightening and magnetism was not made [19].

Certainly computing depends on past technological advances such as electricity. Computing is merely a collection of unconnected technology, e.g., silicon, wire and plastic if the electricity fails. In this respect computer technology fits in with the chain of dependencies that make scientific and technological advance a case of 'wheels within wheels'.

Crude generalising about science is not particularly helpful, but it does at least confirm how CS could be more complicated than it first appears [20]. A lack of agreement on what CS is serves to obscure or even obstruct answers to specific questions [20]. This is a problem for CS as any scientific advance must gain acceptance, at least provisionally [21], in a scientific community. Such acceptance depends on things such as scientific method (SM) which "may be described as the logical scheme used by scientists" [22] and is subject area dependent, e.g., social or natural science disciplines.

We consider, for CS, if the above SM type cycle is broken, or hidden, due to the distinct way the architecture of computing technology does not occur naturally. Computing artefacts are different from other engineered technology due to originating in theoretical (a priori) CS. In discussing the origins, in Physics, of electrical engineering we are talking about an example of a technological advance based in natural science which has a different epistemological starting point to CS. We accept this is the case because the physical world could be perceived to exist (ontologically) prior to the knowledge (epistemology) provided by theoretical physics. Whilst what we are defining (in Section 2) as computing is, at least partially, something that could *not* have been perceived as existing (ontologically) prior to CS providing the (a priori) theoretical knowledge (epistemology) enabling computing machines to be built.

Or to put the case more simply you can exist in a physical world without physicists but computers cannot exist (with you in that physical world) without computer scientists. The technology being built has distinct, not physically, but artificially occurring origins. It is this which gives SE an unusual, in comparison to other types of engineering, position in relation to the respective science. We decided to try, if possible, to move forward by analysing specific epistemological issues for CS with features of science (in general), e.g., SM.

Combining theoretical and empirical scientific paradigms would allow for the "hypotetico-deductive cycle" [22] being accepted in CS based "…on the results obtained through logical reasoning, observations and/or experiments" [22] needed for SM.

The unusual position of SE in relation to CS (as discussed above) means the usual, e.g. natural science, distinction between technology and science is blurred which has implications for the compatibility of SM with CS or SE. The predictive facility of SM would certainly be helpful in SE for example when developers need to anticipate problems with requirements engineering (RE) and stakeholders in IT projects. There are particular widely acknowledged problems for RE [23] concerning the accuracy of requirements for example. Those practitioners who are actually developing computing technology certainly have a vested interest in RE managing to successfully predict requirement outcomes.

It is premature to conclude practitioner experience could lead to possibilities for advancing CS. There are disparities such as [18] observes in SE research which lacks application of SM. This challenges of notion of a generally applicable SM [22] particularly in the case of SE. Nevertheless addressing specific RE problems do appear to provide us a means to progress investigating the compatibility of SM with SE. It is potentially RE which is satisfying a condition of bringing into existence, via requirements, human/computer events which are being experienced (a posteriori), e.g., once computer systems are used. We are testing the compatibility of SM with computing (see Figure 3) and non-deterministic events in the physical world.

The ontological status of SE is a critical part of deciding the compatibility of SM. We consider if *somehow* CS must extend ontologically to incorporate SE in order to make it reasonable to consider SM as a means to advance knowledge in SE. Our concern here is with ontological questions, or what *exists* for SE or what is actually "out there" in SE?

We introduced in Section 2 the example of an "architectural model" (Figure 2) which also illustrates the ontology of SE as a type of puzzle due to the artificial origins of computing technology. Machine code, metaphorically speaking, appears around the arrow B layer (Figure 2). Thus arrow B points to a solvable part of the puzzle in the architecture as a proof can exist, e.g., Turing complete [1] machine code can be written. However such types of pieces in this puzzle appear isolated. The rest of the puzzle is not solvable in the same way, e.g., an equivalent proof cannot exist. What is possible to observe, as discussed above, is an "epistemological gap" [14] in this case between where formal proof exists and where it does not.

Any answer concerning what exists in SE we suggest is possible at architecturally lower levels, e.g., machine code components. It is apparent that the lower the level in the architecture the higher the level of certainty, of existence (proof) (Existence for such components is the proof of correctness.). So isolated parts of this particular puzzle can be "solved", however no clear picture (theory) can be used to deduce what exists for other parts of the puzzle, e.g., layers 6–5. Thus certainty as to what exits for SE starts to drop off for the higher layers in the architecture. This means questions concerning existence for SE cannot be answered purely deductively (a priori) so inductive (a posteriori) answers are also needed. This makes proceeding inductively and employing empirical science a reasonable way to identify what exists for SE at higher architectural layers.

It is far from clear what perceiving is in SE, hence the discussion does not rule out the possibility of a dichotomy between SE and science (in general) due to problems with the compatibility of SM and

CS. This is because cycling between deductions (a priori) and inductions (a posteriori) to test hypothesis, build theories, *etc*. is required in SM. Adopting an empirical approach to observing any phenomena also requires establishing the feasibly of observation. In this case any attempt to identify relevant observable phenomena in SE concerns software development. Elements of a software system such as program code, specifications and test results [24] also raise questions concerning the ontology of software, e.g., what exists for software. This particular subject is not directly addressed here with any specific claims other than what is necessary for our purposes. Though generally we accept all SE products/processes reflect possible underlying CS phenomena. In the Kantian sense this entails human sensory perception which, in modern scientific practice, is routinely enhanced by artificial means of observation. Our investigation into the compatibly of CS and SM, inevitably, employs artificial means of observation to test the ideas we introduce (hypothesis) of human/computing events existing as "virtual phenomena" (Figure 4). We imply human/computing events can be perceived to exist for SE only if proven sufficiently observable. Thus human/computing events become possible CS phenomena susceptible to SM.

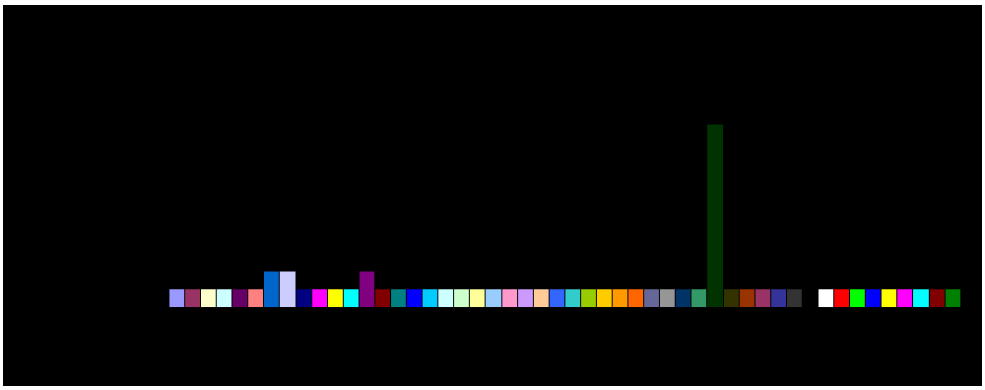

Figure 4. Hypothesis: Human/computing events as virtual phenomena?

An example of some results of such practical work, a graphical representation of web page activity stored in server log files (Figure 4) is acting as an introduction into the next and final section of the paper. This discussion is reporting on an analysis that took place whilst undertaking some practical work in software development, data collection, analysis, *etc*. We have included suggesting some new concepts such as hybrid human/computing events to address the possibility of observing SE phenomena as part of a computing "virtual world".

The practical work aims to resolve the dichotomy, we identified between CS and other sciences (in particular the natural sciences), via investigating the role of SE in this dichotomy. However establishing observation in the virtual world, for other instances see [25], becomes more fraught the closer to the machine you get. For example in Figure 4 a graph depicts some logged web server events for web page components with under a second 'life span' when the hosted website was navigated. Server log files track human/computer behaviour in an unnatural and thus very different dimension of existence. Determining what it is practical to observe becomes an immediate issue. For example does the page component in Figure 4 that had 10 logged events indicate website navigation or some server-side activity superfluous to human/computer behaviour?

4. SE/Science Dichotomy: Is Computing Technology Development Independent from Scientific Discovery?

This section is comparing some established philosophical/scientific background with the discussion in Sections 2 and 3. The discussion here is consolidating the approach we are taking by placing it in a wider context. For example in this section we are reflecting if claiming SE could be "unscientific" due to the artificial origins of the technology being built has consistency with other established work defining what it is to be scientific.

We are not however being critical or evaluating an exhaustive literature review. It is rather we are selecting some obvious (In this case cliché is good.), or as seminal as possible references for cross checking purposes looking at any common ground between such work and our analysis. We are employing a "sounding board" for our empirical approach to investigating SE, e.g., the practical work example in Figure 4.

Aligning our work with a pluralist view of science acknowledges diverse forms of "scientific facts" which differ across alternative scientific paradigms [26] and is countering any highly restricting definition of CS (as discussed in Section 2). However this does not solve the problem (identified in Section 2) with a lack of consensus about what CS is because for scientific progress, according to Kuhn, is acceptance amongst peers is critical. Something we cannot claim to have and can only (optimistically) hope to achieve by results in our empirical practical work.

Our claim to be observing "virtual worlds" (see Section 2, Figure 4) is not currently accepted but it is consistent with disciplines other than CS seeking to adapt SM to differing contexts. In the social sciences SM has been evolving (and differentiates) from natural sciences. In cognitive psychology, for example, as Gigerenzer says "discovery heuristics may be of interest […] for an *a posteriori* understanding of theory development" [27].

5. Conclusions

Adopting and/or adapting SM to gain an understanding of the role of dichotomy in SE and computing (Figure 3) seems reasonable. This results in a need to find a minimal criterion for applying SM successfully in the case of SE. We decided the ability to use (*a posteriori*) observations to develop theories capable of predicting outcomes is a minimal testable criteria. Weighing up the **positives** this has similarities with the evidence in the social sciences and reflects a pluralistic approach.

However cross checking between our work and established thought on the role and place of scientific investigation gives some of the **negatives** for adopting this approach. The boundaries have been clearly marked out between opposing philosophical positions. For example, Hacker makes a critique of Quine calling him "the metaphysician of twentieth-century science" [28] due to Quine's logical positive position. Hacker critiques Quine for believing that "extending the term "science" […] will not make history and the social sciences any more like physics and chemistry than they are […] not very" [28].

This particular debate serves to illustrate that anything, including SE, need not amenable be SM. There are fundamental challenges when applying SM to SE which are same as for applying SM for any discipline. In particular being able to identify observable phenomena is critical. Suggesting SE is

building the "virtually empirical" (see Figure 4 above) potentially makes SE compatible with SM but this needs to be tempered by an awareness of the historical challenges faced by other sciences. Our hypothesis that the "virtually empirical" is observable has some limitations in referring to the Kantian concept of phenomena [29]. It is not at all certain SE is producing what that can be perceived for our purposes. What cannot be ruled out is the opposite in Kant's Critique of Pure Reason SE could be transcendental or in elemental form, as Kantian noumenon [30], and thus not observable. If computing technology replicates or propagates in this unobservable kind there is an insoluble problem with applying SM. As (Hacker, 2006) [28] argues in this case including SE in an empirical scientific investigation couldn't be conceivable.

However our adopting a "virtually empirical" hypothesis (Section 3, Figure 4), has some resonance with 'discovery heuristics' [27] and scientific provenance with for example of the discovery of electricity [19]. Advances in the natural sciences depended on connections being made between apparently unrelated observations, e.g., lightening and electricity. It is interesting if not conclusive for CS, that observations apparently unrelated by the current state of knowledge may still have a connection. Implying sciences have a multi-disciplinary resonance on the basis of limited examples is only hinting at some apparent evidence for SM of common ground. However apparently unconnected observations proved crucial in past scientific discovery and it is not possible, on this basis, to rule out such possibilities for observation of a virtual world. For software developers the words of Mendelssohn on past scientific endeavour will have resonance with software testing today "However discouraging the results might be, they only stimulated more curiosity, and the scientists simply went on experimenting until the results began to make sense" [19]. The challenges of understanding the physical world in the past were no less profound than those presenting now for a digital future (virtual) world.

The questions we are attempting to answer concerns the future-proofing SM. We are assessing the adaptability of SM originally refined for gaining understanding of a physical world. Our minimal conclusion is, we are identifying a specific issue concerning the adaptability of SM, and assessing the relevance of this for hypothesis testing in empirical practical work on SE.

Conflicts of Interest

The authors declare no conflict of interest.

References

1. Eden, A.H. Three Paradigms of Computer Science. *Minds Mach.* **2007**, *17*, 135–167.
2. Maughan, A. Six reasons why the NHS National Programme for IT failed. 2010. Available online: http://www.computerweekly.com/opinion/Six-reasons-why-the-NHS-National-Programme-for-IT-failed (accessed on 30 July 2014).
3. Keil, M.; Tiwana, A.; Bush, A. Reconciling user and project manager perceptions of IT project risk: A Delphi study. *Inform. Syst. J.* **2002**, *12*, 103–119.
4. Cheney, P.H.; Mann, R.I.; Amoroso, D.L. Organizational factors affecting the success of end-user computing. *J. Manag. Inform. Syst.* **1986**, *3*, 65–80.

5. Malkawi, M.I. The art of software systems development: Reliability, Availability, Maintainability, Performance (RAMP). *HCIS* **2013**, *3*, 1–17.
6. Reitano, V. Software Engineering: Art or Science. 8 November 2011. Available online: http://sdt.bz/content/article.aspx?ArticleID=36088&page=1 (accessed on 31 July 2014).
7. Loka, R.R. Software development: What is the problem? *IEEE Computer* **2007**, *40*, 110–112.
8. Jackson, M. How to Talk about the Elephant in the Room. 2013. Available online: http://mitchjackson.com/white-elephants/ (accessed on 20 August 2015).
9. Knuth, D.E. *The Art of Computer Programming, Vol. I: Fundamental Algorithms*; Addison Wesley: Reading, MA, USA, 1968.
10. Computer architecture. Available online: http://cs-exhibitions.uni-klu.ac.at/index.php?id=4&uid=20 (accessed on 20 August 2015).
11. Newell, A.; Simon, H.A. Computer science as empirical inquiry: Symbols and search. *Comm. ACM.* **1976**, *19*, 113–126.
12. Wittgenstein, L. Tractatus Logico-Philosophicus. *Annalen der Naturphilosophie* **1921**, *14*, 185–262.
13. Dijkstra, E.W. The Structure of the "THE"—Multiprogramming System. *Comm. ACM*. **1983**, *26*, 49–52.
14. Smith, L.; Wernick, P.; Veneziano V. Is finding a Black Swan (Popper, 1936) possible in software development? In Proceedings of the First International Conference of the IACAP, Aarhus, Denmark, 4–6 July 2011.
15. Zimmermann, H. OSI Reference Model—The ISO Model of Architecture for Open Systems Interconnection. *IEEE Trans Comm.* **1980**, *28*, 425–432.
16. BBC News. Millennium Bridge. Available online: http://news.bbc.co.uk/hi/english/static/in_depth/uk/2000/millennium_bridge/default.stm (accessed on 20 August 2015).
17. ISO. Systems and software engineering—Architecture description. *IEEE*. **2011**, *4*, 2010.
18. Kitchenham, B.; Al-Khilidar, H.; Babar, M.A.; Berry, M.; Cox, K.; Keung, J.; Zhu, L. Evaluating guidelines for reporting empirical software engineering studies. *Empir. Software Eng*. **2007**, *13*, 97–121.
19. Mendelssohm, K. *Science and Western Domination*; Thames and Hudson: London, UK, 1976.
20. Dijkstra, E.W. The end of Computing Science? *Comm. ACM* **2001**, *44*, 92.
21. Popper, K. *Conjectures and Refutations: The Growth of Scientific Knowledge*; Routledge: London, UK, 1963.
22. Dodig-Crnkovic, G. Shifting the paradigm of Philosophy of Science: Philosophy of Informaiton and a New Renaissance. *Minds Mach.* **2003**, *13*, 521–536.
23. Nuseibeh, B.; Easterbrook, S. Requirements engineering: A roadmap. In Proceedings of the Conference on the Future of Software Engineering, Limerick, Ireland, 4–11 June 2000.
24. Lehman, M.M.; Ramil, J.F.; Wernick, P.D.; Perry, D.E.; Turski, W.M. Metrics and laws of software evolution-the nineties view. In Proceedings of the fourth International Software Metrics Symposium, Albuquerque, NM, USA, 5–7 November 1997.
25. Minocha, S.; Tran, M.; Reeves, A.J. Conducting Empirical Research in 3D Virtual Worlds: Experiences from two projects in Second Life. *JVWR* **2010**, *3*, 1–21.

26. Kuhn, T.S. *The Structure of Scientific Revolutions*; University of Chicago press: Chicago, IL, USA, 2012.
27. Gigerenzer, G. From tools to theories: A heuristic of discovery in cognitive psychology. *Psychol. Rev.* **1991**, *98*, 254–267.
28. Hacker, P. Passing by the naturalistic turn: On Quine's cul-de-sac. *Philosophy* **2006**, *81*, 231–253.
29. Oxford University Press. Phenomenon: n. Oxford English Dictionary OED. Available online: http://www.oed.com/view/Entry/128690?redirectedFrom=noumenon#eid (accessed 20 August 2015).
30. Oxford University Press. Noumenon: n. Oxford English Dictionary OED. Available online: http://www.oed.com/view/Entry/142352?redirectedFrom=phenomenon#eid (accessed 20 August 2015).