The Role of Deliberate Artificial Design Elements in Software Engineering Experiments

Jo E. Hannay, *Member*, *IEEE Computer Society*, and Magne Jørgensen

Abstract—Increased realism in software engineering experiments is often promoted as an important means of increasing generalizability and industrial relevance. In this context, artificiality, e.g., the use of constructed tasks in place of realistic tasks, is seen as a threat. In this paper, we examine the opposite view that deliberately introduced artificial design elements may increase knowledge gain and enhance both generalizability and relevance. In the first part of this paper, we identify and evaluate arguments and examples in favor of and against deliberately introducing artificiality into software engineering experiments. We find that there are good arguments in favor of deliberately introducing artificial design elements to 1) isolate basic mechanisms, 2) establish the existence of phenomena, 3) enable generalization from particularly unfavorable to more favorable conditions (persistence of phenomena), and 4) relate experiments to theory. In the second part of this paper, we summarize a content analysis of articles that report software engineering experiments published over a 10-year period from 1993 to 2002. The analysis reveals a striving for realism and external validity, but little awareness of for what and when various degrees of artificiality and realism are appropriate. Furthermore, much of the focus on realism seems to be based on a narrow understanding of the nature of generalization. We conclude that an increased awareness and deliberation as to when and for what purposes both artificial *and* realistic design elements are applied is valuable for better knowledge gain and quality in empirical software engineering experiments. We also conclude that time spent on studies that have obvious threats to validity that are due to artificiality might be better spent on studies that investigate research questions for which artificiality is a strength rather than a weakness. However, arguments in favor of artificial design elements should not be used to justify studies that are badly designed or that have research questions of low relevance.

Index Terms—Artificiality, realism, generalization, theory, experiments, research methods, empirical software engineering.

1 Motivation

In experiments, variables are controlled in order to isolate the outcomes of deliberate interventions, thus creating a situation that is usually not representative of everyday life. This *structural* artificiality is the methodological essence of control that allows the drawing of inferences about treatment-outcome relations from experimental results [91].

In addition, experiments in many disciplines, including software engineering, typically display what one may call *situational* artificiality. This type of artificiality pertains to the experimental ingredients (e.g., subjects, treatments, settings, tasks, and materials) and is defined relative to the target of applicability or generalization. For example, subjects, treatments, and materials may be artificial relative to the software industry because resources dictate that one uses what is at hand and does what it is possible to complete in the time available. In contrast to structural artificiality, which is mandatory within the framework of experiments, situational artificiality may vary according to both practical and analytical considerations.

Software engineering is a field of practice and the goal of research in empirical software engineering is to address issues that are of interest and use to the software industry. Several authors (ourselves included) have argued that, in order to achieve this goal, situational artificiality should be minimized. This means that the realism and representativeness of experimental subjects, treatments, settings, tasks, and materials relative to industrial software development situations should be increased (while retaining the structural artificiality necessary for causal inference) so that the results of such experiments may be generalized and transferred to the software industry more easily [52], [66], [93], [95]. Such arguments are implicitly based on the idea that experimental results are generalizable when experimental situations are immediately similar to industrial situations or when there is a simple statistical route from sample to population based on known probability distributions. As a result, realism is seen as an important design goal, while situational artificiality is seen as a logistically imposed *necessary evil* that hampers the usefulness of the results to the industry.

However, there are other modes of generalization that rely less on mimicking real-world situations and that allow generalization on the basis of deep or structural similarities, e.g., basic underlying mechanisms. In these modes, carefully administered artificiality may be essential for generalization. For example, when small and artificial programming tasks are introduced merely as substitutes for real-world software projects due to the lack of resources, this clearly limits the usefulness of the results. In contrast, when such tasks are included as a deliberate design element in order to better understand or identify causal relationships with respect to a situation such as learning through feedback,

• *The authors are with the Simula Research Laboratory, Department of Software Engineering, Pb. 134, NO-1325 Lysaker, Norway, and the Department of Informatics, Industrial Systems Development Group, University of Oslo, Pb. 1080 Blindern, NO-0316 Oslo, Norway.*
E-mail: {johannay, magnej}@simula.no.

Manuscript received 20 June 2007; revised 11 Jan. 2008; accepted 21 Jan. 2008; published online 11 Feb. 2008.
Recommended for acceptance by N. Maiden.
For information on obtaining reprints of this article, please send e-mail to: tse@computer.org, and reference IEEECS Log Number TSE-2007-06-0195.
Digital Object Identifier no. 10.1109/TSE.2008.13.

the artificiality may increase the usefulness of the study. In this paper, we focus on the role of such *deliberately introduced situational artificiality* in software engineering experiments. Rather than minimizing situational artificiality, we find arguments for *optimizing* it according to the kind of knowledge that one wishes to acquire. Studies using artificial design elements are to be understood as complementary to studies that are high in realism.

Whereas using *structural* artificiality is part of the standard procedure for conducting experiments and situational *realism* is receiving the attention that it deserves in empirical software engineering, the use of situational artificiality as a deliberate means of acquiring knowledge is, as far as we know, explicitly discussed in depth neither in textbooks nor in advanced literature in empirical software engineering.[1] Our goal is to draw attention to this deficit by pointing out the potential for increased knowledge acquisition that lies in the well-planned use of artificial design elements in software engineering experiments. Although we also promote experiments with more situational realism, we think that it is important that the present focus on increased realism in empirical software engineering should not also lead to the stigmatization of experiments in which situational artificiality may be used constructively. The review presented in this paper suggests that, for most authors, realism is viewed as a design goal by default, while artificiality is something to be excused and avoided. Only a few authors consider the possibility of deliberately utilizing situational artificiality in their designs.

Although the issues of realism and artificiality apply to several types of empirical study, our focus in this paper is on experiments, that is, empirical studies in which a deliberate intervention is administered in order to study treatment-outcome relations. Experiments are, by design and convention, caught in the crossfire between realism and artificiality because the structural artificiality that is necessary for control would also seem to result in what is perceived as undesired situational artificiality.

The main body of this paper has two parts: an analytical part in which we evaluate and discuss arguments pertaining to the potential roles of artificial design elements in software engineering experiments (Section 2) and an empirical part in which we analyze points of view and the state of practice concerning artificiality by a content analysis of articles that report software engineering experiments published over the 10-year period 1993-2002 (Section 3). At the end (Section 4), we discuss the implications of our findings in Sections 2 and 3 for empirical software engineering. Section 5 concludes.

2 Situational Artificiality in Experiments

The terms "artificiality" and "realism" are used in various ways, not only in everyday life but also in experimental science. As a result, the classification of something as artificial or realistic, as well as discussions of whether experimental artificiality is good or bad, are often confounded by discussions about what it is to be artificial or real. For the present discussion, we will make the following simple decision: The defining concept is that of *realism* in terms of actual situations in the software industry. Often the term "mundane realism" is used to refer to the correspondence of an experiment's environment to everyday environments. In our discussion, something is then *artificial* if it deviates from what is realistic in terms of situations occurring in the software industry. This corresponds to how the terms have been used in the empirical software engineering literature [52], [56], [66], [93] and in the reviewed articles. There are also other relevant uses of these terms which we will encounter later on.

For us, the important aspect of artificiality lies in the element of *deliberateness* in order to achieve certain goals [92]. While the structural artificiality that pertains to the treatment intervention in experiments is, of course, deliberate, this aspect of artificiality is not the issue here. Our focus is on situational artificiality, which is explained as follows: Cronbach et al. [24], [25] and Shadish et al. [91] list *subjects* (units), *treatments*, *outcomes*, and *settings* as the operational variables from which one generalizes. In addition, since software engineering can be said to be a *design science* concerning artifacts [92], we also include the operational variables *tasks* and *materials.* The suite of these six variables constitutes the *situational variables* of an experiment. *Situational artificiality* and *situational realism* then pertain to the degree to which an experiment's situational variables are similar (by surface similarity, see the following) to corresponding situational variables in the software industry.

2.1 Situational Artificiality and the Problem of Applicability

The applicability of experimental results is usually expressed through principles of generalization. This involves mapping, in one way or another, the situational variables of the experiment to the corresponding variables in the target domain to which one wishes to generalize. The principle of *statistical generalization* is based on drawing a sample from a population where both the population and the sample have known probability distributions. This allows one to generalize by mapping experimental variables to target variables through formal statistical inference [43]. However, for the situational variables that are relevant to software engineering, it is difficult to fulfill the necessary definedness conditions for population and sample. This is a general problem in social and behavioral sciences [91], [110], with which empirical software engineering shares essential methodological issues.

In response to this general problem, Shadish et al. [91] refer to earlier work by scholars such as Brunswick, Meehl, and Cronbach and summarize patterns of argument for generalized causal inference for use in the absence of perfect random sampling. These patterns of argument define *analytical generalization* and embody five principles:

1. *surface similarity*,
2. *ruling out irrelevancies*,
3. *making discriminations*,
4. *interpolation and extrapolation*, and
5. *causal explanation*.

1. In other fields relevant to software engineering, such as management and social and behavioral sciences, deliberate situational artificiality has been given particular attention in, e.g., a special edition of the *Journal of Economic Methodology* [97] and in textbooks, e.g., [34], [69].

TABLE 1
Principles of Generalization

Statistical Generalization:	Generalization based on statistical logic founded on the probability distribution of a sample relative to a well-defined population; e.g., when a sufficiently large number of randomly selected IT professionals are used as subjects in a software engineering experiment assuming that the population of IT-professionals is well-defined.
Analytical Generalization:	Generalization based on arguments other than statistical logic when probability sampling is not possible; hereunder generalization that is...
1. *surface similarity*	...based on immediate likenesses between experimental and target situational variables; e.g., when IT professionals who are participating as subjects in an experiment are put in offices that resemble their workplace to perform realistic tasks on a full-scale industrial system.
2. *ruling out irrelevancies*	...based on discharging factors that do not affect generalization; e.g., arguing that it is irrelevant to the issue of generalization that tasks are administered via a web interface in an experiment instead of by, say, a project manager.
3. *making discriminations*	...based on clarifying factors that limit generalization; e.g., when arguing that the division of IT consultants into juniors and seniors based on manager-assessed market value is not necessarily a measure of programming skill, so one should not dilute the generalization argument by attempting to include skill.
4. *interpolation and extrapolation*	...based on interpolating from extreme values to other values within a range, or on extrapolating from values within a range to outside the range; e.g., when the fact that a treatment shows an effect in a setting that is presumed to counteract the treatment is used to imply a greater effect in settings (perhaps more realistic) that do not counteract the treatment (interpolation) or when an effect that is thought to have been caused by an increase in levels of programming skills between subject groups is used to argue that a similar effect will manifest itself relative to a third group with even greater skills (extrapolation).
5. *causal explanation*	...based on deep or structural similarities, hereunder testing and developing explanatory theories; e.g., when arguing that the use of students as subjects is meaningful because the relevant cognitive processes are the same as those of professionals, or when arguing that the short tasks of an experiment capture the underlying essence of problem-solving encountered in industry-related tasks.

Table 1 summarizes these principles. Note that the use of such a pattern of argument does not automatically entail valid generalization. Each use of a pattern must be justified individually.

The mapping of experimental situational variables for each of these five principles relies on arguments of *construct validity* and *external validity*. Generalization with respect to construct validity concerns the mapping from the situational variables to the constructs (concepts) that they are intended to represent, that is, from *specific* to *abstract*, while generalization with respect to external validity concerns the inference that a causal relationship holds over relevant *variants* of the situational variables, that is, in other concrete situations [91, chapter 11] (see Fig. 1). Each of the five principles will, in practice, appeal more strongly to either construct validity or external validity. For example, surface similarity could be used to relate experimental variables directly to target variables without involving constructs, thus emphasizing external validity, while causal explanation might involve the constructs of theory and would therefore emphasize construct validity to a greater extent.

Arguments for increased realism in software engineering experiments assume that generalizing is done only on the principles of surface similarity and statistical generalization. However, these principles, which rely on representativeness and realism (and are therefore the most vulnerable to artificiality) only represent one mode of generalizing.

Generalizing by causal explanation (i.e., Principle 5), on the other hand, relies on finding causal relationships that underlie several seemingly disparate phenomena [91], [105]. This principle searches for deep or *structural* similarities between experimental and target situations. In medicinal research, many important findings are based on mouse trials, the rationale for which is the deep similarity of mouse and human DNA, although mice are not men as judged by surface similarity. In empirical software engineering, artificial problems may be structurally similar to realistic tasks with respect to task complexity and students might share the deep similarity of comprehension-related cognition with professionals.

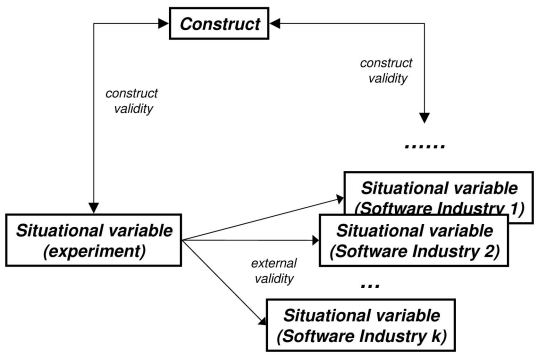

Fig. 1. Analytical generalization through construct validity and external validity. Generalization with respect to external validity is done by arguing that what is observed for a situational variable in the experiment will also be observed for variants of that variable in the software industry. This type of argument may follow one or several of the five principles mentioned in Section 2.1. Generalization with respect to construct validity is done by arguing that a situational variable stands for an abstract construct (concept) that, by virtue of its abstraction, also stands for situational variables in the software industry to which one wishes to generalize. The argumentation that situational variables "stand for" constructs, and vice versa, may again follow one or more of the five principles mentioned in Section 2.1.

Deep or structural similarities may not correspond to similar surface similarities [91, p. 369] and opting to generalize by using only surface similarities runs the risk of generating only partial knowledge. Note that arguing on the level of deep and structural similarities requires such similarities to be established beforehand, either empirically or theoretically.

In what follows, we will investigate the claim that artificiality is beneficial for acquiring additional deeper structural knowledge and that realism, not artificiality, may stand in the way of knowledge gain and generalization in software engineering. However, deliberate artificiality does not replace realism: It complements it. The timeliness of either depends on the kind of knowledge that one is attempting to acquire [55].

2.2 Deliberate Artificial Design Elements

Inspired by Mook [77], we consider four ways in which deliberate artificial design elements on situational variables may promote knowledge acquisition: *isolating basic mechanisms*, *demonstrating the existence of phenomena*, *demonstrating the persistence of phenomena*, and *relating to theory*. These are covered in Sections 2.2.1-2.2.4. Section 2.2.5 discusses the assumption of reductionism that is inherent in these arguments.

2.2.1 Isolating Basic Mechanisms

According to Webster [107], laboratory experiments have features well suited to investigating general principles, independent of time and place,[2] and artificiality is essential for building such abstract knowledge. An example in point is the experiment that shows that a feather and a lead ball drop at the same rate in a vacuum. This experiment has an artificial design element (the vacuum) that does not occur naturally on earth and therefore succeeds in isolating the effects of gravitation. To describe motion in everyday conditions, one must add the effects of air resistance as well. However, our understanding benefits from isolating these two basic mechanisms: gravitation accelerates motion and air resistance retards motion. Modularizing knowledge and explaining observations in terms of basic mechanisms relates phenomena to one another at levels other than the intuitive. This, in turn, allows for wider applicability of the obtained knowledge.

For this purpose, the fact that the conditions necessary for isolating basic mechanisms are not representative of everyday life is beside the point because natural conditions may not allow the observation of the mechanisms under investigation. Webster states that "Trying to recreate natural settings in the laboratory is a misdirection of effort and is sure to fail. ... An experiment is useful when we can see the operating theoretical forces in sharp relief. The simpler the experiment, the better it is for developing theory" [107, p. 62].

Isolating basic mechanisms pertains to both identifying basic mechanisms and investigating them. Generalization need not be the immediate incentive in an experiment that primarily seeks to identify basic mechanisms; instead, this may be left to future studies. Generalization *may* be the focus in an experiment that investigates a basic mechanism if one postulates that the same mechanism is operative in a real-world context, in which case, analytic generalization principle 5 (causal explanation) is useful, rather than generalization through realism and representativeness.

There are, of course, basic or underlying mechanisms to be investigated in software engineering. During our work on a review on the use of theory [42], we found several candidates for basic mechanisms that may be worth isolating and investigating further. For example, Laitenberger et al. [65] construct a probabilistic model in which one may formalize the posited causal mechanisms as to why perspective-based reading (PBR) leads to better inspection effectiveness than checklist-based reading (CBR). The posited mechanisms (extracted from the argumentation in [83], [82], [64]) consist of a two-step sequential relationship. PBR gives an increased likelihood of detecting defects targeted by a given perspective (defect subset focus), which leads to a decreased likelihood of detecting defects outside this focus (defect detection overlap), which, in turn, yields better inspection effectiveness. The basic mechanisms here are *subset focus* and *detection overlap.* That it is interesting to study these underlying mechanisms further is demonstrated by the fact that computation and simulation in the probabilistic model show that the mechanisms are largely independent of each other, which is incompatible with the intuitive close relationship that was posited above [65]. However, although several authors refer to basic mechanisms when explaining observed phenomena, few actually investigate them (Section 3).

Conducting empirical studies designed to investigate the effects of the mechanisms in the above example would require one to administer reading perspectives with varying and even unrealistic degrees of "abilities" regarding focus and overlap. Here, artificial perspectives may help one understand how the mechanisms work so that one may control them in order to develop even better PBR.

A basic mechanism is basic, relative to the frame of reference that one uses for one's research. In empirical software engineering, the top-level defining frame of reference may often be described as *determining what the effects of applying a given software engineering technology used by certain developers in a given setting are.* Relative to this top-level frame, one may then ask deeper questions such as *what the underlying (basic) mechanisms in technology, developers, and settings are which make certain combinations of these outperform others.* Frames of reference might change in a hierarchic manner: In order to investigate a certain basic mechanism, one might want to ask what the underlying mechanisms for that mechanism are, and so on.

There have been previous calls for the study of basic or underlying mechanisms. For instance, Basili et al. state that building a body of knowledge from families of experiments, along with the ability to carry out families of replications, "allows organizations to integrate their experiences by making explicit the ways in which experiences differ ... or are similar and allowing the *abstraction of basic principles*

2. That knowledge is *general* does not mean that it purports to cover all domains. *Scope conditions* delineate the domain of knowledge. Furthermore, *conditionalization* states the conditions under which various consequences (possibly disparate but related) may be derived from the knowledge (see, e.g., [16], [73]).

from this information" [12] (emphasis added), and Höst et al. [51] conclude that researchers need to capture underlying explanatory variables to a greater degree.

2.2.2 Existence of Phenomena

Artificial design elements may be used to demonstrate the existence of a (predicted) phenomenon. Mook states that "we may be asking whether something *can* happen rather than whether it typically *does* happen" [77]. The notion of an *exhibit* is a useful instrument for demonstrating the existence of a phenomenon. An exhibit is an experiment design "which reliably induces some specific regularity (or 'effect' or 'phenomenon')" [98], that is, a design purposefully created to bring out a phenomenon. Such findings "do not represent a class of real-world phenomena: they define one" [77]. Hence, demonstrating the existence of phenomena does not focus primarily on generalization but may be the first step in a series of investigations that may eventually lead to generalization.

For example, in order to highlight subject behavior that relates to risk, "potential loss" may be operationalized by extreme treatments such as software projects that may threaten a company's survival [59], although the intention may be to investigate risk in typical situations. The deliberate use of students as subjects may bring out phenomena related to learning and critical assessment (since students are assumed to be in the "learning mode" and are also being formally assessed regularly). Short tasks where immediate feedback may be given to subjects are another example of artificial treatments designed to bring out learning effects [37].

2.2.3 Persistence of Phenomena

Deliberate artificiality may be used to highlight a mechanism against the backdrop of a particularly *disadvantageous* environment. For example, Waller and Zimbelman [106] examine the dilution effect, which is a bias during human judgment toward underemphasizing relevant information due to the presence of less relevant information. Waller and Zimbelman investigate the mechanism in an auditing setting because it is supposed that the auditing setting inhibits errors of judgment and that, therefore, the persistence of dilution in auditing would imply its persistence in other settings that are less inhibiting. Thus, persistence may be used to generalize analytically by principle 4 (interpolation and extrapolation) from less to more favorable situations.

In software engineering experiments, we may, for example, add artificial elements such as inexperienced users for one of the tools in a study of development tools. If the tool performs just as well as or better than competing tools used by more experienced developers, there are reasons for believing that the tool would perform (even) better in the more favorable situation constituted by experienced users. In other words, we may be in a better position to generalize from the studied situation to larger sets of real situations than what might be reasonable from more realistic experiments.

Failed demonstrations of the existence of a phenomenon may sometimes be viewed as demonstrations of the persistence of the opposite phenomenon whenever the notion of "opposite phenomenon" is meaningful. For example, if learning does not manifest itself in presumed learning-inducing conditions, then the absence of learning, i.e., unaltered behavior, can be seen to persist in these conditions, which should induce altered behavior. This suggests the absence of learning a fortiori in more realistic environments [37]. Although existence and persistence are dually related in this manner, their generalization is not: One may generalize by argument of persistence but not by argument of existence.

2.2.4 Relating to Theory

Theories and models are ways of organizing abstraction. Abstraction is essential to scientific understanding and researchers use abstraction in order to outline the particular constructs (concepts) and relationships of interest for a particular problem [8], [88], [108]. The applicability and explanatory power of a theory then depends (among other things) on the capability of such abstraction to capture salient features of a lesser or a wider range of phenomena.

Several scholars (e.g., Lucas [70], Lynch [71], Webster [107], and Yin [110]) argue that generalization should be done primarily through the abstraction provided by theories. That is, the nature of experiments is such that observations obtained from them cannot be generalized to real life without reference to an encompassing theoretical framework. Generalizing through theory involves generalization principle 5 (causal explanation) in particular since causal explanation is the essence of explanatory theories [101], [90].

Relating experiments to theories means that, rather than generalizing to the real world, empirical studies should relate to theories, which are then, somehow, applied to the real world. The issue is then the extent to which the study reflects the constructs and relationships of the theory and not how representative the study is of a real setting. Thus, Hogarth [48] explicitly refers to representativeness with respect to the constructs of a theory. Other uses of the terms "realism," "representative," and "generalizability" also pertain to constructs rather than to concrete situational variables in the real world. For example, Carlsmith et al. [20] compare mundane realism with *experimental realism*, which concerns how well the experiment involves the subjects (which is a construct validity issue). Aronson et al. [5] further introduce *psychological realism*, which concerns the extent to which psychological mechanisms that occur in the experiment are the same as those that occur in everyday life (see also [6]).

The argument of generalizing through theory simply places an increased emphasis on construct validity, i.e., the relationship between an experiment's situational variables on the one hand and theoretical constructs on the other. In fact, a relationship to abstract constructs is often implicit in empirical research. Although experimenters study treatment-outcome relations between specific situational variables, it is often the cause-effect relationship between general concepts or constructs that is of interest, even though the links to these constructs are often left implicit in practice. When relating to abstract theoretical constructs, which lack surface similarity to the industry, artificial

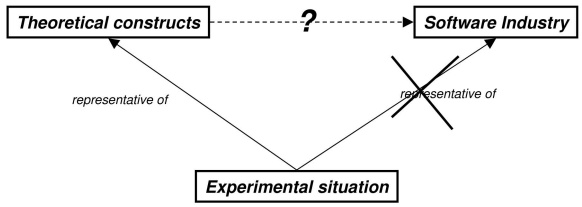

Fig. 2. How can an experiment relate to the field when it is not representative of it?

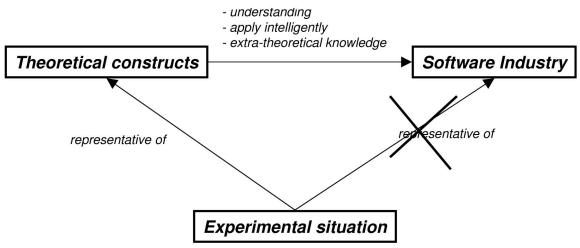

Fig. 3. Applying theory to the field by extratheoretical argument.

situational variables in experiments may yield increased construct validity and hence strengthen the link to theory.

This increased emphasis on construct validity also leads some scholars to dismiss external validity as a basis for generalization. External validity pertains to generalization over variants of situational variables. However, one might argue that situational variables are specific and that it is meaningless to speak of their "variations." Webster concludes that "1) External Validity is the wrong question to ask, 2) experimental results never generalize directly to other situations, and 3) experiments must become more, not less, artificial to be most useful in understanding everyday group processes" [107, p. 60].

For example, in an experiment that investigated judgment upon uncertain information, Abdel-Hamid et al. [1] used student subjects in a role-playing game that simulated a software project over several stages. At the beginning of the simulation, initial productivity estimates were given to the subjects and, at each stage, the subjects were given status reports of the project's progress and were asked to provide new estimates. The study found that the subjects' estimates tended toward the lowest of the initial estimate and the implicit estimates given in the status reports. This finding contradicts predictions based on theories on anchoring [46], [100], which state that biases in judgment arise from both initial estimates and successive supplied estimates. The study was artificial in that the entire simulation took only about 1 hour; however, from the perspective of theory, the value of the study lies not in its representativeness of reality but in its relation to the theoretical construct of "anchoring" in a software engineering context. In this case, the existing theory may be refined by conditioning it with respect to uncertainty.[3]

If the obligation of generalizing results to the industry does not lie with the experimenter, the onus is on the theory to be applicable to real life. This "blaming the theory" [97] is, of course, a source of dispute because, for those who are concerned that experiments lack applicability in the first place, the problem has seemingly been pushed even farther into abstract realms and away from, for example, the software industry. Given that an experiment enjoys high construct validity with theoretical constructs (due to its deliberate artificial design elements), how does one relate this experiment to real life through theory (Fig. 2)?

3. Ten percent of the subjects' grades depended on their performance on this exercise. This may not be representative of the actual cost commitment in a real setting. However, it is worth studying anchoring, even in the absence of cost commitment, so that one may understand and build theory on the isolated effect of anchoring.

The answer to this question lies in the construct validity between a theory's constructs and the situational variables in real life (Fig. 1). But, whereas the construct validity between constructs and experimental variables may often be assessed through statistical means, the construct validity between constructs and real-world variables are often less tractable. Therefore, purely argumentative reasoning may have to be employed in order to establish the construct validity between constructs and real-world variables. We present three examples of such reasoning. The first two have in common that no strict logical link from theory to practice is enforced. Rather, the application of theory relies on extratheoretical arguments (Fig. 3).

The first argument is to dismiss the issue as irrelevant, "because the [theoretical] *processes* [that] we dissect in the laboratory also operate in the real world" [77]. The processes mentioned in the above quotation are our basic mechanisms (Section 2.2.1) and, according to a reductionist view, these are just as real as other more natural phenomena, although basic mechanisms may be integrated with other processes in a real-world situation. Thus, if one observes the theorized anchoring effect in an artificial role-playing experiment such as the one above, then the theory and the experiment enable us to understand that anchoring can occur in a software engineering context, although, in a real situation, anchoring may be masked or reversed by other processes (for example, "endowment" [58]). Note that one must adhere to reductionism in order to claim that anchoring, as a process, survives in more complex situations even when it is not manifest (presumably because it is being masked by other processes). According to this line of argument, of all scientific knowledge, what applies to the real world is this type of *understanding* that is offered by the development of theory.

The second related argument is based on the nature of theories. Theories are often conceptual simplifications of a complex reality designed to facilitate understanding. Theories are therefore not necessarily true in reality (which we may never truly understand) but are instead approximations that are useful for particular purposes (e.g., education, technology development, prediction, and further understanding). Examples abound: the ideal gas model, quantum electrodynamics as expressed by Feynman diagrams, rational choice theory, the models of cognition, etc. Whether such theories are justified and the degree to which they, in fact, represent the way that the world is are epistemological issues. In any event, the transfer of knowledge from such theories to real life does not follow deductive rules. In addition, the causal relationships

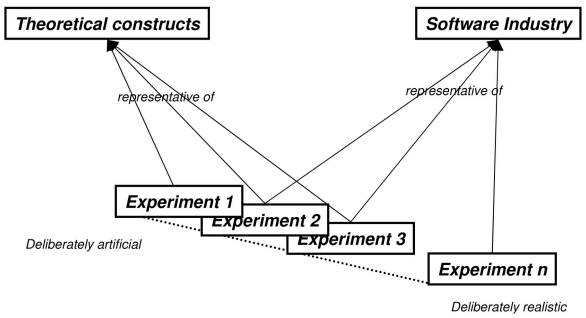

Fig. 4. Applying theory to field by progressive experimentation.

specified in theories are, by nature, incomplete in the sense that a cause is usually neither necessary nor sufficient for the effect to occur.[4]

All this means that extratheoretical knowledge is required to put a theory to work and this requires a lot of training and experience [38]. In short, by their very nature, understanding and explanation never apply directly to the field but "must be applied intelligently, making all the adjustments that are required from case to case" [38]. Thus, deciding whether anchoring applies to a specific estimation process is a matter of experience and, perhaps, intuition.[5]

Perhaps, a more appealing scenario, at least for experimentalists, is the one depicted in Fig. 4. Here, the link between theory and field is not argued for directly but is demonstrated by a range of experiments. In the figure, the experiments toward the left relate more strongly to theoretical constructs and the ones toward the right relate more strongly to the software industry, while the ones in between have some situational variables that relate to theory and some that relate to the industry. In the leftmost experiments, artificial design elements may be appropriate and, in the rightmost experiments, realistic design elements may be appropriate. (Of course, other types of study, in addition to experiments, are also highly relevant and not only on the realistic end of the scale.)

It is important to realize the theory centeredness in the scenario in Fig. 4: The more realistic and representative of the software industry an experiment is, the more specific it becomes and, hence, the less general it becomes. Thus, the potential for generality lies in the abstraction of theory, while the series of experiments simply demonstrates the viability of this potential in terms of ever more concrete exemplars.

Although there might seem to be a temporal implication that artificial experiments should be done prior to realistic ones, this need not be the case. Artificial experiments may be conducted in response to a desire to investigate some phenomenon at a deeper level than what might have been done previously. This should be particularly relevant for the current state of empirical software engineering.

An instance of the scenario in Fig. 4 is given in [106]. Waller and Zimbelman present a bridging strategy for generalizing theoretical propositions via the laboratory to field settings by conducting experiments with different degrees of artificiality/realism. It is not expected that one and the same experiment should cover the whole spectrum. Thus, for experiments that are focused entirely inward toward theory, realism is only an issue in as much as it is not present: "Experimenters suppress mundane realism, employing generic tasks and settings that do not resemble 'real-world' tasks and settings but nevertheless activate the ... mechanism underlying [the theory]" [106]. On the way toward validating theoretical propositions in the real world, other experiments may retain certain artificial aspects while seeking realism with respect to others and, finally, studies that are entirely focused on realism (and holism) may be conducted to complement the other studies.

2.2.5 Holism, Ecology, and Complex Systems

The arguments in Sections 2.2.1-2.2.4 for adding deliberate artificiality rely on the assumption that the world may be analyzed in a reductionist fashion. This assumption, along with the usefulness of artificiality, is challenged by several schools. Some of the more systematic views on the issue come from directions inspired by *probabilistic functionalism*, a direction in psychological research founded by Brunswik [41]. These directions, which study many of the social and behavioral processes relevant to empirical software engineering, are skeptical of the usefulness and even the possibility of isolating mechanisms and phenomena. Furthermore, they are skeptical of Fisher's factorial design [33] (the experiment design usually adopted in empirical software engineering) because it imposes the principle of orthogonality of causes on research. While appropriate for some disciplines (e.g., agriculture, which motivated Fisher's views), the assumption of orthogonality is seen as inappropriate for disciplines in which the researcher has little control over the environment such as software engineering.

Rather, the appropriate object of study for probabilistic functionalists is an *ecology*: "the organism-environment system as a whole, where the behavior of the organism is molded by the forces of adaptation and intertwined with the properties of the environment" [57, p. 404]. In ecological models, the holistic intertwining of the object of study with its environment is accompanied by entangled probabilistic relationships, which is "causally ambiguous, because the relationships between cause ... and effect ... are uncertain in both directions ..." [23] (see also [22], [40]). The causal ambiguity inherent in ecological models makes isolating causes less meaningful.

Moreover, the central methodological tenet for generalizability in probabilistic functionalism is *representative design*, which entails sampling from an entire ecology, that is, sampling all relevant situational variables (not just subjects). Note that this idea is apparent in Shadish et al.'s insistence that one should ensure construct and external validity for all the situational variables (Section 2.1).[6]

4. Mackie makes the point succinctly by stating that a cause is an INUS condition, that is, "an *insufficient* but *necessary* part of a condition, which itself is *unnecessary* but *sufficient* for the result" [72]. For example, a match is an INUS condition for lighting a fire. For the strategy of lighting a fire with a match, the match is insufficient on its own but is necessary, and lighting a fire by this strategy is not the only way (unnecessary), although it is sufficient.

5. Intuition in the sense of Hogarth [47] and Dreyfus and Dreyfus [29] is a result of experience and learning.

6. Campbell was Brunswik's student.

Hammond states that representative design "simply means that the circumstances of an experiment should be representative of the conditions to which the result of the experiment are intended to apply" [39], which is more or less the message for empirical software engineering in [93].

Thus, the holistic arguments of probabilistic functionalism and its focus on the representativeness of the real-world leave little room for artificial design elements. Furthermore, it is easy to see that software engineering may be described in holistic terms: Any stage of software development can be seen to be comprised of a multitude of entangled psychological, sociological, and technological interactions. The question then becomes: How may the uses of artificiality that we discussed in Sections 2.2.1-2.2.4 be defended in light of this?

One answer may be seen by considering *complex systems*. Complex systems are systems (e.g., physical, biological, artificial, and social) that have a large number of parts with many interactions [92]. These interactions may have all sorts of effects on overall causal relationships, including cancellation, enhancements, and reversals [91, chapter 12]. In addition, *emergent* phenomena may arise which, in one sense or another, are more than their constituent parts [7], [9].

Simon [92] distinguishes between *strong* emergence and *weak* emergence. The former entails a creative principle where the effects of strong emergence cannot be deduced from one of the interacting mechanisms alone and, hence, this principle does not lend itself to a reductionist explanation. Weak emergence, on the other hand, is also synergistic, but, although its effects might be nontrivial, it is still possible to deduce weakly emergent phenomena from properties of the individual parts. An example, drawn from [92], is an object's pull of gravity on another object. This phenomenon cannot be studied without the presence of at least two objects, even though gravitation is a property of isolated objects. Weak emergence remains in the realms of reductionism. Simon further argues that complex systems (at least those that are within our capabilities to fathom) have inherent hierarchical structure in such a way that they are "nearly decomposable systems." This means that complex systems may be seen as being composed of subsystems that have high-frequency interactions within themselves but have low-frequency interactions between them. As an approximation, it is therefore possible to reason about a given subsystem disregarding other subsystems or its role in the system as a whole.[7] Simon suggests that this inherent structure is an evolutionary necessity not just in the biological sense but as a general principle of stability and fitness for systems that evolve and survive (such as the social and technological systems relevant to software development). The simplification that the approximation of near decomposability gives us suffices to make complex systems tractable in a reductionistic manner.[8] Simon states, "For this reason, we should not despair of unraveling the web of causes" [92, p. 207].

To summarize, the reality of software development is certainly complex and is comprised of multiple intermingled relationships. Nevertheless, there are substantive arguments in favor of factoring out and studying (by using deliberate artificiality) subunits of causal relationships from the whole, at the very least, because of the difficulty of conducting studies and building understanding solely by holistic approaches.

3 Attitudes and State of Practice

Section 2 discussed analytically how situational artificiality may be used constructively in empirical software engineering. In order to get an impression of attitudes and practices regarding situational artificiality in empirical software engineering, we also carried out a qualitative content analysis of articles describing software engineering experiments. Our research questions for the analysis were the following:

1. *What are the attitudes toward situational realism versus situational artificiality in experiment design and analysis*?
2. *What is the state of practice with regard to deliberate artificial design elements*?

3.1 Method of Analysis

The objective of a content analysis is to elicit semantic content from texts in a systematic manner. Qualitative content analysis [62], [36] draws on the ideas of classical quantitative content analysis [63]. Whereas quantitative content analysis is based on (mechanical) word counts, qualitative content analysis relies on semantic analyses of larger portions of text. Central to both approaches is a stepwise process of abstraction from the text base to the message or meaning expressed in the text. This involves abstracting text passages (the recording units) into prototypical sentences (codes) and then categorizing the codes into thematic categories that represent semantic meanings. We follow the steps of content analysis as summarized in [85].

3.1.1 Sampling Strategy

The very first step is to determine the material to be analyzed. Our focus in this paper is on experiments. We therefore assessed all 103 articles[9] found to be describing experiments identified by Sjøberg et al. [96] from a total of 5,453 articles published in nine leading software engineering journals and three conference proceedings from the decade 1993-2002. These journals and conference proceedings were considered to be leaders in software engineering, in particular empirical software engineering. The journals are listed as follows: *ACM Transactions on Software Engineering Methodology* (*TOSEM*), *Empirical Software Engineering* (*EMSE*), *Computer*, *IEEE Software*, *IEEE Transactions on Software Engineering* (*TSE*), *Information and Software Technology* (*IST*), *Journal of Systems and Software* (*JSS*), *Software Maintenance and Evolution* (*SME*), and *Software: Practice and*

7. In quantum electrodynamics, Feynman diagrams model interactions between photons and electrons in an infinite recursion. Thus, quantum electrodynamics is a complex system with infinite feedback loops. Computation would be impossible were it not for the fact that successive iterations of feedback at increasing recursive depth cancel each other out to a tolerable degree so that local computation on subsystems suffices [31]. Simon applies this principle to social systems as well [92].

8. Approximation based on inherent structure is the key here. This is very much in the spirit of modern-day science, where models are sufficient approximations "as substitutes for a complete understanding that science may not be able to attain" [88, p. 70].

9. Due to limitations of space, the 103 articles are not referenced. However, the data analysis and catalog of the articles may be provided upon request to the corresponding author.

Experience (*SP&E*). The conferences are the International Conference on Software Engineering (ICSE), the IEEE International Symposium on Empirical Software Engineering (ISESE), and the IEEE International Symposium on Software Metrics (METRICS).

The term "experiment" is used in an inconsistent manner in the software engineering community. Furthermore, it is often used synonymously with the term "empirical study." Therefore, Sjøberg et al. [96] defined a *controlled experiment in software engineering* as a study in which individuals or teams (the experimental units) conduct one or more software engineering tasks for the purpose of comparing different populations, processes, methods, techniques, languages, or tools (the treatments). Randomized experiments (in which units are assigned randomly to treatments) and quasi-experiments (in which units are assigned nonrandomly to treatments) in the sense of [91] were both included because both experiment designs are widely used in empirical software engineering [66]. In this paper, we consistently use the term "experiment" in the above-mentioned sense of "controlled experiment." The process of selecting articles was determined on the basis of predefined criteria, as suggested in [60] (see [96] for full details).

3.1.2 Recording Unit

Our recording units, that is, the basic units of text chosen for analysis, were passages of text (sentences or paragraphs) that ostensibly bore relevance to the thematic categories (A-C) below. Although the entirety of an article was scanned, a particular emphasis was placed on sections that described experiment design, threats to validity, discussions, and conclusions.

3.1.3 Themes

Content analyses may relate to predefined themes. Alternatively, they may be exploratory, where themes are defined as a result of the analysis. We used the former approach, defining themes of interest that provided the context for our analysis on the basis of the research questions stated above. As this is often necessary in initial research, we decided to use 15 percent of the material to test the reliability of the themes (hence introducing an "inductive" element [75]). As a result, one of the themes was phrased more precisely before all articles were analyzed according to the revised set of themes. The themes are listed as follows:

A. *Focus on realism.* The design and rationale of the experiments are discussed and assessed according to immediate likenesses to industrial situations.
B. *Focus on understanding.* The design and rationale of the experiments are discussed and assessed according to an increased understanding of the phenomena under investigation, despite situational artificiality.
C. *Deliberate artificial design elements.* The design and rationale of the experiments are discussed and assessed according to deliberate artificiality in one or more of the following respects:
   - *isolation of basic mechanisms*,
   - *existence of phenomena*,
   - *persistence of phenomena*,
   - *relating to theory*, and
   - *ecology and holism* (in contrast).

These themes are intended to categorize phrases and, hence, articles into those that employ deliberate artificial design elements (Theme C) and those that do not (Themes A and B). Among the latter, we wish to distinguish between two kinds of phrase/article: those that simply view artificiality as a threat because it compromises likenesses to industry (Theme A) and those that hold that or investigate whether an increased understanding is possible, in spite of situational artificiality (Theme B). The three themes express three quite distinct attitudes toward situational artificiality. Texts on content analysis state that categories should be exhaustive and mutually exclusive. However, high-level themes such as the above may be exempted from this requirement because they categorize the so-called latent meanings of a text, which may be several, and, so, a phrase may express more than one theme [63].

3.1.4 Conducting the Analysis

The above themes (A-C) served as targets for the content analysis. The semantic content of text passages was determined using a bottom-up process (e.g., see [36] for examples and references). Each text passage that satisfied the definition of a recording unit (Section 3.1.2) was analyzed for its *manifest content* (the visible content), which was extracted and expressed in short form. These short forms were then used to abstract codes, that is, headings under which the passages may be summarized. The resulting codes are listed in Table 2. Our next step was to map the codes in Table 2 to the predefined themes: codes 1-5 map to Theme A, codes 6-11 map to Theme B, and codes 12 and 13 map to Theme C.[10]

The first author analyzed all of the articles. The second author then independently analyzed a random selection of 33 (29 percent) of the articles. Of these 33 articles, nine (27 percent) were discussed with respect to matters of interpretation (eight) and oversights (one). The discussions concerning matters of interpretation were prompted by subtle differences in our interpretations of the meaning of certain passages. The net adjustment of the first author's categorization for the selected 33 articles was an addition of one count to Theme A.

3.1.5 Limitations

The main limitations of this study are bias with respect to the selection of publications and threats to reliability. The former is addressed in [96].

Threats to reliability address the interpretative process of the content analysis. Reliability consists of *stability* (whether the process is stable over time), *reproducibility* (whether other researchers may replicate the process), and *accuracy* (whether the process yields what is specified) [63]. Stability is assessed by replicating the study and accuracy is assessed relative to another process which is assumed to be correct.

10. In contrast, in a purely exploratory content analysis, the codes would give rise to categories, which would then be the bases for creating themes that describe the analyst's high-level interpretation of whatever is being analyzed. Thus, the themes would be generated primarily from the data rather than from research questions, as in the case here.

TABLE 2
Codes Generated from Text Passages

1) We succeeded in creating situational variables that correspond to those of industry.
2) The situational variables pose threats to external validity, because they are not like those of industry or because they are artificial.
3) The situational variables are generic and therefore pose threats to external validity, because they are not like those of industry.
4) Future studies should be more realistic.
5) Steps were taken to avoid threats to external validity; in other words, to avoid artificiality.
6) Contrary to common beliefs regarding a certain threat to external validity, our results show that this threat is not a threat.
7) Although some of the situational variables do not correspond to those of industry, the observed treatment-outcome covariations are expected to hold in industry.
8) The purpose of the study is to investigate concepts central to research.
9) This study with artificial elements is a first exploratory step to refine experimental design and to identify factors that are relevant for further research.
10) Specific (but possibly realistic) situational variables were chosen in order to obtain stronger experimental results.
11) One goal was to investigate the differences between realistic and artificial variables.
12) Situational variables were deliberately made (more) artificial in order to obtain a certain type of knowledge.
13) A holistic approach is chosen because a reductionist approach does not capture the infinite combinations of interactions in real systems.

Both of these measures are difficult to assess at present, although accuracy is somewhat supported by the rate of agreement indicated in Section 3.1.4. Hence, reproducibility is the aspect of reliability that is most relevant to this present study.

Interpretive research is a product of the objects of study and the persons who interpret them. As such, content analysis is a systematic method for determining the reviewers' perceptions of a text. Although several meanings may be elicited from a given text, content analysis should yield reproducible results within a given interpretative framework. In our case, this framework was given by 1) the predefined themes (Section 3.1.3), 2) the motivation for our analysis, and 3) our expectations of the results. Thus, our motivation was to bring into discussion the possibilities of deliberate artificiality and our expectations were that the overwhelming majority of the articles would have passages that we would classify under Theme A, few would have passages that we would classify under Theme B, and very few would fall under Theme C. Reproducibility is threatened if it is not possible for other researchers to enter the same interpretive framework as that of the current study or if the motivation and expectations are seen as unreasonable. The main threat in these respects is that both reviewers had similar motivations and expectations, although they have diverse backgrounds as researchers. We therefore encourage other researchers to replicate this study.

3.2 Findings

We found that 73 of the 103 reviewed articles had text passages pertaining to Theme A, 16 articles had passages pertaining to Theme B, and seven had passages pertaining to Theme C. Nine articles had passages from both Themes A and B and six articles had passages from both Themes A and C. One article had passages from both Themes B and C. Thus, in total, we found that 80 of the 103 reviewed articles discuss the issues of realism, representativeness, or artificiality according to the three themes. Table 3 summarizes.

The trends for each theme over the years covered by the analysis are indicated in Fig. 5. It seems evident that the number of articles that exhibit a theme follows the trend of the total number of articles that describe experiments for a particular year. Our data does not suggest that any of the themes gained or lost in prevalence over the years.

In order to see whether there was a higher prevalence of a particular theme in a certain research area or within a certain community, we performed similar analyses for software engineering topics and for authors and affiliations. Software engineering topics were defined in two ways: 1) according to the IEEE Keyword Taxonomy [54], which is an extended version of the ACM Computing Classification System [2], and 2) according to a scheme developed by Glass et al. [35] (see [96] for details). As observed in [96], the two prominent areas of software engineering in which experiments are conducted are *code inspections and walkthroughs* (35 percent of the 103 article) and *object-oriented design methods* (8 percent of the 103 articles), while the numbers of experiments are limited for other areas. No particular prevalence was evident in any topic. If anything, the number of articles that exhibited any of our themes within a topic seemed to be determined by the total number of articles that described experiments for that topic. Nothing of interest emerged for authors or affiliations.

Passages that were relevant to the themes were found in all parts of an article: external validity sections, design sections, conclusions, introductions, abstracts, and even footnotes. Our general impression is that very few articles had up-front discussions regarding the role of artificiality as such. We now describe our findings in more detail.

3.2.1 Focus on Realism

Categorizations under Theme A were typically due to discussions of threats to external validity. A typical phrase is: "One of the limitations of this study is the use of student subjects. Hence, one should be careful in generalizing the results of this experiment to professional programmers" [13]. Several articles claimed to have succeeded in creating realistic situations: "With respect to external validity, we took a specification from a real application context to deal with an inspection object that was representative of an industrial development situation. Moreover, we used

TABLE 3
Number of Articles That Discuss Themes A-C

Theme(s):	A	B	C	A+B	A+C	B+C	A+B+C
Articles that discuss:	73	16	7	9	6	1	0

Total number of articles that discuss themes: 80

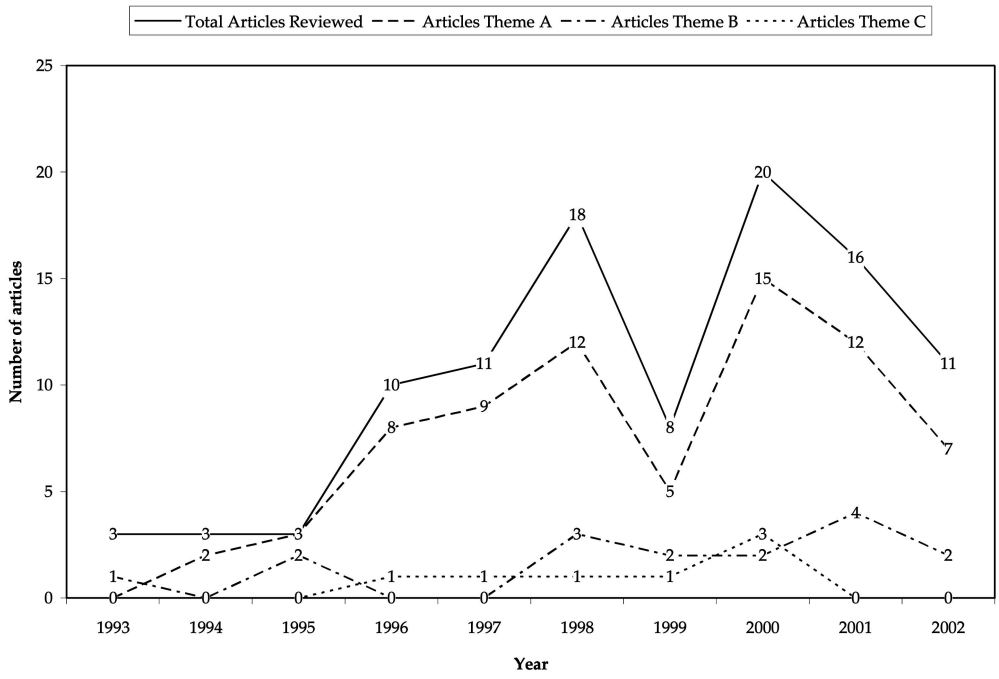

Fig. 5. Articles classified to themes per year.

inspection activities that had been implemented in a number of professional development environments" [15] or "The experiment was conducted with professional developers and with documents from an industrial context, so these factors should pose little threat to external validity" [11]. Some articles argue that certain situational variables that are traditionally viewed as being artificial are, in fact, realistic for the purpose at hand: "The subjects [who were students] had some prior experience with process-oriented modeling; this is precisely the population [that] we sought to generalize to. The existing information systems workforce in [the] industry today has prior experience in PO modeling, and it is important to gain insights into the relative performance of such individuals with the newer OO models" [4].

Two particularly telling examples are the following quotations from external validity discussions: "The results for the generic documents cannot be generalized to the [organization-specific] documents due to the difference in nature between the two sets of documents. The results for the [organization-specific] documents, on the other hand, may be valid since we used parts of real [organization-specific] documents," [11] and "Clearly, the results for the generic documents cannot be generalized to specific application domain documents of the organization" [64]. Here, external validity is viewed as being dealt with through the use of the specific documents of an industrial partner and, at the same time, generic documents are seen as less generalizable. Another example expresses much the same: "Our study was performed with subjects and code documents from a single organization. While this enjoys greater external validity than doing studies with students in a 'laboratory' setting, it is uncertain the extent to which the results can be generalized to other organizations" [65].

These examples, together with the large amount of material classified under Theme A, illustrate four things: First, there is a strong focus on imitating industrial situations in experiments. Second, artificiality is overwhelmingly perceived as posing a threat to generalization and is therefore something to be minimized.

Third, "external validity" seems to be used synonymously with "generalizability," although external validity is but one aspect of generalizability together with construct validity. In addition, it is assumed that conducting an experiment in an industrial setting will increase external validity automatically. Fourth, generalizability is viewed overwhelmingly in terms of *surface similarity* (principle 1) and in terms of statistical generalization.

Given that the external validity of an experiment is understood as the robustness of inferences across *variants* of situational variables [91], then conducting an experiment in a specific industrial context, at the outset, more likely affects external validity adversely (when arguing in terms of surface similarity) because the results do not transfer to other organizations unless the situational variables of those organizations are very similar to those of the experiment. Therefore, the use of specific industrial situations demands explicit justification with regard to external validity and generalizability as much as the use of design elements of lesser degrees of realism does.

Generalization relies on both external validity and on construct validity. Issues of external validity were discussed in 66 of the 103 articles [96], but construct validity was discussed in only 15 articles. (Note that we reviewed whether authors *discuss* these validity issues and not the quality of the studies with regard to these issues.) Construct validity is necessary for relating specific experimental situations to constructs on a level sufficiently abstract to allow the transfer of results beyond the specific. The lack of

TABLE 4
Number of Articles That Discuss Deliberate Artificial Design Elements (Theme C)

Subtheme:	Isolation of Basic Mechanisms	Existence	Persistence	Relating to Theory	Ecology and Holism
Articles that discuss:	2	3	2	0	1

One article discusses both Existence and Persistence.
Total number of articles that discuss Theme C: 7

discussions of construct validity therefore corroborates our general impression that generalization is intended at a superficial level only. (There was no evident trend in the amount of discussions of construct validity being mentioned over the 10-year period.)

3.2.2 Focus on Understanding

Passages from Theme B represent a shift of focus away from realism. Instead, a certain emphasis is placed on understanding phenomena, in spite of experimental situations that may not be realistic or representative of industry. Examples of passages in Theme B are: "The specific purpose of these experiments is to produce empirical results to support some of the concepts central to the research," [44] and "It is important to stress that the scope of this research is novel and ambitious and that the current results comprise more of a 'proof of concept' than a set of findings to be widely used by the community of practitioners. ... we make no claims for 'truth', rather we present some results that others might attempt to refute, thereby extending knowledge and understanding in the field" [104].

Some of the passages acknowledge the idea that generalizability need not depend on the particular situational variables administered in an experiment: "Experiments in a student setting can always be questioned concerning validity in an industrial environment. In this case, this is not regarded as particular critical as one objective of the course is to model an industrial environment. In particular it should be noted that the study is based on comparison of different methods for effort estimation and the evaluation should provide similar results independent of the environment (university or industry)" [80]. Note that this relies on principles of generalization other than those that depend on immediate likenesses.

3.2.3 Deliberate Artificial Design Elements

Seven articles had passages that we categorized under Theme C. This theme takes the modest appreciation of artificiality expressed by Theme B to a more active level in that it pertains to issues of artificiality as a deliberate means of acquiring knowledge or takes an active position against artificiality. Table 4 summarizes the findings for Theme C.

Isolation of basic mechanisms. Among the 103 reviewed articles, we found two articles that stated explicitly that artificiality was introduced in order to isolate mechanisms. The first article states that "comprehension may be affected by factors other than [the] type of construct used. In this experiment, an attempt was made to minimize these factors," [13] and then goes on to describe the deliberate situational artificiality introduced into tasks and materials to minimize these other factors: "Both code segments were very small, the programming style was the same, and procedure and function name tokens were eliminated in both" [13]. The second article introduced artificiality only in materials: "Because the focus of the study was on effectiveness and efficiency of testing, the spreadsheets contained no faults. This may be unrealistic; however, including faults in the spreadsheets would have confounded the data about testing effectiveness and efficiency since the subjects would not be focused on the single task of testing the spreadsheets" [89].

In addition, we found one article that argued in favor of artificial design elements post hoc: "The software systems used for the experiments were not large and may not be representative of real software systems. ... it may be that to control and isolate the effect of inheritance on the maintainability of object-oriented software, small systems are required otherwise the effect may become too difficult to detect" [26]. However, this afterthought does not, in our opinion, represent a deliberate act. One article also describes modifying tasks in a manner related to our discussion: "Both groups then 'solved' a system design problem that was intentionally designed to be data intensive" [53]. In order to investigate the theory of cognitive fit [103], [3], a real-world problem was made more data intensive (and, hence, less process intensive) so as to act as a match for a data-centered methodology in comparison to a process-centered methodology. However, although a situational variable is modified, it is not clear that the variable becomes artificial in the sense of becoming unrealistic (Section 2). These two examples were therefore not included in our analysis.

Three articles explicitly suggest (future) steps to investigate basic mechanisms (but without mentioning deliberate artificiality for isolating them.) One states: "We will attempt to separate the effects of some external sources of variation from the effects due to changes in the process structure ... We hope to identify mechanisms that drive the costs and benefits of inspections so that we can engineer better inspections" [81]. Another states: "A more appropriate hypothesis may be that a three-step process (playing, teaching, playing) is more successful than just teaching" [28]. The third claims: "A comprehensibility index or metric would have to involve the factors of naming, commenting and structure, as well as others, but it is not clear, a priori, what the relative weights should be for these factors in the overall index. We suggest that such weights are best determined as a result of measurement of performance evaluations on the target group of users" [32].

Thus, there seems to be some awareness of the usefulness of identifying basic mechanisms and even some awareness that introducing situational artificiality deliberately may be a way of accomplishing this. However, there seems to be a general lack of actual investigation of basic principles (with or without deliberate artificiality). Relative

to the top-level frame of reference of *determining what the effects of applying a given software engineering technology used by certain developers in a given setting are* (Section 2.2.1), one may ask where, in the "hierarchy of basic mechanisms," software engineering experiments are situated at present. It is not trivial to operationalize the concept of this hierarchy, but our impression, after examining the research questions in the 103 articles describing experiments, is that the experiments are at or very near the top-level, that is, underlying mechanisms are rarely investigated.

This impression is corroborated by findings on the use of theory in the same set of articles [42] since basic mechanisms are essential ingredients in explanatory theories. Only 24 of the 103 reviewed articles employ theory in the service of explaining the investigated cause-effect relationships. Of these, only three actually evaluate theories and thereby potentially investigate basic mechanisms. The impression that there is very little further investigation into underlying explanatory basic principles is also supported by the fact that all but a very few theories are used in more than one article [42].

Existence of phenomena. Among the 103 reviewed articles, we found three that designed the experiment with the explicit purpose of demonstrating the existence of phenomena. For example: "The manipulations chosen for this experiment were designed to maximize our 'signal-to-noise' ratio in testing the relationships among the constructs specified in our model" [59]. The experimenters deliberately administered an extreme but favorable treatment that is atypical of everyday software development. Another example is this: "There is some question about the level of industrial usage of the code reading technique employed. Inspection techniques in industry tend to be less formal, but consequently less easily taught, and their successful application requires a significant amount of experience. For this reason it was felt that the subjects would perform better with a technique that is more methodical to apply and hence the code reading technique was kept" [87].

In addition, three articles argue in terms of the existence of phenomena in postexperiment comments. These arguments are defensive in nature and do not imply a purposeful introduction of conditions that are favorable for demonstrating the existence of a phenomenon. Here is an example: "One possible explanation for the poor performance of the control subjects is that they were college students, not professional programmers. The goal of the experiment, however, was to demonstrate improvement due to the treatment condition" [74].

Persistence of phenomena. Two of the reviewed articles argue in terms of deliberate situational artificiality to show the persistence of phenomena (generalizing from unfavorable to favorable conditions): "If the effort experience base [the method under investigation] does not work when the projects are almost identical, then it is hard to believe that it will work in an industrial environment (where hopefully all projects are unique in some way)" [80]. The second article has a paragraph that argues in terms of both persistence and existence: "The programming task, while perhaps unrepresentative of real-world conditions, was designed to address the effect of treatments whose outcomes should be similar irrespective of the simplicity or complexity of the task. That is, if exception handling coverage on this admittedly simple task is poor, one can hardly hope that it would be much better on programs of higher complexity. Hence, the programming task used in this study serves as a barometer that warns against expecting substantial improvement in the real world and, concomitantly, implies that steps taken to increase exception handling coverage in simple programs would be similarly effective on more complex ones" [74].

In addition, six articles give defensive or post hoc arguments that relate to the persistence of phenomena. Four use these arguments to defend the design of the experiment ("erring on the conservative side"). For example, an article that investigates estimation judgments based on fallible information states: "The inclusion of such additional cues [as found in a realistic setting] would not have altered the fundamental experimental task, namely, weighing two sets of unreliable information (initial estimates and status information) in estimating the project team's software productivity" [1]. Another example is this: "Hence, although our cost-benefit results may not be as good with students as with professional developers, our findings are conservative with respect to the calculation of cost-benefit levels" [15]. Of the two remaining articles that argue post hoc, one concludes in retrospect: "Even in the case of a highly contrived problem ..., which was essentially designed to be dealt with by a specialisation solution, subjects still seemed to find the flat version easier to work with" [21]. The experiment is not described as having introduced artificial treatments (problems) deliberately in order to obtain a specific kind of knowledge. Nevertheless, the observed failure of a treatment that presumably favors a specialization solution implies (assuming validity) that a realistic problem will stimulate such a solution even less.

Relating to theory. Although 24 articles use theory to explain the investigated cause-effect relationship [42], none argues that artificiality is an asset when relating to theory. (However, one of the articles that explicitly mentions the deliberate introduction of artificial design elements for demonstrating existence also uses theory.)

Moreover, among the 24 articles that use theory, all but seven discuss experimental external validity relative to the software industry (while only four consider construct validity). Hence, one must assume that most researchers intend to generalize to the real world rather than relate to the constructs of theory in the sense of Section 2.2.4.

Interestingly, the seven articles that do not discuss experimental external validity all have in common that they refer to theories that deal with cognition, which is, one could argue, a topic that is relatively independent of the specific type of subjects and settings in an experiment. Two of the seven articles argue that students are appropriate for such tasks. One refers to [49] and states: "Advanced students and professional programmers are statistically similar in terms of comparing their mental representation and various performance measures" [44]. The other refers to [102] and states: "Novice analysts are not biased by experience with other methodologies and have not had time to adopt a personal

'favorite' methodology" [53]. However, only one of the seven articles discusses construct validity.

Theories are not used for providing a theoretical framework or paradigm within which studies are conducted and interpreted. Rather, most of the theories are used somewhat locally to support and motivate the study rather than the study being conducted as a result of theoretical deliberations [42]. This lack of theory centeredness may be one reason that authors seemingly do not relate their experiments to theory in the strong sense of Section 2.2.4 (using deliberate artificial design elements).

Ecology and holism. One article argues in holistic terms: "In this research, the focus is upon studying diagram clarity in realistic situations. Therefore, we chose a holistic approach rather than a reductionist approach. In a holistic approach, the system (or the model of a system) is tested without decomposing it into component parts" [67]. Furthermore, they refer to [109] and state that "a reductionist approach is fruitless for investigating realistic systems because differences are due to an infinite combination of interactions. In the present study, the holistic approach yields the important benefit of increasing ecological validity ... via studying the actual, complete diagram versus a more artificial, piecemeal, diagram. Employing a holistic approach, a researcher is afforded the opportunity to combine and apply highly reliable and valid findings inside a contextually valid experimental design" [67].

One article argues holistically but with the prospect of future reduction: "Rather than studying each principle separately, we examine the Coad and Yourdon principles as a whole for a number of reasons: ... As a first step, we want to see whether the application of these principles as a whole [has] any practical significance before designing more complicated experiments where the various principle effects would have to be isolated from each other" [17].[11]

3.3 Focus on Multimethod Research

Many arguments in favor of artificial design elements emphasize that artificiality and realism should complement each other and be introduced in various degrees over several studies. Hence, we were interested in whether the reviewed articles expressed perspectives of this nature. However, passages to this effect were heavily confounded with (very frequent) general phrases regarding the view that laboratory research should be complemented by more realistic studies and it was difficult to determine whether these phrases pertained to artificiality that was deliberately introduced. Therefore, we refrained from doing a structured analysis of this issue.

However, here is one insightful, albeit atypical, example: "Student based experiments can provide useful results for several reasons. First, they can be used to focus weak hypotheses on phenomena which appear to be important. These hypotheses can then be tested in more realistic settings with a better chance of important and interesting findings. Second, they can be used as a basis for deciding whether a hypothesis is worth investigating further in, e.g., an industrial case study" [18]. A more typical example expresses concerns for external validity: "All these threats are inherent to running classroom experiments and can only be overcome by conducting replications with people, products, and processes from an industrial context" [14].

3.4 Summary

In summary, the research questions posed at the beginning of Section 3 may be answered as follows: With regard to the state of practice, our findings confirmed our expectations. The overwhelming majority of the articles had passages belonging to Theme A (focus on realism), only a few had passages pertaining to Theme B (focus on understanding), and very few had passages pertaining to Theme C (deliberate artificial design elements).

With regard to attitude, the overwhelming majority of the reviewed articles are concerned about limiting artificiality on the grounds that it is a threat to the external validity of their study. Relating to theory and generalization through theory is also not a primary concern. The main focus is thus on 1) generalizing through arguments of surface similarity (and ruling out irrelevancies in the sense of ticking off these threats as dealt with) and 2) attempting statistical generalization.

4 Discussion

We now discuss the implications of our content analysis in Section 3 in the context of our earlier analytical deliberations in Section 2. We also suggest ways to incorporate artificial design elements in future studies.

4.1 Implications

We believe that the prevailing attitude toward artificiality indicated by our analysis may have had unfortunate consequences for experiments in empirical software engineering (including our own). Generalizing only through immediate likenesses and viewing artificiality as a vice entail that one misses out on valuable opportunities to investigate additional aspects of issues of interest. Indeed, our content analysis reveals that there is a uniformity in the way experiments are being conducted in this respect. A consequence is that new knowledge is not acquired and only certain (superficial) aspects of a problem are revealed. For example, in several areas of empirical software engineering, considerable experimentation has been done and replications (even close ones) in some of these areas continue to produce diverging results [96]. Although this may not be surprising considering the complexity of experimental situations in empirical software engineering, there might also be other reasons for such divergence. Höst et al. observe that "seemingly identical replications of controlled experiments result in different conclusions" and that "this indicates that the research community has not managed to capture the relevant underlying explanatory variables satisfactorily" [51].

11. It should be noted that the specific term "ecological validity" in Brunswik's original formulation denotes the relationship between so-called *distal criteria* (stimuli from the environment) and *proximal cues* (the translation of these stimuli) in a perceiving organism in the environment [22], [23]. However, "ecological validity" is very often used in the sense of "representativeness" or "external validity," a usage that deviates from Brunswik's intention [41]. This is the case for the reference to ecological validity in the quotation above and also for the reference in the following quotation found during our review: "Population validity concerns the generalization of the results to other subjects; ecological validity concerns the generalization of the results to other settings or environmental conditions similar to the experimental setting or condition" [84].

It is vital that experiments in empirical software engineering do not become static repetitions of each other on only one level of inquiry (see also [76]). Hence, the quest for realism should not mask other modes of experimentation. Both the literature and the examples found in our analysis suggest that generalization is not necessarily facilitated by only conducting studies in real situations. Indeed, the role of realism with respect to generalization seems to be misunderstood in several instances. In what follows, we offer some recommendations as to how one might incorporate both realistic and artificial design elements into research.

4.2 The Purpose of One's Empirical Study

Varying degrees of artificiality and realism are appropriate at different stages of empirical inquiry. Concentrating entirely on artificiality is as pointless as expending all one's energy on realism. Therefore, in order to reap benefits from both realistic and artificial design elements, it is essential that researchers view their studies as part of an iterative stepwise process for the refinement and transfer of knowledge in which empirical studies fulfill a wide spectrum of roles.

Conscious decisions regarding realism, artificiality, and the purpose of one's study carry with them the obligation to make conscious decisions regarding the mode by which one wishes to argue for generalization. However, explicit arguments as to if and, if so, by which mode one intends to generalize are seldom given in articles that report software engineering experiments [96]. In particular, this is the case for the articles that we managed to classify according to our themes in Section 3. This increases the obscurity as to the roles of realistic and artificial design elements that are, in fact, used in the various studies.

That empirical studies should play a variety of roles is, of course, something that all empirical disciplines strive for. This is also echoed in empirical software engineering. For example, Juristo and Moreno [56] state that one must first determine a cause-effect model under controlled conditions. Thereafter, one can use in vivo studies on the "early adopters" in the industry (in the language of technology acceptance models [86], [78]). However, more relevant for this discussion are frameworks that relate explicitly to deliberately introduced artificiality. One example already mentioned (Section 2.2.4) is the bridging strategy of Waller and Zimbelman [106]. Also, Davis and Holt [27] describe the relationships among theoretical, experimental, and natural environments and classify experiments into five different types, with decreasing degrees of artificiality: *theory component test*, *theory test*, *stress test*, *search for empirical regularities*, and *field test*. A theory test is conducted on the domain for which a theory gives prediction. This could pertain to the isolation of basic mechanisms and/or to demonstrating the existence of phenomena. A stress test, on the other hand, examines the performance of a theory on more complex (realistic) domains. This might pertain to demonstrating the persistence of phenomena. Theory component tests are conducted in more artificial domains, typically in order to examine why a theory test does not confirm a theory (isolation of basic mechanisms, existence of phenomena). Searches for empirical regularities are not conducted with any close reference to theory and a field test is an experiment in which only treatment variables are controlled and everything else is realistic. The review in [42] suggests that most experiments in empirical software engineering are searches for empirical regularities.

Both frameworks mentioned above involve theory. Artificial design elements are particularly important when building or relating to theory. Although theories for software engineering are not yet common, we think that this is about to change [10], [30], [42], [45], [61], [94], [95], [99], so it is important to be aware of the role of deliberate artificial design elements.

Our analysis shows that there is some awareness of the different roles that empirical studies might play. However, this awareness does not seem to affect how artificiality is addressed in experiments. Furthermore, we have seen that uses of theory do not seem to have affect attitudes toward artificiality.

Experiments need not be theory driven nor do they have to be artificial. However, the position of one's study in relation to theory and artificiality should be the result of conscious decisions. If one uses realistic design elements in a particular study, it is necessary to provide arguments for why realism is essential and to present sound arguments for why artificiality may be detrimental to the study. If one uses artificial design elements, it is necessary to provide convincing arguments for why this is essential and for why realism may be detrimental. What is *not* sufficient is to accept realism as the default alternative without providing specific details of how the realistic design elements will benefit the study.

4.3 Future Directions

We have argued that experiments that contain design elements that are deliberately artificial are useful, but the real test of their utility lies in conducting such experiments. An added incentive is that conducting experiments with artificial design elements need not demand the often vast amount of resources that experiments high in mundane realism demand. Consequently, artificial experiments can often be made small and such small-scale experiments may be integrated into other frameworks such as seminars, field studies, and other experiments.

Our review revealed that it is common to conduct experiments where situational artificiality clearly poses a threat, followed by arguments to compensate with realistic replications. However, a study with validity threats does not become more valid by virtue of future studies. Rather than advocate attempts to compensate for an experiment's inadequacy by conducting realistic replications, we suggest that one use frameworks (e.g., [106], [27]) that incorporate situational artificiality in a deliberate and purposeful manner. If a study cannot, for logistical reasons, fulfill the necessary elements of realism that one's research question demands (and, hence, will have obvious threats relative to this research question), one should instead conduct studies that 1) address research questions that are related by theory or by frameworks such as the ones mentioned above and 2) are suitable for the resources that one has at one's disposal. Using artificial design elements for, for example, the purposes suggested in this paper, gives the opportunity to investigate research questions for which artificiality is not a threat.

In addition, empirical metaresearch should validate the usefulness of artificial situational design elements. Several scholars point to what seems to be a general assumption in many empirical sciences: that mundane realism automatically leads to increased generalizability [20], [68]. Our review indicates that this assumption is widely assumed in empirical software engineering as well. The assumption has, however, come under attack. Campbell states that "a statement to the effect that research is valid because it takes place in a real organization or that a method is invalid because it is used in a laboratory study is no argument at all and is unbecoming [of] a scientist/scholar" [19]. Locke [68] concludes that it is impossible to determine a priori that realism leads to better generalizability. He calls for empirical evidence and initiated comparative reviews of laboratory and field experiments within the three disciplines of industrial-organizational psychology, organizational behavior, and human resource management. (The results indicate a strong correspondence between the two types of experiment with regard to the direction of effect and a slightly lesser correspondence with regard to the size of effect [69].) We are not aware of any such studies in empirical software engineering, although there have been studies that compared the use of students and professionals as subjects [50]. Thus, studies that compare software engineering experiments (and studies of other types) that possess artificial elements (even those introduced by accident or default) with realistic studies should be conducted.

It is also worth considering whether, if a desired phenomenon occurs in artificial laboratory settings, one might not attempt to create these artificial settings in the industry if feasible. This is the reverse of mimicking field settings in experiments. As an example, Ilgen [55] refers to the *teaching machine* of Nash et al. [79], where a learning environment was honed in the laboratory, but, at the same time, the methods were designed to be implementable in the field in specially constructed learning centers. This idea may be applied to software engineering as well.

Devising better methods for designing, for estimating effort, for learning a new technology, etc., demands not only that we descriptively investigate averages but also that we investigate extremes (the best and the worst learners, designers, estimators, etc.) in order to identify what it is that makes the best performers do so well. The study of extremes is, in fact, foundational to the four ways in which we discussed the use of deliberate artificial design elements in this paper. Thus, a shift from descriptive research to research that will produce better software engineering technology (methods, techniques, languages, and tools) demands a greater appreciation of the opportunities offered by experiments that contain deliberately introduced artificial design elements.

5 Conclusion

In disciplines relevant to software engineering, many scholars argue in favor of artificiality in experiments, while, in empirical software engineering, realism is the dominating goal and artificiality is mainly applied and addressed apologetically. Our analysis of attitudes toward situational realism and artificiality in experiments reveals a one sidedness that, in the long run, may have negative consequences for the diversity of knowledge acquisition. We encourage a more balanced view regarding the benefits of deliberately introduced artificiality and the benefits of deliberate realism. Well-designed and well-analyzed artificial experiments need not be viewed as inferior to realistic experiments, provided that the reasons for using artificial design elements are sound.

We have given analytical arguments for and examples of the kinds of knowledge acquisition for which deliberately introduced artificial design elements may be used. We encourage researchers to utilize these and other possible opportunities offered by artificial experiments. Finally, we believe that the usefulness of realistic and artificial design elements can only be understood fully from the perspective of multimethod frameworks that incorporate full ranges of realism and artificiality for various purposes.

Acknowledgments

The authors are grateful to the anonymous referees for their insightful in-depth comments. The authors also wish to thank Chris Wright for proofreading the article, Vigdis By Kampenes, Amela Karahasanović, Ove Hansen, Nils-Kristian Liborg, and Anette Rekdal for their work in extracting articles and data that describe experiments, and Jørgen Busvold and Magnar Martinsen for their assistance in compiling and formatting data.

References

[1] T.K. Abdel-Hamid, K. Sengupta, and D. Ronan, "Software Project Control: An Experimental Investigation of Judgment with Fallible Information," *IEEE Trans. Software Eng.*, vol. 19, no. 6, pp. 603-612, June 1993.

[2] ACM Computing Classification System, http://www.acm.org/class, 2004.

[3] R. Agarwal, "Cognitive Fit in Requirements Modeling: A Study of Object and Process Methodologies," *J. Management Information Systems,* vol. 13, no. 2, pp. 137-162, 1996.

[4] R. Agarwal, P. De, and A.P. Sinha, "Comprehending Object and Process Models: An Empirical Study," *IEEE Trans. Software Eng.*, vol. 25, no. 4, pp. 541-556, July/Aug. 1999.

[5] E. Aronson, T.D. Wilson, and R.M. Akert, *Social Psychology: The Heart and the Mind*. HarperCollins, 1994.

[6] E. Aronson, T.D. Wilson, and M.B. Brewer, "Experimentation in Social Psychology," *The Handbook of Social Psychology,* fourth ed., D.T. Gilbert, S.T. Fiske, and G. Lindzey, eds., chapter 3, vol. 1, pp. 99-142, McGraw-Hill, 1998.

[7] R. Axelrod and M.D. Cohen, *Harnessing Complexity: Organizational Implications of a Scientific Frontier*. Basic Books, 2001.

[8] S.B. Bacharach, "Organizational Theories: Some Criteria for Evaluation," *Academy of Management Rev.*, vol. 14, no. 4, pp. 496-515, 1989.

[9] Y. Bar-Yam, *Dynamics of Complex Systems (Studies in Nonlinearity).* Westview Press, 2003.

[10] V.R. Basili, *Empirical Software Eng.*, editorial, vol. 1, no. 2, pp. 105-108, Jan. 1996.

[11] V.R. Basili, S. Green, O. Laitenberger, F. Lanubile, F. Shull, S. Sorumgard, and M.V. Zelkowitz, "The Empirical Investigation of Perspective-Based Reading," *Empirical Software Eng.*, vol. 1, no. 2, pp. 133-164, Jan. 1996.

[12] V.R. Basili, F. Shull, and F. Lanubile, "Building Knowledge through Families of Experiments," *IEEE Trans. Software Eng.*, vol. 25, no. 4, pp. 456-473, July/Aug. 1999.

[13] A.C. Benander, B. Benander, and H. Pu, "Recursion versus Iteration: An Empirical Study of Comprehension," *J. Systems and Software,* vol. 32, no. 1, pp. 73-82, 1996.

[14] A. Bianchi, F. Lanubile, and G. Visaggio, "A Controlled Experiment to Assess the Effectiveness of Inspection Meetings," *Proc. Seventh IEEE Int'l Symp. Software Metrics,* pp. 42-50, 2001.
[15] S. Biffl, B. Freimut, and O. Laitenberger, "Investigating the Cost-Effectiveness of Reinspections in Software Development," *Proc. 23rd Int'l Conf. Software Eng.,* pp. 155-164, 2001.
[16] T. Boswell and C. Brown, "The Scope of General Theory," *Sociological Methods and Research,* vol. 28, no. 2, pp. 154-185, 1999.
[17] L.C. Briand, C. Bunse, and J.W. Daly, "A Controlled Experiment for Evaluating Quality Guidelines on the Maintainability of Object-Oriented Designs," *IEEE Trans. Software Eng.,* vol. 27, no. 6, pp. 513-530, June 2001.
[18] L.C. Briand, C. Bunse, J.W. Daly, and C. Differding, "Technical Communication: An Experimental Comparison of the Maintainability of Object-Oriented and Structured Design Documents," *Empirical Software Eng.,* vol. 2, no. 3, pp. 291-312, Sept. 1997.
[19] J.P. Campbell, "Labs, Fields, and Straw Issues," *Generalizing from Laboratory to Field Settings,* E.A. Locke, ed., pp. 269-279, Lexington Books, 1986.
[20] J.M. Carlsmith, P.C. Ellsworth, and E. Aronson, *Methods of Research in Social Psychology.* Addison-Wesley, 1976.
[21] M. Cartwright, "An Empirical View of Inheritance," *Information and Software Technology,* vol. 40, no. 14, pp. 795-799, Dec. 1998.
[22] R.W. Cooksey, *Judgment Analysis: Theory, Methods and Applications.* Academic Press, 1996.
[23] R.W. Cooksey, "The Methodology of Social Judgement Theory," *Thinking and Reasoning,* vol. 2, no. 2/3, pp. 141-173, 1996.
[24] L.J. Cronbach, *Designing Evaluations of Social and Educational Programs.* Josey-Bass, 1982.
[25] L.J. Cronbach, S.R. Ambron, S.M. Dornbusch, R.D. Hess, R.C. Hornik, D.C. Phillips, D.F. Walker, and S.S. Weiner, *Toward Reform of Program Evaluation.* Josey-Bass, 1980.
[26] J. Daly, A. Brooks, J. Miller, M. Roper, and M. Wood, "Evaluating Inheritance Depth on the Maintainability of Object-Oriented Software," *Empirical Software Eng.,* vol. 1, no. 2, pp. 109-132, Jan. 1996.
[27] D.D. Davis and C.A. Holt, *Experimental Economics.* Princeton Univ. Press, 1993.
[28] A. Drappa and J. Ludewig, "Simulation in Software Engineering Training," *Proc. 22nd Int'l Conf. Software Eng.,* pp. 199-208, 2000.
[29] H.L. Dreyfus and S.E. Dreyfus, *Mind over Machine.* The Free Press, 1988.
[30] A. Endres and D. Rombach, *A Handbook of Software and Systems Engineering: Empirical Observations, Laws and Theories.* Fraunhofer IESE Series on Software Eng., Pearson Education, 2003.
[31] R.P. Feynman, *QED: The Strange Theory of Light and Matter.* Penguin Science, 1985.
[32] K. Finney, K. Rennolls, and A. Fedorec, "Measuring the Comprehensibility of Z Specifications," *J. Systems and Software,* vol. 42, no. 1, pp. 3-15, July 1998.
[33] R.A. Fisher, *The Design of Experiments.* Oliver and Boyd, 1935.
[34] *Group Processes,* M. Foschi and E.J. Lawler, eds. Nelson-Hall, 1994.
[35] R.L. Glass, I. Vessey, and V. Ramesh, "Research in Software Engineering: An Analysis of the Literature," *Information and Software Technology,* vol. 44, no. 8, pp. 491-506, 2002.
[36] U.H. Graneheim and B. Lundman, "The Challenge of Qualitative Content Analysis," *Nurse Education Today,* vol. 24, pp. 105-112, 2004.
[37] T.M. Gruschke and M. Jørgensen, "Assessing Uncertainty of Software Development Effort Estimates: Learning from Outcome Feedback," *Proc. 11th IEEE Int'l Symp. Software Metrics,* p. 4, 2005.
[38] F. Guala, "Economics in the Lab: Completeness vs. Testability," *J. Economic Methodology,* vol. 12, no. 2, pp. 185-196, 2005.
[39] K.R. Hammond, "Upon Reflection," *Thinking and Reasoning,* vol. 2, nos. 2/3, pp. 239-248, 1996.
[40] K.R. Hammond, T.R. Brehmer, and D.O. Steinmann, "Social Judgement Theory," *Human Judgment and Decision Processes,* pp. 271-312, 1975.
[41] K.R. Hammond and T.R. Stewart, *The Essential Brunswik.* Oxford Univ. Press, 2001.
[42] J.E. Hannay, D.I.K. Sjøberg, and T. Dybå, "A Systematic Review of Theory Use in Software Engineering Experiments," *IEEE Trans. Software Eng.,* vol. 33, no. 2, pp. 87-107, Feb. 2007.
[43] W.L. Hays, *Statistics,* fifth ed. Wadsworth Publishing, 1994.
[44] S.M. Henry and K. Todd Stevens, "Using Belbin's Leadership Role to Improve Team Effectiveness: An Empirical Investigation," *J. Systems and Software,* vol. 44, no. 3, pp. 241-250, Jan. 1999.
[45] J.D. Herbsleb and A. Mockus, "Formulation and Preliminary Test of an Empirical Theory of Coordination in Software Engineering," *Proc. Fourth Joint European Software Eng. Conf./ACM SIGSOFT Symp. Foundations of Software Eng.,* pp. 112-121, 2003.
[46] R. Hogarth, "Beyond Discrete Biases: Functional and Dysfunctional Aspects of Judgmental Heuristics," *Psychological Bull.,* vol. 90, no. 2, pp. 197-217, 1981.
[47] R.M. Hogarth, *Educating Intuition.* Univ. of Chicago Press, 2001.
[48] R.M. Hogarth, "The Challenge of Representative Design in Psychology and Economics," *J. Economic Methodology,* vol. 12, no. 2, pp. 253-263, 2005.
[49] R.W. Holt, D.A. Boehm-Davis, and A.C. Schultz, "Mental Representations of Programs for Student and Professional Programmers," *Proc. Second Workshop Empirical Studies of Programmers,* pp. 33-46, 1987.
[50] M. Höst, B. Regnell, and C. Wohlin, "Using Students as Subjects: A Comparative Study of Students and Professionals in Lead-Time Impact Assessment," *Empirical Software Eng.,* vol. 5, no. 3, pp. 201-214, Nov. 2000.
[51] M. Höst, C. Wohlin, and T. Thelin, "Experimental Context Classification," *Proc. 27th Int'l Conf. Software Eng.,* 2005.
[52] F. Houdek, "External Experiments—A Workable Paradigm for Collaboration between Industry and Academia," *Lecture Notes on Empirical Software Eng.,* N. Juristo and A.M. Moreno, eds., vol. 12, chapter 4, pp. 133-166, World Scientific, 2003.
[53] G.S. Howard, T. Bodnovich, T. Janicki, J. Liegle, S. Klein, P. Albert, and D. Cannon, "The Efficacy of Matching Information Systems Development Methodologies with Application Characteristics: An Empirical Study," *J. Systems and Software,* vol. 45, no. 3, pp. 177-195, Mar. 1999.
[54] IEEE Keyword Taxonomy, http://www.computer.org/mc/keywords/software.htm, 2004.
[55] D.R. Ilgen, "Laboratory Research: A Question of When, Not If," *Generalizing from Laboratory to Field Settings,* E.A. Locke, ed., pp. 257-267, Lexington Books, 1986.
[56] N. Juristo and A.M. Moreno, *Basics of Software Engineering Experimentation.* Kluwer Academic, 2003.
[57] P. Juslin, "Representative Design: Cognitive Science from a Brunswikian Perspective," *The Essential Brunswik,* K.R. Hammond and T.R. Stewart, eds. Oxford Univ. Press, pp. 404-408, 2001.
[58] D. Kahneman, J.L. Knetsch, and R.H. Thaler, "The Endowment Effect, Loss Aversion, and Status Quo Bias: Anomalies," *J. Economic Perspectives,* vol. 5, no. 1, pp. 193-206, 1991.
[59] M. Keil, L. Wallace, D. Turk, G. Dixon-Randall, and U. Nulden, "An Investigation of Risk Perception and Risk Propensity on the Decision to Continue a Software Development Project," *J. Systems and Software,* vol. 53, no. 2, pp. 145-157, Aug. 2000.
[60] B.A. Kitchenham, "Procedures for Performing Systematic Reviews," Keele Univ. Technical Report TR/SE-0401/NICTA Technical Report 0400011T.1, 2004.
[61] B.A. Kitchenham, S.L. Pfleeger, L.M. Pickard, P.W. Jones, D.C. Hoaglin, K. El Emam, and J. Rosenberg, "Preliminary Guidelines for Empirical Research in Software Engineering," *IEEE Trans. Software Eng.,* vol. 28, no. 8, pp. 721-734, Aug. 2002.
[62] S. Kracauer, "The Challenge of Qualitative Content Analysis," *The Public Opinion Quarterly,* special issue on int'l comm. research, vol. 16, no. 4, pp. 631-642, Winter 1952.
[63] K. Krippendorff, *Content Analysis: An Introduction to Its Methodology,* second ed. Sage, 2004.
[64] O. Laitenberger and J.M. DeBaud, "Perspective-Based Reading of Code Documents at Robert Bosch GMBH," *Information and Software Technology,* vol. 39, no. 11, pp. 781-791, Oct. 1997.
[65] O. Laitenberger, K. El Emam, and T.G. Harbich, "An Internally Replicated Quasi-Experimental Comparison of Checklist and Perspective Based Reading of Code Documents," *IEEE Trans. Software Eng.,* vol. 27, no. 5, pp. 387-421, May 2001.
[66] O. Laitenberger and H.D. Rombach, "(Quasi-)Experimental Studies in Industrial Settings," *Lecture Notes on Empirical Software Eng.,* N. Juristo and A.M. Moreno, eds., vol. 12, chapter 5, pp. 167-227, World Scientific, 2003.
[67] K.B. Lloyd and D.J. Jankowski, "A Cognitive Information Processing and Information Theory Approach to Diagram Clarity: A Synthesis and Experimental Investigation," *J. Systems and Software,* vol. 45, no. 3, pp. 203-214, Mar. 1999.

[68] E.A. Locke, "Generalizing from Laboratory to Field: Ecological Validity or Abstraction from Essential Elements," *Generalizing from Laboratory to Field Settings,* E.A. Locke, ed., pp. 3-9, Lexington Books, 1986.
[69] *Generalizing from Laboratory to Field Settings,* E.A. Locke, ed. Lexington Books, 1986.
[70] J.W. Lucas, "Theory-Testing, Generalization, and the Problem of External Validity," *Sociological Theory,* vol. 21, no. 3, pp. 236-253, 2003.
[71] J.G. Lynch Jr., "Theory and External Validity," *J. Academy of Marketing Science,* pp. 367-376, 1999.
[72] J.L. Mackie, "Causes and Conditions," *Causation,* Oxford Readings in Philosophy, E. Sosa and M. Tooley, eds., pp. 33-55, Oxford Univ. Press, 1993.
[73] B. Markovsky, "The Structure of Theories," *Group Processes,* M. Foschi and E.J. Lawler, eds., pp. 3-24, Nelson-Hall, 1994.
[74] R.A. Maxion and R.T. Olszewski, "Eliminating Exception Handling Errors with Dependability Cases: A Comparative Empirical Study," *IEEE Trans. Software Eng.,* vol. 26, no. 9, pp. 888-906, Sept. 2000.
[75] P. Mayring, "Qualitative Content Analysis," *Forum Qualitative Sozialforschung/Forum: Qualitative Social Research,* vol. 1, no. 2, http://www.qualitative-research.net/fqs-texte/2-00/2-00mayring-e.htm, June 2000.
[76] J. Miller, "Replicating Software Engineering Experiments: A Poisoned Chalice or the Holy Grail," *Information and Software Technology,* vol. 47, pp. 233-244, 2005.
[77] D.G. Mook, "In Defense of External Invalidity," *Am. Psychologist,* vol. 38, pp. 379-387, 1983.
[78] G.A. Moore, *Crossing the Chasm,* revised ed. Harper Business, 2002.
[79] A.N. Nash, J.P. Muczyk, and F.L. Vettori, "The Relative Practical Effectiveness of Programmed Instruction," *Personnel Psychology,* vol. 24, pp. 397-410, 1971.
[80] M.C. Ohlsson, C. Wohlin, and B. Regnell, "A Project Effort Estimation Study," *Information and Software Technology,* vol. 40, nos. 11/12, pp. 831-839, Dec. 1998.
[81] A.A. Porter, H. Siy, A. Mockus, and L. Votta, "Understanding the Sources of Variation in Software Inspections," *ACM Trans. Software Eng. Methodology,* vol. 7, no. 1, pp. 41-79, 1998.
[82] A.A. Porter and L. Votta, "Comparing Detection Methods for Software Requirements Inspections: A Replication Using Professional Subjects," *Empirical Software Eng.,* vol. 3, no. 4, pp. 355-379, Dec. 1998.
[83] A.A. Porter, L.G. Votta, and V.R. Basili Jr., "Comparing Detection Methods for Software Requirements Inspections: A Replicated Experiment," *IEEE Trans. Software Eng.,* vol. 21, no. 6, pp. 563-575, June 1995.
[84] S. Ramanujan, R.W. Scamell, and J.R. Shah, "An Experimental Investigation of the Impact of Individual, Program, and Organizational Characteristics on Software Maintenance Effort," *J. Systems and Software,* vol. 54, no. 2, pp. 137-157, Oct. 2000.
[85] C. Robson, *Real World Research,* second ed. Blackwell Publishing, 2002.
[86] E.M. Rogers, *Diffusion of Innovations,* fifth ed. Free Press, 2003.
[87] M. Roper, M. Wood, and J. Miller, "An Empirical Evaluation of Defect Detection Technique," *Information and Software Technology,* vol. 39, no. 11, pp. 763-775, Oct. 1997.
[88] A. Rosenberg, *Philosophy of Science: A Contemporary Introduction.* Routledge, 2001.
[89] K.J. Rothermel, C.R. Cook, M.M. Burnett, J. Schonfeld, T.R.G. Green, and G. Rothermel, "WYSIWYT Testing in the Spreadsheet Paradigm: An Empirical Evaluation," *Proc. 22nd Int'l Conf. Software Eng.,* pp. 230-239, 2000.
[90] W.C. Salmon, "Four Decades of Scientific Explanation," *Scientific Explanation XIII,* Minnesota Studies in the Philosophy of Science, P. Kitcher and W.C. Salmon, eds. pp. 3-219, Minnesota Press, 1989.
[91] W.R. Shadish, T.D. Cook, and D.T. Campbell, *Experimental and Quasi-Experimental Designs for Generalized Causal Inference.* Houghton Mifflin, 2002.
[92] H.A. Simon, *The Sciences of the Artificial,* third ed. MIT Press, 1996.
[93] D.I.K. Sjøberg, B. Anda, E. Arisholm, T. Dybå, M. Jørgensen, A. Karahasanović, E. Koren, and M. Vokáč, "Conducting Realistic Experiments in Software Engineering," *Proc. 18th Int'l Symp. Empirical Software Eng.,* pp. 17-26, Oct. 2002.
[94] D.I.K. Sjøberg, T. Dybå, B.C.D. Anda, and J.E. Hannay, "Building Theories in Software Engineering," *Advanced Topics in Empirical Software Eng.,* F. Shull, J. Singer, and D.I.K. Sjøberg, eds. Springer-Verlag, 2008.
[95] D.I.K. Sjøberg, T. Dybå, and M. Jørgensen, "The Future of Empirical Methods in Software Engineering Research," *Proc. Conf. Future of Software Eng.,* pp. 358-378, 2007.
[96] D.I.K. Sjøberg, J.E. Hannay, O. Hansen, V.B. Kampenes, A. Karahasanović, N.K. Liborg, and A.C. Rekdal, "A Survey of Controlled Experiments in Software Engineering," *IEEE Trans. Software Eng.,* vol. 31, no. 9, pp. 733-753, Sept. 2005.
[97] R. Sugden, "Experiment, Theory, World: A Symposium on the Role of Experiments in Economics," *J. Economic Methodology,* vol. 12, no. 2, pp. 177-184, 2005.
[98] R. Sugden, "Experiments as Exhibits and Experiments as Tests," *J. Economic Methodology,* vol. 12, no. 2, pp. 291-302, 2005.
[99] W.F. Tichy, "Should Computer Scientist Experiment More? 16 Excuses to Avoid Experimentation," *Computer,* vol. 31, no. 5, pp. 32-40, May 1998.
[100] A. Tversky and D. Kahneman, "Judgement under Uncertainty: Heuristics and Biases," *Science,* vol. 185, no. 27, pp. 1124-1131, Sept. 1974.
[101] B. Van Fraassen, *The Scientific Image.* Oxford Univ. Press, 1980.
[102] I. Vessey and S.A. Conger, "Requirements Specification: Learning Object, Process, and Data Methodologies," *Comm. ACM,* vol. 37, no. 5, pp. 102-113, 1994.
[103] I. Vessey and D. Galletta, "Cognitive Fit: An Empirical Study of Information Acquisition," *Information Systems Research,* vol. 2, pp. 63-84, Mar. 1991.
[104] R. Vinter, M. Loomes, and D. Kornbrot, "Applying Software Metrics to Formal Specifications: A Cognitive Approach," *Proc. Fifth IEEE Int'l Symp. Software Metrics,* pp. 216-223, 1998.
[105] S. Vosniadou and A. Ortony, "Similarity and Analogical Reasoning: A Synthesis," *Similarity and Analogical Reasoning,* S. Vosniadou and A. Ortony, eds., pp. 1-17, Cambridge Univ. Press, 1989.
[106] W.S. Waller and M.F. Zimbelman, "A Cognitive Footprint in Archival Data: Generalizing the Dilution Effect from Laboratory to Field Settings," *Organizational Behavior and Decision Processes,* vol. 91, pp. 254-268, 2003.
[107] M. Webster Jr., "Experimental Methods," *Group Processes,* M. Foschi and E.J. Lawler, eds., pp. 43-69, Nelson-Hall, 1994.
[108] D.A. Whetten, "What Constitutes a Theoretical Contribution?" *Academy of Management Rev.,* vol. 14, no. 4, pp. 490-495, 1989.
[109] J. Whiteside, S. Jones, P.S. Levy, and D. Wixon, "User Performance with Command, Menu, and Iconic Interfaces," *Proc. ACM Conf. Human Factors in Computing Systems,* pp. 185-191, 1985.
[110] R.K. Yin, "Case Study Research: Design and Methods," *Applied Social Research Methods Series,* third ed., vol. 5, Sage Publications, 2003.

Jo E. Hannay received the Cand.Scient. degree in computer science from the University of Oslo in 1995 and the PhD degree in type theory and logic from the University of Edinburgh in 2001. He was an IT developer in the insurance industry. He is currently an associate professor at the University of Oslo and a visiting researcher at the Simula Research Laboratory. His interests include the use and development of theories in empirical software engineering, the nature of knowledge that is useful to software engineering, and the development of validated constructs for software engineering. He is a member of the IEEE Computer Society.

Magne Jørgensen received the Dipl.Ing. degree in wirtschaftswissenschaften from the University of Karlsruhe, Karlsruhe, Germany, in 1988 and the Dr.Scient. degree in informatics from the University of Oslo in 1994. He was a software developer, project leader, and manager. He is currently a professor of software engineering at the University of Oslo and a member of the Software Engineering Research Group at the Simula Research Laboratory, Oslo, working on judgment-based software cost estimation.