Viewpoints from Computing to the Epistemology of Experiments

John Pajunen[1] and Matti Tedre[2] and Nella Moisseinen[3]
[1] Dept. of Social Sciences and Philosophy, University of Jyväskylä, Finland
[2] Stockholm University, DSV, Sweden
[3] Dept. of Computer Science and Information Systems, University of Jyväskylä, Finland
[1] firstname.lastname@jyu.fi
[2] firstname.lastname@acm.org
[3] firstname.a.lastname@student.jyu.fi

Abstract

Although experiments have been a core element of the scientific method since the 1600s, experiments per se only caught philosophers' interest in the 1980s. Since the 1980s dozens of philosophical analyses of experiments have been presented, based mostly on physics and biology. A number of philosophers of science have called for bottom-up, 'naturalistic' investigations of experiments in various disciplines, especially fields other than physics. This paper presents an epistemological analysis of experiments in computing fields in terms of epistemological characteristics, research milieux, and epistemological features of results. Our analysis of experiments, based on how the term is operationalized in computer science papers, opens new critical viewpoints to the role of experiments in computing, as well as complementary viewpoints to the concept of experiment in the philosophy of science.

1 Introduction

The Aristotelian tradition tested ideas through observation and deduction—ideas, which dominated discussions about scientific investigation until the 1600 (Arabatzis, 2008) despite various critical commentaries. It was in the 1600s when experimentation started to gradually gain ground in scientific practice. Francis Bacon proposed several ideas concerning the still undetermined role of experiments (Gower, 1997). However, aside from fleeting passages pointing towards the need for the philosophy of experiment, little serious analysis can be found in the philosophy of science literature before the 1980s when a number of philosophers called for increased understanding of the role of the experiment in scientific practice (e.g., Hacking, 1983).

In the next decade, much of the sprung interest lost its momentum (Radder, 2003); the philosophy of experiments has yet developed steadily in the past 30 years, as witnessed by numerous articles and monographs on the topic. Despite the advances in the philosophy of experiment, there are frequent calls for increased investigation on the epistemology of experiments. Radder (2003:p.1) lamented the fact that even though many scientists—perhaps even most of them—spend most of their time with experiments, that "*is not reflected in the basic literature in the philosophy of science*." Hacking (1983), on the other hand, stated that "*no field in the philosophy of science is more systematically neglected than experiment*." Hon (2003) argued that no forceful and cohesive treatment of the experiment could yet be found in the philosophy literature.

The new turn towards the epistemology of experiments has been driven by 'naturalistic', bottom-up, grassroots or shop floor-level accounts of what really goes on in the laboratory. But while there are a plethora of arguments based on physics, there are fewer arguments based on other academic disciplines. This article presents an account of experiments in computing fields and analyzes the epistemological dimensions of those experiments.

2 Experiments in Computing

Computing as a discipline is a combination of three 'irrevocably intertwined' traditions—the theoretical, empirical, and engineering tradition (e.g., Denning, 1989). Those three traditions give rise to unique crossbreeds between theoretical, technical, and empirical activities, and accordingly, a tripartite analysis of computing offers a unique window to the epistemology of experiments. Our previous work identified five distinct realizations of experimentation in computing: the demonstration experiment, the trial experiment, the field experiment, the comparison experiment, and the controlled experiment; these are summarized in this section.

The Demonstration Experiment. The first, and loosest, use of the word 'experiment' is found in texts that report and describe new techniques and tools. In those texts, it is not known if task *t* can be automated efficiently, reliably, feasibly, cost-efficiently, or by meeting some other criterion of success. A demonstration of experimental (novel, untested, newly implemented) technology shows that it can indeed be done. Feitelson (2006) described such work as "demonstration of feasibility" type of experimental computer science. Basili & Zelkowitz (2007) called it "more an existence proof" than what they considered to be a real experiment. For an example of experimentation in computing, Plaice (1995) proposed development of large software systems. In his Turing Award lecture, Hartmanis (1994) urged computer scientists to accept the defining role of demonstrations in experimental work.

The Trial Experiment. The second, also common use of the word 'experiment' is found in texts that evaluate and report various aspects of a computing system using a pre-defined set of variables. The trial experiment requires a set of expectations (specifications) against which the system is evaluated. Typically, in those studies, it is not known how well a new system *s* meets its specifications, how well it performs, or how well it models another system. A trial (or test, or experiment) is designed to evaluate (or test, or experiment with) the qualities of the system *s*. Those tests are typically simulated or laboratory-based, and hence limited in various ways.

The central documents in experimental computer science discuss experiments in the 'trial' sense of the word. For example, McCracken et al. (1979) argued that experimental computer science should not stop at feasibility demonstrations, but should also measure and test the constructed systems. Gustedt et al. (2009) introduced emulation, simulation, and benchmarking as experimental methodologies in computing (see also Amigoni et al., 2009). Each of them aims at testing the qualities of the system, while they differ on the 'toy-versus-real' dimensions (e.g., Fenton et al., 1994). Glass (1995) and Fletcher (1995) defended the place of experiments in theoretical computer science, too.

The Field Experiment. The third type of experiment commonly found in computing literature, the field experiment, is similar to the trial experiment, but with the computing system taken out of the laboratory. In field experiments, it is not known how well the system meets its requirements in its intended technical and social environment of use. The system is tested, in a live environment, for attributes like robustness, reliability, performance under real-life conditions, productivity, or usability attributes. Field experiments offer less control than trial experiments, thus limiting reproducibility and generalizability of their results. However, field experiments can involve various controls for

experimental variables, so they can offer more control than many other common field research methods, such as surveys or case studies do (Palvia et al., 2003).

In information systems research 'field experiment' is common (e.g., Palvia et al., 2003), while 'in-situ experiments' are found in large-scale systems (Gustedt et al., 2009). Freeman (2008) called them "*experimentation under real-world conditions*", and used the DARPA robot car race as an example. Field experiments can be used to evaluate models, such as systematically measuring the errors between response-time estimates given by a queuing network model and the real response times in a live system (Denning, 1980).

Comparison Experiment. The fourth common use of the word 'experiment' in computing refers to controlled comparison between competing solutions. In reports of those studies, it is not known if (or rather, 'not yet shown that') one solution for a given task is better than another. The criteria for 'better' are often things like average execution time, the average memory footprint, or accuracy of output. An experiment is set up to measure and compare the two solutions, and the report shows that with some inputs and parameters, the proposed system beats its competition.

The comparison experiment naturally fits those branches of computing research that are concerned with looking for the 'best' solution for a specific problem (e.g., Fletcher, 1995). Johnson (2002) called that type of experimental analysis "horse race papers". While comparison experiments are reproducible, they are still susceptible to bias in numerous ways (Feitelson, 2006; Fletcher, 1995). Yet, many fields of computing have introduced standard tests, input data, and expected outputs, against which results can be compared.

Controlled Experiment. A fifth common use of the term 'experiment' refers to the controlled experiment: an experiment that tests association between variables, tests the validity of a hypothesis, or determines whether an intervention makes a difference between the experimental (treated) group and control (untreated) group or comparison (alternative treatment). The controlled experiment is often seen as the gold standard of scientific research—especially in natural sciences—and it typically enables generalization and prediction.

In the experimental computer science debates, the term 'experimental' is often meant to be read 'based on controlled experiments'. Peisert and Bishop (2007) advocated controlled experiments for research on computer security. Morrison and Snodgrass (2011) wanted to see more generalizable results in software development. Feitelson (2007) promoted evaluations under controlled conditions for all applied computer science. For instance, one can study the effect that monitor screen size may have to productivity of users by randomly dividing users into groups with small monitors and large monitors and having both do the same tasks.

3 Epistemological Aspects of Experiments in Computing

The view above can be analyzed from various viewpoints, looking at things like aims, typical methods, or modes of justification. As one of the most fundamental aims of experimentation is increased knowledge, this view takes an epistemological viewpoint towards those five experiment realizations. Epistemology deals with questions such as 'what is knowledge,' 'how is knowledge acquired,' and 'what kinds of characteristics and types of knowledge are there?' Typical characteristics of knowledge involve, for example, certainty, consistency, coherence, usefulness, reliability, and validity. For the purposes of this paper, we take that *scientific knowledge* involves data, models, theories, and procedural knowledge.

The first section below is concerned with typical questions, typical points of reference, and typical epistemological criteria involved in the five realizations of experimentation above. The second section discusses research milieux, and the third section below is concerned with epistemic features of results in the five realizations of experiment in computing discussed above.

3.1 Epistemological Characteristics of Experiments in Computing

In computing fields the realization of experiment as 'theory testing' is increasing in popularity. A rare curiosity in the 1980s, the advocates of 'experimental computer science' have popularized the view that computer scientists should increasingly aim at theory generation and experiment-based testing. However, among the five realizations of experiment in computing, 'theory-testing' is best compatible with only the 'controlled experiment'. All other realizations of experiment evaluate their results against something other than theory or hypothesis. Table 1 presents examples of typical epistemological characteristics for each experiment realization. However, it must be noted that none of the 'realizations' are well-defined, mutually exclusive categories: many 'typical' elements can be found in other 'realizations' too.

Table 1: Typical Questions, Points of Reference, and Epistemological Criteria for the Five Experiment Realizations

Experiment Realization	***Typical Question***	***Typical Point of Reference***	***Typical Epistemological Criteria***
Demonstration experiment	Does *o* exist? Can *a* exist?	Success criteria; an instantiation of a model	Existence or feasibility proof of an artifact / property / relation
Trial experiment	To what extent does system *s* meet specifications *S* in a restricted environment?	Specifications; system's internal properties	The system has properties $p_1 \ldots p_n$
Field experiment	How well does system *s* meet requirements *R* in an open environment?	Requirements; do the internal properties satisfy externally defined values	The system meets requirements *R* (to some specified extent)
Comparison experiment	Given criteria *C*, which solution fares best?	Competing systems with respect to externally set values	Rank ordering of solutions $s_1 \ldots s_n$ according to criteria *C*
Controlled experiment	Does the prediction *p* hold? Are variables *V* associated?	Hypothesis or theory; results of an experiment	Fit between theory and observations; Corroboration / falsification

In Table 1, points of reference range from *success criteria* (such as effectiveness, proof of feasibility) to *specifications* (low-level requirements on system behavior) to *requirements* (high-level standards on the system in its environment) to *competing solutions* (selected characteristics between the proposed system and its competing systems) to *hypotheses and theories*.

3.2 Research Milieux

Research milieu is a combination of various elements—most significantly the research environment and research subjects. Environment plays an important role in characterizing experiments: simulated experiments, lab experiments, and "in nature" or "in situ"-experiments place the experiment in very different frameworks. The issue is not the location as such, but rather the qualities of the location *with respect to epistemic goals*. When the goal is to isolate a cause of a phenomenon, then the fewer possible or plausible causes have to be eliminated the better. When the goal is to measure the robustness of an application in its intended environment of use, then the ultimate standard is, naturally, to introduce the application in real use. The environmental boundaries

are not clear-cut, because there are a large variety of intermediate environments (e.g., Gustedt et al., 2009).

Another aspect of epistemology of experiments arises from the ontology of experimental subjects. It is commonplace that the ideal of a controlled lab experiment comes from the natural sciences, and the view on the nature of the object under study is that there is an underlying cause in the material world that a scientist wants to explore. As long as the objects studied are ontologically objective—such as semiconductors and electromagnetic radiation—research designs from natural science are easily justified. Cue in humans and intentionality, and the picture changes.

Unlike the physical sciences, which often aim at establishing causal laws or increasing the precision at which constants are measured, research on human participants—especially studies on the behavioral and social levels—does not always do that. On the one hand, there is a lot of research on physiological, and, on the other, the mental and social aspects, where one can establish statistical but not causal laws. Research on subjective phenomena or qualia is not always aimed at establishing generalizable results at all. Ontologically subjective things, like aesthetic preferences, social norms, and learned conventions are a part and parcel of computing research, like it or not.

Table 2: Example Combinations of Three Ontological Frameworks and Four Research Milieux

	Real application	*Model application*
Lab environment	*Physical*: How high throughput does our social networking server software have on test equipment with simulated data?	*Physical*: Which algorithms for our social networking service produce the best throughput with mock data under laboratory conditions?
	Mental: How well does the release version of the social software score in standard usability tests and criteria?	*Mental*: How well do our informants respond to the prototype version's usability in usability lab?
	Social: To what extent do test users agree with our social networking program's suggestions of 'people of interest'?	*Social*: Which set of basic conventions for social interaction work best with several groups of test informants on a prototype platform?
Real environment	*Physical*: What are the average latencies of the deployed system in real use situations?	*Physical*: What does the automatic crash data tell about the beta version's stability?
	Mental: How often do users add the system's recommended friends as their friends?	*Mental*: Do beta-testers understand the program's icons similarly?
	Social: What kinds of social functions does the 'like' button develop over time?	*Social*: What are the best algorithms for predicting how well do people 'match'?

The subject matter relates to epistemology of experiments in that a researcher faces new kinds of challenges when humans enter the picture. When dealing with material objects, repetition of an experiment can be done to *confirm* that a result was not just a fluke, but the same is not as straightforward with human subjects, since each subject is unique and each test changes the subject. More generally, it is the case that mental and social phenomena are not isolatable in the same manner as material objects are. Different experimental set ups are needed for different subjects of study. In Table 2 physical, mental, and social aspects of ontology are exemplified by questions pertaining to four research milieux: lab / 'real' environments vs. toy / 'real' applications.

3.3 Epistemological Features of Results

Franklin (1990, p.104) presents nine "epistemological strategies" that are used to support the validity of experiments: experimental checks and calibration, reproducing artifacts by experiment, experiment-based intervention, independent confirmation, elimination of errors, establishing validity, explaining results, using apparatus, and using statistical arguments. Franklin does not claim that these strategies – that he exemplifies by cases from natural science – are exclusive or exhaustive (1989, p. 190). In a similar way to Franklins approach, we take it that strategies to validate experimental results in computing include feasibility, criteria conformance, contextual fit, relative merits, and generalizability and predictive power (Table 3).

Table 3: Validation Strategies of Experiments

Experiment Realization	***Validation strategy***
Demonstration experiment	"*Secure knowledge*" (McCarthy, 2006): A successful demonstration is enough to establish feasibility (i.e., "It can be done").
Trial experiment	*Criteria conformance*: The more previously defined specifications the system meets, the 'better' it is deemed.
Field experiment	*Contextual fit*: The extent to which the system meets its requirements in its intended sociotechnical context of use, the 'better' it is deemed.
Comparison experiment	*Relative merits*: The system's behavior in comparison with the competing systems defines its ranking among the competition.
Controlled experiment	*Generalizability and predictive power*: The higher the statistical significance, the more plausible the findings.

Each experiment realization produces different kinds of knowledge, and hence, they are evaluated against different kinds of criteria and they are used in different ways for justifying the experimental approach. *Demonstration* is a special case, as the knowledge it produces is often procedural, not propositional: While knowledge in natural sciences is tentative by nature—a sort of 'best explanation at the moment'—procedural engineering knowledge is different. A revolution in science shows that the previous explanation was wrong; for example, the æther theory of light was shown to be wrong. Technology and procedural knowledge do not become obsolete because they never worked in the first place (they did), but because there are newer alternatives that do the same thing better in some way. Hence, the term "secure knowledge" (McCarthy, 2006).

The trial experiment judges its findings by how well the results from a trial run fit a set of previously defined specifications. The results are typically quantitative. The field experiment evaluates how well the installed system (or a model of it) meets its requirements in the intended sociotechnical context of use. Both qualitative and quantitative measures are used. The comparison experiment judges the results by two or more system's merits relative to each other: the results are often rank-ordered lists of variables. Finally, the results of controlled experiments are judged by statistical criteria—level of generalizability, validity, or some of the many alternatives.

4 Conclusions

An analysis of the five types of experiment in computing reveals a different picture than similar analyses in physics and biology have done. Firstly, although scientific instruments are central to all the three disciplines, in computing the value of technology is not only instrumental. In addition to the theory of computing, large branches of computing focus on design, construction, testing, and

development of systems: this gives many branches of computing an engineering flavor. Secondly, the role of theory is different in computing and natural sciences: while in natural sciences theories are often used for making predictions and hypotheses, in computing theoretical computer science rarely plays that role—instead, different branches have different theory bases. Thirdly, unlike natural sciences, computing also intersects the social and behavioral sciences. Computational systems are not used in sociocultural void, but the human aspects of systems are an integral part of computing research. Given the field's diversity, it is unsurprising that in computing experimentation terminology lives a life of its own.

The five experiment realizations, as found in computing literature, differ in fundamental ways from each other and from the typical examples found in philosophical literature. While the 'demonstration' experiment is relatively common in computing, it is often criticized in computing literature. But 'existence proof' types of demonstrations are not unknown in other fields, either. For instance, a proof of existence of the Higgs Boson would be a major achievement in physics. The trial and field experiments, which commonly use specifications and requirements as a reference point, are discussed in the philosophy of experiment, although often from viewpoints of the subject matter and not from technological viewpoint. The comparison experiment is not unknown to the philosophy of experiment in reference to both theory and apparatus. The controlled experiment, or 'theory testing' type of experiment is relatively rare in computing, although there is a strong movement that advocates it.

A deeper analysis of the epistemology of experiment in computing can be beneficial in two ways. Firstly, it has been natural to bring methodology from the old and established disciplines like physics into computing research, but it should not be assumed that the "borrowed" methodology is *all* that can be used in computing. Insofar as computing is a unique, independent discipline, philosophy (and methodology) of computing—or of computer science—should be sensitive to that uniqueness, and should not exclude methods and views unsuitable for physics or other natural sciences. A deeper analysis of experiments can expand computing's disciplinary self-understanding.

Secondly, the discussion around experiments in the philosophy of science is excessively about experiments in natural sciences. Such image might not be easily generalizable to other sciences, and it ignores special features of other sciences. Hence, viewpoints from special sciences like computing bring richness and depth to the philosophy of experiment and the philosophy of science—take, for instance, the physical, mental, and social aspects of experimentation, as well as experiments and modeling.

Acknowledgments

This research was partly funded by The Academy of Finland grant #132572.

Bibliography

F. Amigoni, M. Reggiani, and V. Schiaffonati. An insightful comparison between experiments in mobile robotics and in science. *Autonomous Robots*, 27(4):313–325, 2009.

T. Arabatzis. Experiment. In S. Psillos and M. Curd, editors, *The Routledge Companion to Philosophy of Science*. Routledge, Abingdon, UK, 2008.

V. R. Basili and M. V. Zelkowitz. Empirical studies to build a science of computer science. *Communications of the ACM*, 50(11):33–37, November 2007.

P. J. Denning. ACM president's letter: What is experimental computer science? *Communications of the ACM*, 23(10):543–544, 1980.
P. J. Denning, D. E. Comer, D. Gries, M. C. Mulder, A. Tucker, A. J. Turner, and P. R. Young. Computing as a discipline. *Communications of the ACM*, 32(1):9–23, 1989.
D. G. Feitelson. Experimental computer science: The need for a cultural change. Unpublished Manuscript, December 3, 2006, 2006.
N. Fenton, S. L. Pfleeger, and R. L. Glass. Science and substance: A challenge to software engineers. *IEEE Software*, 11(4):86–95, 1994.
P. Fletcher. The role of experiments in computer science. *Journal of Systems and Software*, 30(1–2):161–163, 1995.
A. Franklin. *Experiment, Right or Wrong*. Cambridge University Press, Cambridge, UK, 1990.
A.Franklin. *The neglect of experiment*. Cambridge University Press, Cambridge, UK, 1989.
P. A. Freeman. Back to experimentation. *Communications of the ACM*, 51(1):21–22, 2008.
R. L. Glass. A structure-based critique of contemporary computing research. *Journal of Systems and Software*, 28(1):3–7, 1995.
B. Gower. *Scientific Method: An Historical and Philosophical Introduction*. Routledge, Abingdon, UK, 1997.
J. Gustedt, E. Jeannot, and M. Quinson. Experimental methodologies for large-scale systems: A survey. *Parallel Processing Letters*, 19(3):399–418, 2009.
I. Hacking. Experimentation and scientific realism. *Philosophical Topics*, 13(1):71–87, 1983.
J. Hartmanis. Turing award lecture on computational complexity and the nature of computer science. *Communications of the ACM*, 37(10):37–43, 1994.
G. Hon. An attempt at a philosophy of experiment. In M. C. Galavotti, editor, *Observation and Experiment in the Natural and Social Sciences*, pages 259–284. Kluwer Academic Publishers, New York / Boston / Dordrecht / London / Moscow, 2003.
D. S. Johnson. A theoretician's guide to the experimental analysis of algorithms. In M. H. Goldwasser, D. S. Johnson, and C. C. McGeoch, editors, *Data Structures, Near Neighbor Searches, and Methodology: Fifth and Sixth DIMACS Implementation Challenges*, volume 59 of *DIMACS Series in Discrete Mathematics and Theoretical Computer Science*, pages 215–250. American Mathematical Society, Providence, Rhode Island, USA, 2002.
N. McCarthy. Philosophy in the making. *Ingenia*, 26:47–51, March 2006.
D. D. McCracken, P. J. Denning, and D. H. Brandin. An ACM executive committee position on the crisis in experimental computer science. *Communications of the ACM*, 22(9):503–504, 1979.
C. T. Morrison and R. T. Snodgrass. Computer science can use more science. *Communications of the ACM*, 54(6), 2011.
P. Palvia, E. Mao, A. F. Salam, and K. S. Soliman. Management information systems research: What's there in a methodology? *Communications of the Association for Information Systems*, 11(16):1–32, 2003.
S. Peisert and M. Bishop. I am a scientist, not a philosopher! *IEEE Security and Privacy*, 5(4):48–51, 2007.
J. Plaice. Computer science is an experimental science. *ACM Computing Surveys*, 27(1):33, 1995.
H. Radder. Toward a more developed philosophy of scientific experimentation. In H. Radder, editor, *The Philosophy of Scientific Experimentation*, pages 1–18. University of Pittsburgh Press, Pittsburgh, PA, USA, 2003.