Stretching the Traditional Notion of Experiment in Computing: Explorative Experiments

Viola Schiaffonati[1]

Abstract Experimentation represents today a 'hot' topic in computing. If experiments made with the support of computers, such as computer simulations, have received increasing attention from philosophers of science and technology, questions such as "what does it mean to do experiments in computer science and engineering and what are their benefits?" emerged only recently as central in the debate over the disciplinary status of the discipline. In this work we aim at showing, also by means of paradigmatic examples, how the traditional notion of controlled experiment should be revised to take into account a part of the experimental practice in computing along the lines of experimentation as exploration. Taking inspiration from the discussion on exploratory experimentation in the philosophy of science—experimentation that is not theory-driven—we advance the idea of explorative experiments that, although not new, can contribute to enlarge the debate about the nature and role of experimental methods in computing. In order to further refine this concept we recast explorative experiments as socio-technical experiments, that test new technologies in their socio-technical contexts. We suggest that, when experiments are explorative, control should be intended in a posteriori form, in opposition to the a priori form that usually takes place in traditional experimental contexts.

Keywords Experiment · Computing · Experimental control · Explorative experiment

Received: 23 October 2014 / Accepted: 14 May 2015 / Published online: 28 May 2015

✉ Viola Schiaffonati
viola.schiaffonati@polimi.it

[1] Dipartimento di Elettronica, Informazione e Bioingegneria, Politecnico di Milano, Piazza Leonardo da Vinci 32, 20133 Milan, Italy

Experiment and Computing: Not Only Simulations

Experimentation represents today a ‘hot’ topic in computing. Not only experiments made with the support of computers, such as computer simulations, play an essential role in every domain of science and technology, but also questions such as “what does it mean to make experiments in computer science and engineering and what are their benefits?” have recently taken a center stage in the debate over the disciplinary status of the discipline and its methodological accounts. However, if the experimental capabilities of computer simulations have received increasing attention from philosophers of science, the same attention has not been devoted to analyze how the notion of experiment should be defined in the realm of computing. Thus, our aim in this work is to show that an extension of the traditional notion of controlled experiment is necessary to give reason for different experimental practices in computing.

It is undoubtedly evident that science has entered what has been called the ‘age of computer simulation’ (Winsberg 2010); the massive use of computer simulations in virtually every domain of science has drawn attention to their epistemological justification: if computer simulations started as tools to build tractable models for solving the equations provided by theories, nowadays their role expanded and, besides dealing with the construction of models of greater and greater complexity, computer simulations can be used to increase the exploration opportunities. This is in accordance with the idea of ‘modeling from above’ (Keller 2002) and ‘modeling from the ground up’ (McLeod and Nersessian 2013), where the theoretical model behind the simulation is under construction and shaped by the simulation results themselves. Accordingly, a different relation between theory and experiment emerges, where the latter actively participates in the settling of the former, instead of aiming only at testing or rejecting the theory itself.

Recently the experimental properties of computer simulations have been examined, and philosophers have begun to consider in what sense, if any, computer simulations are experiments (see Winsberg 2013 for a detailed analysis of this debate). Positions range from a full acceptance of the identity thesis (“Computer simulation studies are literally instances of experiments”) to its rejection in different degrees. Computer simulations can be *used* as experiments in the case in which the purposes of simulation and those of experiment coincide, but they are not necessarily experiments: it is perfectly plausible to have computer simulations at work that do not have any experimental purpose (think for example of simulations adopted for didactical purposes).

The possibility of using simulations as experiments resides in the ability (and in the necessity) they both possess of *controlling* the features under investigation, thus implementing the original idea of experiment as a *controlled experience*. The source of credibility for the models used in the different cases of simulations as experiments greatly varies: if the case in which the credentials are provided by the theoretical ancestors, on which simulations are based, is not problematic, the ‘modeling from above’ or ‘from the ground up’ practices require an epistemological justification for their use. Indeed the use of computer simulations also in those fields that do not have secure theoretical foundations and/or in which data are sparse has reshaped the way experimental results are considered reliable. In these contexts experiments (in the form of computer simulations) cannot be considered as pure *controlled experiences* anymore, but are better conceived as *explorations*, where the theoretical background is shaped by simulation results.

Let us move now from experiments made *with* computing (simulations) to experiments made *in* computing: a closer look on whether and to what extent the traditional experimental protocol can be applied to computing provides evidence that the same explorative approach used in computer simulations is already at work in this context, even if it is not yet properly conceptualized. Our goal in this paper is to show how the traditional notion of experiment should be revised to take into account a part of the experimental practice in computing along the lines of experimentation as exploration. Taking inspiration from the discussion on exploratory experimentation in the philosophy of science—experimentation that is not theory-driven—we advance the idea of explorative experiments that, although not new, can contribute to enlarge the debate about the nature and role of experimental methods in computing. In order to further refine this concept we recast explorative experiments as socio-technical experiments, that test new technologies in their socio-technical contexts. We suggest that, when experiments are explorative, control should be intended in a posteriori form, in opposition to the a priori form that usually takes place in traditional experimental contexts.

Before we embark on our discussion, however, we need to provide a working definition for the two core concepts on which this paper relies, namely those of *explorative experiment* and *a posteriori control*. These initial definitions will be expanded and further clarified, mostly with the help of examples, throughout this paper. In general an experiment is a set of observations and actions, performed in a controlled context, to test theories and to provide the basis for scientific knowledge (Hacking 1983; Franklin 1986; Radder 2003). Control, intended as an active manipulation of the phenomena under investigation where the choice of the factors to be controlled is critical, is usually considered a central feature of experimentation so that experiments are also labelled ‘controlled experiments’. In this paper by explorative experiments we mean a form of investigation of novel and interesting ideas or techniques without the typical constraints of rigorous experimental methodologies. These are experiments that are driven by the desire of investigating the realm of possibilities pertaining to the functioning of an artefact and its interaction with the environment in the absence of a proper theory or theoretical background. So hypotheses cannot be clearly stated and, even if the ultimate goal is to acquire knowledge about the performance of the artefacts under investigation and to find out proper concepts to formulate possible regularities, the experimenter is not in full control of the experimental setting due to the impossibility of anticipating all the plausible outcomes. Therefore, when experiments are explorative, control should be intended in *a posteriori form*, in opposition to the a priori form of the traditional experimental contexts. If in the latter experimental factors are in control of the experimenter in a sort of anticipation of the scenario to be tested, in the former the possibility of full anticipation disappears and control is in part carried out after the artefact has been inserted into society.

In this paper, after discussing how the notion of experiment has been differently conceived in computer science and engineering in the last 40 years ("Experimenting in Computing" section), reasons and examples to move beyond the traditional idea of controlled experiment in the direction of exploration are presented ("Some Reasons (and Examples) that Stretch the Traditional Categories" section). Then, the concept of explorative experiment is defined and discussed ("Toward Explorative Experiments" section), together with the idea of a posteriori control, while reflecting on the possible extension of a posteriori control to social experiments intended as an introduction of new technologies in society ("Explorative Experiments, Social Experiments and a Posteriori Control" section). Finally, some concluding remarks are advanced ("Conclusion" section).

Experimenting in Computing

A trend has recently emerged toward making the experimental scientific method take center stage in computer science and engineering, also as a way for reflecting on the disciplinary status of this discipline in between science and technology (see for instance Denning 2013; Schiaffonati and Verdicchio 2014; Hatleback and Spring 2014). The call for a more rigorous experimental practice puts attention on numerous questions: from the dispute on the name (should this discipline be called a science or not?) to the investigation of the sciences of the artificial, including the debate on whether and how traditional experimental principles (control, comparison, repeatability, reproducibility, etc.) could be applied to computing.

Probably one of the first and most famous concepts of computer science as an experimental science goes back to the 1976 paper by Allan Newell and Herbert Simon published in the occasion of their acceptance of the Turing award: "Computer science is an empirical discipline. We would have called it an experimental science, but like astronomy, economics, and geology, some of its unique forms of observation and experience do not fit a narrow stereotype of the experimental method. None the less, they are experiments. Each new machine that is built is an experiment. Actually constructing the machine poses a question to nature; and we listen for the answer by observing the machine in operation and analyzing it by all analytical and measurement means available" (Newell and Simon 1976, 114). This conception of machines and programs as experiments has been influential for many years, promoting the idea that the appeal to experience is fundamental in contrast with the view of computer science as a pure mathematical and deductive discipline. However, the rather ingenuous view of experiments, without any more specific reference to some principles of experimentation, may have contributed to spread an oversimplified conception of how the experimental method can be applied to computing.

The quest for experiments in computing began to be treated systematically at the beginning of the 1980s, following a crisis in what was then called *experimental computer science*. In a report of the Association for Computing Machinery (ACM) published in 1979 (Feldman and Sutherland 1979), experimental research in computer science is strongly related to the measurement and testing of computing algorithms and systems. In the same issue of the journal (McCracken et al. 1979) where the ACM report was published, the call for experimentation is expressed in terms of the recognition of the possibility for major advantages in different fields of computing. At the same time, a 'rejuvenation' of experimental computer science is advocated from very concrete perspectives: for example, by promoting experimental facilities for computer systems research. However, experimental computer science is seldom defined in a precise way in this context, and experiments are conceived mainly as explorations. Experimental computer science is to be rejuvenated also according to Peter Denning, who proposes in a short article that the experimental work produced in computer science should be judged by *traditional* standards: "Let us employ traditional measures when assessing experimental computer science. Let us always have a clear plan for testing a clear hypothesis. Let us not call "hacking" science. These are the criteria by which the rest of the world will evaluate our field's experimental work. If we do not live up to the traditional standards of science, there will come a time when no one takes us seriously" (Denning 1980, 543). Denning advances the idea that to implement experimentally a computer system is not just to build the system and "see what happens". In a way, this approach tries to go beyond the 'construct and test' paradigm of Newell and Simon, by proposing that experimental computer science has to deal with the process of supporting and testing a hypothesis, thus making computing closer to the standards of rigor and the practice found in the progression of the traditional sciences.

Although some efforts along this direction have been put to work over the years (Tichy 1998; Langley 1988), the above-mentioned guidelines, whether reductive or not, remained mostly unattended. More recently, a trend has once again emerged toward making the experimental scientific method take center stage in computing. These recent efforts in several projects have shown a renewed need for an experimental methodology in this discipline (Freeman 2008; Morrison and Snodgrass 2011). Experiments are deemed to have an impact on several aspects of computing: their importance is recognized for assessing computing systems' performance and for triggering new developments (Freeman 2008; Morrison and Snodgrass 2011; Tichy 1998) and experimentation with prototypes is considered essential in use-inspired research and product design (Snir 2011). Moreover, the use of the experimental scientific method is advocated to understand computations that are often too complex for mathematical analysis, to prove their correctness, to check consistency with hypotheses, to uncover performance constraints, and to show whether original goals are met (Denning and Freeman 2009).

'Experimental computer science' has become a quite common label to which today different meanings can be associated; (Feitelson 2006) acknowledges at least three of them. The first one refers to the type of research devoted to the realization of concrete systems; this kind of activity lies in the realm of engineering rather than science, and thus its experimental side is related to the demonstration of the feasibility of these systems, whether software or hardware. The second meaning (Denning 1981) views experimental computer science as the mathematical modeling of the behavior of complex systems, where the anticipated properties of the systems have to be tested experimentally. This 'experimental feedback' is recognized elsewhere as the underlying force of computing (Newell and Simon 1976). The third meaning defines the discipline as the evaluation of computer systems by means of the traditional methodologies of natural sciences (Tichy 1998), such as repeatability.

Despite the increasing interest in a more rigorous methodological approach to computing, many lament that the current methodology is inadequate and that, in comparison with other fields (e.g., natural sciences), computer scientists should experiment more (Denning 2005). Many recommendations (see Harrison and Basili 1996; Barni et al. 2007; Vandewalle et al. 2009; Mayer and Nordio 2010) present common traits: they stem from the acknowledgment of a crisis in computing that is meant to be overcome with a greater maturity of the discipline, in terms of more rigorous experimental methods to the search for solutions. Moreover, they accept the view that computing is a science without much discussion: "The question of 'scienceness' of computing has always been complicated because of the strong presence of science, mathematics, and engineering in the roots and practice of the field. [...] Computing is now accepted as science. Some of us even believe computing is so pervasive that it qualifies as a new domain of science alongside the traditional domains of physical, life, and social sciences" (Denning 2013, 37–38). As a consequence of computing being considered a fundamental science, the scientific method should apply, but no systematic analysis on how it should be intended in this particular context is provided: "Experimentation is central to the scientific process. Only experiments test theories. Only experiments can explore critical factors and bring new phenomena to light so that theories can be formulated and corrected. Without experiments, computer science is in danger of drying up and becoming an auxiliary discipline. The current pressure to concentrate on application is the writing on the wall. I don't doubt that computer science is a fundamental science of great intellectual depth and importance. Much has already been achieved. Computer technology has changed society, and computer science is in the process of deeply affecting the world view of the general public. There is also much evidence suggesting that the scientific method does apply. As computer science leaves adolescence behind, I hope to see the experimental branch of this discipline flourish" (Tichy 1998, 40). Another recurrent theme in this discussion is the success of scientific reasoning and the necessity to extend it to computer science: "These examples and other extant computer science theories emphasize that by embracing the methodology of developing and evaluating predictive models through experimentation over multiple members of a class of software systems, a more complete understanding of such artifacts will emerge. [...] How can these benefits be realized? How might we change what we do? We can adapt our already very skilled hypothesis testing in debugging and broaden it by asking more general questions [...] The pristine presentations of scientific reasoning and the tremendous successes of such reasoning in other fields may appear to the practicing computer scientist as out of reach. But many of our colleagues have started down this path, the tools are accessible, and the promise is great" (Morrison and Snodgrass 2011, 38).

In what follows we propose to enlarge this debate, without taking for granted that the traditional experimental method should be applied to computing, but rather analyzing from a critical standpoint the limits of this application. In our view, a good starting point is the acknowledgment of the variety of ways in which the term experiment is used in the field of computing. As it has been reconstructed in detail in (Tedre 2015), at least five different views of experiments can be recognized in the practice of the field. There are the so called *feasibility experiments* aimed at empirically demonstrating (demonstration and experimental are terms commonly used as synonymous in computer science) the proper realization and working of a technology. There are *trial experiments*, evaluating some aspects of the system using predetermined variables, and *field experim*ents, aimed at evaluating these aspects of the system outside of the laboratory in the real world. There are also *comparison experiments* devoted to compare among different solutions by looking for the best solution for a specific problem. And finally there are *controlled experiments*, those more similar to the traditional notion of experimentation aimed at achieving generalization and prediction. What is important in this account is not how the notion of experiment should be used, but how in fact it is used: "Many would object against calling, for instance, feasibility demonstrations 'experiments,' arguing that the term 'experiment' has a special meaning in science. They are right. But if one looks at how authors in computing have used the term—not how it should be used—those five uses are easily found" (Tedre 2015, 190).

Some Reasons (and Examples) that Stretch the Traditional Categories

The five views introduced by Tedre (2015) constitute the starting point for extending the conception of an experiment in computing in the direction of experiment as *exploratory work* on novel and interesting ideas or techniques. However, contrary to Tedre, we are not interested here in a comprehensive description of the ways the notion of experiment is currently intended in computing, but rather on a reflection on how some traditional categories, such as that of the controlled experiment, are challenged in current computing practice.

The coexistence of these different views on experimental computer science is probably due to a variety of reasons: surely the large number of the subareas of computing plays a key role; moreover, the novelty of the whole field (with respect to more traditional scientific disciplines) contributes to a lack of uniformity regarding methodological issues. Anyway, besides the investigation of these reasons (which are out of the scope of this work), what seems important to us is the acknowledgment of the wide differences between the various components of computer science and engineering that are reflected in the differences in the ways experiments are intended and performed. In the following, we present two paradigmatic examples that show how the emphasis on the experimental method is differently acknowledged and carried out in two subareas of the field.

Let us consider first *empirical software engineering*, which aims at the study and application of techniques for the design, development, operation, and maintenance of software. Here more and more attention has been progressively devoted to methodological issues trying to adopt a rigorous approach inspired by the scientific experimental tradition, namely the controlled experiment. If we consider the 50 most cited papers (according to Scopus) from “Empirical Software Engineering: An International Journal” (Basili and Briand 1996) in the 2003–2012 decade, it is significant to observe that, besides case studies, reviews, empirical analysis, comparative analysis and field studies, more than 1/5 of the articles present a form of controlled experimentation (even if not all of them explicitly refer to the presented work as a controlled experiment). Indeed, they pay attention to replication and present results derived from replicated rounds; they carefully design the experimental setting, with an informed reflection on the choice and variation of controlling factors; they devote attention to the issue of the number of subjects involved in the experimentation trying to enlarge it in a considerable way; they present in most of the cases a considerable statistical analysis of their results; they are interested in both the internal and external validity of their results so making it possible to generalize these results.

A closer look at one of these articles, explicitly referring to the presented work as a controlled experiment (Vokac et al. 2014), shows a replication of a previous experiment aimed at verifying the beneficial effects of design patterns (i.e. proven solutions to design problems organized into reusable software modules) with the motivation to support the industrial use of design patterns. Accordingly, to increase experimental realism, a real programming environment, instead of pen and paper, is used, and 44 paid professionals from multiple companies as subjects, instead of the 29 volunteers from a single organization like in the previous attempt. Moreover, the repeated experiment is carefully designed not just in terms of subject selection, background and group assignment, but also in terms of the expectations and hypothesis that are clearly expressed in the description of the experiment conduct. This is a very important aspect that, together with the authors’ claim to add a qualitative analysis (beside the quantitative analysis of dependent variables) in the discussion, clearly shows their purpose to try to explain the reasons why the quantitative results were observed. This attention to qualitative explanations, and not only to quantitative analysis, can be seen as an important step toward a more articulated view of experiments, more similar to the traditional notion of the controlled experiment.

Let us consider now another area of computing that is autonomous mobile robotics. This is a field oriented to develop robotic systems that are autonomous in the sense that they have the ability to maintain a sense of position and to navigate without human intervention, in order to operate in places hardly accessible by humans or in cooperation with humans in common environments. Note that human operators are not completely excluded from autonomous mobile robotics, but they evolve from being active tele-controllers of the robotic systems to being more passive tele-supervisors of the same robotic systems. The emphasis on experiments and the effort in developing good experimental methodologies have gained a growing attention also in autonomous robotics in the last years. Basically, this community started to recognize that experimental methodologies have not yet reached the level of maturity of other computing disciplines. To cope with this situation, a number of initiatives have been promoted, ranging from workshop series on these topics, to special issues of journals, to European projects funded under different programs. However, the analysis of experimental trends that emerge for instance from the autonomous mobile robotics papers presented over the last 10 years at the international conference on autonomous agents and multiagent systems (AAMAS) (Amigoni et al. 2014) shows that, if from the one side the principles of the experimental method (such as comparison, reproducibility and repeatability, justification and generalization) play an inspirational role in the direction of a more rigorous approach to experiments (say controlled experiments), from the other side these rigorous approaches are not yet a full part of the current research practice.

From the systematic analysis presented in (Amigoni et al. 2014) and from another more preliminary research (Amigoni et al. 2009), it emerges that none of the experiments considered in autonomous mobile robotics can be properly labelled as controlled. The increasing use of public data over which different systems can be run and compared is surely a sign of how comparison is acquiring a crucial importance in this field, as well as the recent trend toward the development of comparable implementations of systems, starting from their descriptions provided in papers and reports, and using the same code that was used in previous experiments. Moreover, the public distribution of code and/or problem instances (data sets) is a positive sign that experimentation is moving toward a more rigorous approach. However, experiments involving several data sets referring to different environments (indoor or outdoor) are still not so common and, hence, the implementation of similar experiments, to understand for instance which parameters influence a robotic system, is very difficult. Plus, the report of anomalies in performance, that should help in detecting those issues deserving further attention, is rare. Finally, weak attention is given to statistical analysis of results, thus compromising the possibility of justifying and explaining them.

The case of autonomous robotics, and the way experiments are performed, represents a clear example of a subarea of computing in which, although the importance of the notion of controlled experiment is recognized, the traditional experimental scenario is not fully applied. We believe that the reasons for this are due not only to a lack of methodological maturity of this field, but also (and perhaps principally) to more intrinsic reasons, as we are going to argue in the following section of the paper.

Toward Explorative Experiments

The examples presented, the notion of directly action-guiding experiments (Hansson 2015), and some preliminary reflections on intervention and control in social experiment (Kroes 2015) represent the starting points for introducing the idea of explorative experiment in computing that is discussed in this section. To a first approximation, explorative experiments are investigations on the functioning of an artefact and its interaction with the environment in the absence of a proper theory or theoretical background and without the typical constraints of controlled experiments.

Let us consider the notion of directly action-guiding experiments as a way to stress the kind of experimental intervention that characterize explorative experiments in computing. In particular, the difference between epistemic experiment and directly action-guiding experiments, as recently conceptualized in (Hansson 2015), can help emphasize in this discussion not only that explorative experiments are performed on artefacts (and not on natural phenomena), but also that they have different purposes than the epistemic ones. An experiment is *epistemic* when aims at providing us with information about the workings of the natural world, whereas an experiment is *directly action-guiding* if and only if satisfies two criteria: a) the outcome looked for should consist in the attainment of some desired goal of human action, and b) the interventions studied should be potential candidates for being performed in a non-experimental setting in order to achieve that goal. A clinical trial of an analgesic is one of the examples provided by Hansson of a directly action-guiding experiment, where the outcome looked for is the efficient pain reduction and the experimental intervention is the treatment that might be administered. A systematic test on an autonomous robot employed to assist an elderly person in her home is also an example of a directly action-guiding experiment: the outcome looked for is the proper interaction of the robot with the person and the experimental intervention consists in the careful tuning of the abilities that the robot must possess to positively achieve this goal.

Let us focus now on the explorative character as a form of investigation of the unknown. Although the concept of experiment as exploration is not new, we introduce it as a further category of experimentation in computing, with respect to the five categories already introduced in (Tedre 2015), to give reason of a significant part of the experimental practice in computer science and engineering. In some recent philosophical research *exploratory experimentation* has been used to label those forms of experimentation in science which are not always guided by theories. One of the first authors to recognize the epistemic importance of exploratory experiments (Steinle 1997) defines exploratory experimentation as driven by the desire to obtain empirical regularities when no well-formed theories or no conceptual framework are available. What is important in this characterization (that in this case it is based on a detailed reconstruction of the early research in electromagnetism) is that the experimental activity may be highly systematic and driven by the typical experimental guidelines, despite its independence from specific theories. The same term is used with a slightly different meaning in an another article in the same year but in the context of some early research in protein synthesis (Burian 1997), where exploratory experimentation is seen as a style of inquiry not guided by theory. These and other similar works are mainly directed against the theory-driven approaches of most of the philosophy of science in the spirit of experimentation as having a life on its own (Hacking 1983). Even if they recognize that exploratory experimentation is typically not free of theory, they aim at showing the epistemic significance of those inquiries that are not primarily theory-driven by presenting several detailed case-studies. The idea that "the aim of exploratory experiments is to generate significant findings about phenomena without appealing to a theory about these phenomena for the purpose of focusing experimental attention on a limited range of possible findings" (Waters 2007, 5) is probably that serving better as an inspiration for more recent works devoted to provide evidence of the exploratory shift observed in the methodology of some areas of biology (Franklin 2005).

To our purposes, however, this emphasis on theory (even when theory is relegated as background knowledge) is out of scope, as it is not even completely clear what a theory in computer engineering is or whether references to a theoretical background play a key role in experiments in computing. This is also the reason why we use the term ‘explorative’ instead of ‘exploratory’ to mark our difference from the philosophical work focused on accounting the distinction between exploratory and theory-driven experiments on the ways in which experiments depend on theory. In our attempt to characterize explorative experiments in computing we are interested, rather, in the appeal to complexity that has been stressed in the philosophical literature (Burian 2007), where some systems are considered too complicated to be investigated by means of a theory-driven approach. If this appeal to complexity certainly applies to biology, we believe that there are good reasons to apply it to computing fields as well, in particular when considering that what is the subject of the experimentation are not just the artefacts per se, but also the ways these artefacts are able to interact with the surrounding physical and social world.

Indeed the reference to complexity helps in defining one important aspect we wish to stress in our characterization of explorative experiments: the fact that there is not sufficient information (in most of the cases for the lack of a proper theoretical background and/or previous experience) to provide exact expectations of what investigators will find. And, thus, explorative experimentation is a way to find patterns of activities from which later scientists could generate new hypotheses. In this sense explorative experiments are forms of investigation on novel and interesting ideas or techniques without the typical constraints of rigorous experimental methodologies.

Controlling the experimental factors that are to be investigated constitutes one of the key factors of the experimental method. To realize an experimental system, knowledge and control of the interactions between the system and its environment need to be managed. Controlled experiments are usually performed having in mind quite precise expectations of the possible outcomes. The research questions are clearly stated and the hypotheses to be investigated are made explicit. Then, experiments are designed and performed, varying the different experimental parameters in order to determine which of the different experimental conditions are indispensable and, then, looking for stable empirical rules. Amongst the strategies for producing stable and repeatable experiments, experimenters vary a number of factors in their experimental systems to examine whether they are relevant or not. The fact that experiments are performed in laboratories responds exactly to this attempt of control.

Traditionally the control paradigm for experimentation, as has been devised in the history of science, relies on two assumptions (Kroes 2015): the experimenter is not part of the system on which the experiment is performed and (s)he is in control of the independent variables and of the experimental set-up. Accordingly, (s)he is able to intervene both by changing these variables to evaluate their influence on the dependent ones and by varying the experimental set-up. As it has been argued by Kroes in the case of new technologies, this traditional control paradigm becomes problematic and a shift in the notions of intervention and control can be observed when considering new technologies as socio-technical systems or as involving socio-technical systems, namely as hybrid systems composed of natural objects, technical artefacts, human actors and social entities: the idea of controlling the experimental system from a center of command and control outside becomes highly problematic (Kroes 2015). Not only the distinction between the experimental system and its environment is critical, but also the environment is complex being composed of both technical artefacts and natural and social elements.

It is interesting to note that the same crisis in the traditional notion of control can be observed also in a part of the current experimental practice in computing. Although the kind of technology (computer science and engineering) we are discussing here does not possess in a full and complete way the features of large-scale socio-technical systems, such as the world civil aviation system (Vermaas et al. 2011), it presents already some of the characteristics of a socio-technical system. In other words, we could say that the experimental system in the case, for instance, of experiments with autonomous mobile robots is hybrid, in the sense that not just technical components play an essential role for the functioning of the system, and thus have to be evaluated, but also natural objects, human actors and social entities need to be taken into account. Moreover, if in the natural sciences it is prescribed that the experimenter should be an outsider of the phenomenon to be explained, it is not clear how a person working in computing, which is aimed at producing computation-based artefacts, could be an outsider with respect to a phenomenon (i.e. an artefact) that (s)he has created (Tedre 2011). Except from some significant examples, experiments in computing are usually performed by the same people that has created the artefacts and, at the same time, need to test them, losing the sort of independence of the experimenter prescribed in the classical experimental protocol.

One could ask what is the reason for the same person to test the artefacts that (s)he has created and, thus, should know in detail. To answer this question, it is important to recognize a certain level of unpredictability arising both in the artefact, due to its complex nature, and in its interaction with the physical environment (including the persons) surrounding it. This is particularly evident in the case of autonomous robotics, where the goal is of having robots that do not require continuous human supervision. Autonomous robots are very complex entities whose behavior is hardly predictable, even by their own designers, especially when considering their interaction with the physical (and the social) world. Not only parameters and factors to test that a given robot is working properly, and possibly better than others, have to be taken into a rigorous account without the required independence of the experimenter, given that the experimenter and the creator of the artefact are the same; but also autonomous robots have to be tested for their interaction with an environment (including in most of the cases human beings) that is hardly predictable. This does not mean that experiments to test the functioning of these artefacts cannot be made, but that some constraints relative both to the type of object (technical artefacts rather than natural phenomena) and to the procedure of evaluation (explorations rather than controlled experiments) are to be taken into account.

A good example is provided by one of the challenging application areas in autonomous mobile robotics that is search and rescue, which aims at developing robotics systems to assist human rescuers in detecting and reaching victims after a disaster. Tasks range from moving between locations of disaster environments, building spatial representations (maps) of the environments, to searching environments for victims. Generally autonomous robots for search and rescue are developed to explore the largest possible amount of the area of an initially unknown environment in a given period of time, or to explore the whole area of an initially unknown environment in the shortest possible time. During exploration, robots can be required to collect information about the presence of victims, about the possible paths traversed by human rescuers, or about the structural stability of buildings. As the use of autonomous robots in search and rescue applications is rather complex, the presence of human operators is required to supervise operations and to actively intervene on the system in case of unexpected problems. Hence, the instructions to use these robots are complex and usually require human operators to undergo some training. Due to these characteristics, to experiment on the functionalities and behavior of these robots, they need to be evaluated at run-time, usually being the distance from the planned (expected behavior) at design-time larger than expected. The attempt to anticipate at design-time, through modeling, the possible problems that can arise at run-time is one of the fundamentals of engineering (Vincenti 1990). However, when considering autonomous mobile robotics, the above distance increases because the modeling at design-time of the interactions between robots and real environments in which they are embedded becomes very complex. The reasons are multiple, but they can be summarized in the wide variability of situations in which autonomous robots can find themselves and in the high difficulty in modeling their interactions with the environment, especially when humans (as in the search and rescue scenario) are involved. To cope with this difficulty, experiments in autonomous robotics aim not only at testing whether and how the design proves have been correctly translated in practice, but also at understanding the often not fully predictable behavior of these robots when interacting with the physical and social world.

As we have already noticed in "Some Reasons (and Examples) that Stretch the Traditional Categories" section, the attempt of autonomous robotics to conform its experimental methodology to the protocol of controlled experiment cannot be fully carried out. The examples considered indicate that a purely controlled form of experimentation is not possible due to the lack of some features that characterize the traditional protocol and have to do with the control of experimental factors. This is particularly evident when we consider, for example, the kind of experiment performed to understand the behavior of a robotic system moving from location A to location B in the presence of obstacles that were not explicitly modeled in the design of the robot. Indeed a behavior for obstacle avoidance (considering only polygonal obstacles) could be designed for the robot, but predicting its performance for non-polygonal obstacles (such as round ones), that can be easily encountered in real applications, is very difficult. This difficulty is due to a number of reasons, such as errors in deciding what to do to negotiate a perceived obstacle, errors in the locomotion for avoiding the obstacles, bugs in the software program deciding where to move in order to avoid a detected obstacle, errors in performance, when for instance a properly decided action is not performed as expected. If some of the above mentioned causes (such as that related to the control of software programs) can be addressed adopting the classical software testing tools, others are specific to robotics and can be hardly addressed using tools originally designed for software programs that interact with "controlled" environments.

Modeling and predicting all these aspects is not only far beyond the current and near-future technical knowledge, but also not in the experimenter's control due to the intrinsic reasons we mentioned above: the experimenter is part of the system and (s)he is not in full control of the experimental set-up (Kroes 2015). For example, predicting ex ante which features of the environment are not properly represented, before deploying the robotic system in the actual environment, is almost impossible, making the traditional control paradigm not applicable, and moving the current practice to a form of experimentation that is more suitable to call exploration. In this case, experiments are carried out to explore possibilities, to investigate opportunities, and they give back information that is iteratively used to improve the artefacts both in their structure and in their interaction with a complex environment. What is explored is only partially known in advance, and surely not at the level of being expressed in the form of clear hypotheses to be tested later in an experimental campaign.

With this discussion and these examples in mind, we can now attempt a less tentative and more general definition of explorative experiments in computing. We can say that explorative experiments are a special kind of directly action-guiding experiments which possess the following features:

- They are devoted to testing artefacts, meant as artificial entities purportedly built by humans to fulfill a purpose and, therefore, having a technical function.
- They are not devoted to hypothesis testing, but to investigate the realm of possibilities pertaining to the functioning of an artefact and its interaction with the environment in the absence of a proper theory or theoretical background.
- The control of the experimental factors cannot be managed from the beginning, but it is in part carried out after the artefact has been inserted into its environment.

Explorative Experiments, Social Experiments and a Posteriori Control

In this section we intend to argue on how the notion of explorative experiment and that of social experiment share some commonalities and how they can contribute to their respective shaping through the notion of a posteriori control. *A posteriori control* is the form of control that characterizes explorative experiments that are driven by the desire of investigating the realm of possibilities pertaining to the functioning of an artefact and its interaction with the environment in order both to acquire knowledge about the performances of the artefact itself and to find out proper concepts to formulate possible regularities. A posteriori control refers, in particular, to the fact that the experimenter is not in full control of the experimental setting due to the impossibility of anticipating plausible outcomes for the lack of a proper theory or theoretical background that make impossible to clearly state hypotheses. In this situation, establishing a priori dependent and independent variables is problematic so that to precisely track what aspects will be controlled during the experiment becomes impossible.

The above described characteristics of explorative experiments in computing resemble some aspects of the view of new technologies as social experiments (Science and Engineering Ethics, this special issue), both being forms of experimentation in society intended as a laboratory to experiment with technologies. The idea of technology as social experiment is not new and derives from how the notion of social experiment has been developed in the social sciences where various conceptualizations, even if they emphasize one aspect or another of this idea, conceive society as a sort of laboratory to experiment in with technologies. According to (van de Poel forthcoming) a social experiment is an experiment *in* society, *on* society and done *by* society. This means not only that technologies can be (and sometimes they must be) experimented in society, but also that these experiments are conducted to learn about the consequences of the technologies on the society and that, in a sense, it is the society itself that carries out such form of experimentation. Even if the notion of social in the case of explorative experiments in computing has to be intended still in a very weak sense,[1] the idea that such technologies are experimented in the real world, and not in laboratories, and thus their results are shaped also by society (where society is restricted in our case to the actual users of the technology) clearly recalls social experimentation.

Moreover, the notion of control needs to be reconsidered in the case both of social experiments and explorative experiments, as in both cases the traditional control paradigm cannot be completely applied. In the social experimenting tradition the notion of uncontrolled experiment plays a significant role and can be interpreted in various ways. In particular van de Poel allows for the possibility of uncontrolled experiments (in the sense that they are not controlled by the experimenter); so it is not control that is one of the defining elements of experiment, but the following two conditions: the first is that the phenomenon under investigation is the result of an intervention; the second is that learning should be one of the aims of experimentation. Although agreeing with these remarks, we believe that the notion of control should maintain its centrality, as it is precisely control that distinguishes experimentation from other forms of investigation. Not only experiments have to intervene actively in the material world producing all kinds of new objects, substances, phenomena and processes, but "clearly not any kind of intervention in the material world counts as a scientific experiment. Quite generally, one may say that successful experiments require, at least, certain stability and reproducibility, and meeting these requirements presupposes a measure of control of the experimental system and its environment as well as a measure of

[1] Consider for example autonomous robotics, where it is plausible to think that in the future this social scenario will progressively extend both for the increasing use of autonomous robots in everyday life and for the social implication this use will entail.

discipline of the experimenters and the other people involved in realizing the experiment" (Radder 2009, 3).

To maintain the centrality of control in explorative experiments conceptualized in the broader framework of social experiments, while recognizing a different weight of it with respect to controlled experiments, a promising strategy seems that of mitigating the notion of control itself. In experiments as explorations control could be intended in the a posteriori form, in opposition to the a priori form that usually takes place in traditional experimental contexts. If in the latter experimental factors are fully in control of the experimenter in a sort of anticipation of the scenario to be tested, in the former the possibility of anticipation disappears and the option for control is in the experimentation as exploration after new technologies have been introduced within society. Therefore, although a different meaning for experiment is required when dealing with a posteriori control, nevertheless control, in one form or in another, still represents one of the defining properties of an experiment.

Let us consider again the example discussed in "Toward Explorative Experiments" section about search and rescue applications in autonomous mobile robotics, and in particular the experiment devoted to understand the behavior of a robotic system moving from location A to location B in the presence of obstacles not explicitly modeled in the design of the robot and of the experiment. We have already discussed how the notion of control cannot be applied according to the traditional control paradigm, as the possibility of anticipating the scenario to be tested disappears. Still some forms of control are possible after robots are inserted in real world environments in which explorative experiments are conducted. First, the research questions to be explored are set by the experimenter: this is true also in the case of the explorative experiments we have discussed so far where, even if initial hypotheses cannot be clearly stated, directions of exploration are devised and ways of implementing them are realized. Secondly, control is exercised in the ways robots' behaviors are organized for the specific experiment at stake. The possibility of failure for an autonomous robot given in the case in which the properly decided action is not performed as expected, for instance due to the surface of the environment that makes wheels slip, is something that cannot be always predicted in advance, but can be reconsidered a posteriori after the technology has been introduced in society. Finally, the acts of measuring performances, of analyzing and explaining data, of generalizing solutions also represent weaker forms of control that mitigate the idea that, when control does not work as in the traditional experimental paradigm, experiments become suddenly and completely uncontrolled.

By introducing this notion of a posteriori control, although in a very preliminary way, we aim at emphasizing some features of explorative experiments in computer engineering and, possibly, suggesting that this notion could be considered to characterize control in social experimentation. We are well aware that different forms of a posteriori control are likely to be observed in different fields of application so that it is difficult at this stage to characterize this notion in a general way. We suspect, however, that to consider explorative experiments in computing as a very special form of social experimentation and control in social experiments as a case of a posteriori control typical of experimentation in computing constitutes a good starting point to stretch the traditional notion of experiment within socio-technical systems and to go beyond the concept that social experiments can take place all the time, whether we are aware of them or not.

Conclusion

We hope to have substantiated the need of reconsidering the traditional notion of controlled experiment within the field of computer science and engineering. The well-known notion of controlled experiment is not sufficient to give reason for the different experimental practices that characterize this field and, therefore, we have introduced the notion of explorative experiment to characterize experiments which are not theory guided and should be better intended as explorations rather than as hypothesis testing. To better clarify this notion we have discussed examples coming from a recent area of computer engineering, namely autonomous mobile robotics, where the unpredictability that arises both at the level of the artefact and at the level of its interaction with the physical and social environment dissolves the idea of traditional controlled experiments. Finally, by recasting explorative experiments as a peculiar form of social experiment, intended to examine the introduction of new technologies in society, we have considered how these two notions can mutually benefit, in particular when considering how a posteriori control could provide a form of control when working in explorative experiments.

In conclusion, this preliminary and partial analysis of experimentation in computing already forces us to extend the traditional notion of the experiment. This extension could be useful also in shaping the notion of a social experiment as experiment in society, on society, and done by society.

Acknowledgments I am grateful to all the participants and organizers of the workshop "New Technologies as Social Experiments", held at TU Delft in January 2014, and two anonymous reviewers.

References

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

Amigoni, F., Schiaffonati, V., & Verdicchio, M. (2014). Good experimental methodologies for autonomous robotics: From theory to practice. In F. Amigoni & V. Schiaffonati (Eds.), *Methods and experimental techniques in computer engineering. Springer briefs in applied sciences and technology* (pp. 37–53). Berlin: Springer.

Barni, M., Perez-Gonzalez, F., Comesana, P., & Bartoli, G. (2007). Putting reproducible signal processing into practice: A case study in watermarking. In *Proceedings of IEEE international conference on acoustics speech and signal processing*.

Basili, V. R., & Briand, L. C. (Eds.). (1996). *Empirical software engineering: An international journal.* Berlin: Springer.

Burian, R. M. (1997). Exploratory experimentation and the role of histochemical techniques in the work of Jean Brachet, 1938–1952. *History and Philosophy of the Life Sciences, 19*, 27–45.

Burian, R. M. (2007). On microRNA and the need for exploratory experimentation in post-genomic molecular biology. *History and Philosophy of Life Sciences, 29*(3), 285–311.

Denning, P. J. (1980). What is experimental computer science. *Communications of the ACM, 23*(10), 543–544.

Denning, P. J. (1981). ACM's president letter. Performance analysis: Experimental computer science at its best. *Communications of the ACM, 24*(11), 725–727.

Denning, P. J. (2005). Is computer science science? *Communications of the ACM, 48*(4), 27–31.

Denning, P. J. (2013). The science in computer science. *Communications of the ACM, 56*(5), 35–38.

Denning, P. J., & Freeman, P. (2009). Computing's paradigm. *Communications of the ACM, 52*(12), 28–30.

Feitelson, D. G. (2006). Experimental computer science: The need for a cultural change. Unpublished manuscript available at http://www.cs.huji.ac.il/~feit/papers/exp05.pdf. Last accessed October 2014.

Feldman, J. A., & Sutherland, W. R. (1979). Rejuvenating experimental computer science. *Communications of the ACM, 22*(9), 497–502.

Franklin, A. (1986). *The neglect of experiment*. Cambridge: Cambridge University Press.

Franklin, L. R. (2005). Exploratory experiments. *Philosophy of Science, 72*, 888–899.

Freeman, P. (2008). Back to experimentation. *Communications of the ACM, 51*(1), 21–22.

Hacking, I. (1983). *Representing and intervening*. New York: Cambridge University Press.

Hansson, S. O. (2015). Experiments before science? What science learned from technological experiments. In S. O. Hansson (Ed.), *The role of technology in science. Philosophical perspectives*. Dordrecht: Springer.

Harrison, W., & Basili, V. R. (1996). Editorial. *Empirical Software Engineering, 1*(1), 5–10.

Hatleback, E., & Spring, J. (2014). Exploring a mechanistic approach to experimentation in computing. *Philosophy and Technology, 27*(3), 441–459.

Keller, E. F. (2002). *Making sense of life: Explaining biological development with models, metaphors, and machines*. Cambridge, MA: Harvard University Press.

Kroes, P. (2015). Experiments on socio-technical systems: The problem of control. *Science and Engineering Ethics Special Issue on Experiments, Ethics, and New Technologies*. doi:10.1007/s11948-015-9634-4.

Langley, P. (1988). Machine learning as an experimental science. *Machine Learning, 3*, 5–8.

Mayer, B., & Nordio, M. (Eds.). (2010). *Empirical software engineering and verification. Lecture notes in computer science*. Berlin: Springer.

McCracken, D. D., Denning, P. J., & Brandin, D. H. (1979). An ACM executive committee position on the crisis in experimental computer science. *Communications of the ACM, 22*(9), 503–504.

McLeod, M., & Nersessian, N. J. (2013). Building simulations from the ground up: Modeling and theory in systems biology. *Philosophy of Science, 80*(4), 533–556.

Morrison, C., & Snodgrass, R. (2011). Computer science can use more science. *Communications of the ACM, 54*(6), 38–43.

Newell, A., & Simon, H. (1976). Computer science as empirical inquiry: Symbols and search. *Communications of the ACM, 19*(3), 113–126.

Radder, H. (Ed.). (2003). *The philosophy of scientific experimentation*. Pittsburgh: The University of Pittsburgh Press.

Radder, H. (2009). The philosophy of scientific experimentation: A review. *Automated Experimentation, 1*, 2. doi:10.1186/1759-4499-1-2.

Schiaffonati, V., & Verdicchio, M. (2014). Computing and experiments. *Philosophy and Technology, 27*(3), 359–376.

Snir, M. (2011). Computer and information science and engineering: One discipline, many specialties. *Communications of the ACM, 54*(3), 38–43.

Steinle, F. (1997). Entering new fields: Exploratory uses of experimentation. *Philosophy of Science, 64*, S65–S67.

Tedre, M. (2011). Computing as a science: A survey of computing viewpoints. *Minds and Machines, 21*, 361–387.

Tedre, M. (2015). *The science of computing*. Boca Raton: CRC Press, Taylor and Francis Group.

Tichy, W. (1998). Should computer scientists experiment more? *IEEE Computer, 31*(5), 32–40.

Van de Poel, I. (forthcoming). Society as a laboratory to experiment with new technologies. In E. Stokes, D. Bowman, & A. Rip (Eds.), *Embedding and governing new technologies*. Singapore: Pan Stanford Publishing.

Vandewalle, P., Kovacevic, J., & Vetterli, M. (2009). Reproducible research in signal processing. *IEEE Signal Processing Magazine, 37*, 37–47.

Vermaas, P., Kroes, P., van de Poel, I., Franssen, M., & Houkes, W. (2011). *A philosophy of technology. From technical artefacts to sociotechnical systems*. USA: Morgan and Claypool.

Vincenti, W. (1990). *What engineers know and how they know it*. Baltimore: The Johns Hopkins University Press.

Vokac, M., Tichy, W., Sjøberg, D. I., Arisholm, E., & Aldrin, M. (2014). A controlled experiment comparing the maintainability of programs designed with and without design patterns—A replication in a real programming environment. *Empirical Software Engineering, 9*, 149–195.

Waters, C. K. (2007). The nature and context of exploratory experimentation. *History and Philosophy of the Life Sciences, 19*, 275–284.

Winsberg, E. (2010). *Science in the age of computer simulations*. Chicago and London: The University of Chicago Press.

Winsberg, E. (2013). Computer simulations in science. In E. N. Zalta (Ed.), *The Stanford encyclopedia of philosophy*. http://plato.stanford.edu/archives/sum2013/entries/simulations-science/.