View metadata, citation and similar papers at core.ac.uk

brought to you by CORE

provided by Birkbeck Institutional Research Online

Epistemic Opacity, Confirmation Holism and Technical Debt: Computer Simulation in the light of Empirical Software Engineering

Julian Newman
Birkbeck College, University of London

The output of a simulation model does not, prima facie, appear to have an objective status comparable with data captured by observation or experiment. Counter to this Winsberg, Humphreys and others emphasise parallels between experiment and simulation in practices which are said to "carry with them their own credentials" (Winsberg, 2010). Humphreys holds that that the physical implementation of computer models places constraints on simulation methods not present in traditional mathematics, creating essential epistemic opacity. By this he means that it is impossible for a cognitive agent, given its characteristics, to know all of the epistemically relevant elements of a computational process. Humphreys views essential epistemic opacity as reflecting the limitations not of the simulation method itself but of the human agent, and thus as evidence for a "non-anthropocentric epistemology" recognising computational tools as a superior epistemic authority.

The possibility of testing a highly parameterised simulation model via the hypothetico-deductive method can indeed be open to doubt; moreover empirical measurements are often not available on the scale needed to evaluate model outputs. Even were appropriate data available, Lenhard & Winsberg (2010) argue that climate simulation models face epistemological challenges associated with a novel kind of "confirmation holism": it is impossible to locate the sources of the failure of any complex simulation to match known data, so that it must stand or fall as a whole. This is because of three interrelated characteristics which they regard as intrinsic to the practice of complex systems modelling – "fuzzy modularity", "kludging" and "generative entrenchment".

- In "fuzzy modularity", different modules simulating different parts of the complex system are in continual interaction, thus it is difficult to define clean interfaces between the components of the model.
- A kludge is an inelegant, 'botched together' piece of program, very complex, unprincipled in its design, ill-understood, hard to prove complete or sound and therefore having unknown limitations, and hard to maintain or extend.
- Generative entrenchment refers to the historical inheritance of hard-to-alter features from predecessor models.

Is confirmation holism

(1) essential to and unavoidable in complex systems modelling?

(2) embedded in specific disciplinary practices of climate science?

or

(3) does it exemplify a failure to observe, recognise and apply available and well-established sound Software Engineering practices when developing simulation software (as promoted, for example, by the Sustainable Software Institute)?

Belief in the essential epistemic opacity of computational science points to (1) but we shall argue for (3) on two main grounds.

- Firstly, where a large software system is epistemically opaque with respect to a human agent, this opacity is not an essential characteristic arising from its size but is contingent upon development practices and in particular upon architectural design. A software architecture captures basic design decisions which address such issues as performance, reliability, security, maintainability and interoperation. Decomposition into manageable components is an essential architectural strategy for managing complexity. Failure to perform such decomposition at the design stage will certainly make the software itself epistemically opaque, but it would be perverse to regard this as endowing the results with superior authority or the capacity to carry its own credentials.
- Secondly, credulity towards simulation software runs entirely counter to the institutionalised local scepticism common to both experimental science and software engineering practice.

The Engineering Science oriented towards underpinning and evaluating Software Engineering practice is widely known as “Empirical Software Engineering”. An important subject of recent empirical research is that of “Architectural Defects” in large software systems. As remarked above, a kludge is code which is ill-understood and (therefore) hard to maintain: it is a software defect waiting to manifest itself. A study of defects in a long-lived large software system, which had undergone multiple versions and releases, found that defects which span more than one software component required more than 20 times as many changes to correct, compared to single-component defects, and were 6 to 8 times more likely to persist from one release to another (Li et al, 2011). A further line of research concerns the consequences of making early programming decisions on a purely pragmatic basis (e.g. in order to get the system working); it has been shown that such short cuts create “Technical Debt” on which interest will accrue in the form of error and maintenance costs throughout the lifecycle of the software product (Kruchten et al, 2012).

In software engineering practice, defects are expected: human activity is error-prone. Yet well architected software is not epistemically opaque: its modular structure will facilitate reduction of initial errors, recognition and correction of those errors that are perpetrated, and later systematic integration of new software components. Nothing intrinsic to complex simulation modelling prevents the application of these principles, but kludging in the early stages of model building will create “Technical Debt” which will be charged in the form of actual (not essential) epistemic opacity. Simulation software is epistemically opaque (when it is) not because of the inability of human agents to check through every possible execution path from beginning to end (an impossible feat, but unnecessary in well architected software), but because of a failure of model builders to adopt the practices which are known to promote surveyability and effective error management. A simulation model must be understood as a tool which can play a part, along with other resources, in a scientific argument; such an argument depends upon human judgement which, fallible though it may be, cannot legitimately be replaced by an allegedly superior epistemic authority. The argument from the essential epistemic opacity of computational science to a non-anthropocentric epistemology runs counter to best practice in software engineering and to empirical results of software engineering science. In this respect it is self-defeating.

References

P. Kruchten, R.L. Nord, I. Ozkaya “Technical debt: from metaphor to theory and practice”, IEEE Software, v. 29 n. 6, 2012, pp. 18-21.

P. Humphreys "The Philosophical Novelty of Computer Simulation Methods", Synthèse, v. 169, 2009, pp. 615-626.

Z. Li et al., "Characteristics of Multiple-component Defects and Architectural Hotspots: A large system case study" Empirical Software Engineering v. 16, 2011, pp. 667-702.

J. Lenhard, E. Winsberg, "Holism, entrenchment and the future of climate model pluralism", Studies Hist. Phil. Mod. Physics, v. 41, 2010, pp. 253-263.

E. Winsberg, Science in the Age of Computer Simulation, Chicago University Press, 2010.