A Systematic Review of Theory Use in Software Engineering Experiments

Jo E. Hannay, Dag I.K. Sjøberg, *Member*, *IEEE*, and Tore Dybå, *Member*, *IEEE*

Abstract—Empirically based theories are generally perceived as foundational to science. However, in many disciplines, the nature, role and even the necessity of theories remain matters for debate, particularly in young or practical disciplines such as software engineering. This article reports a systematic review of the explicit use of theory in a comprehensive set of 103 articles reporting experiments, from of a total of 5,453 articles published in major software engineering journals and conferences in the decade 1993-2002. Of the 103 articles, 24 use a total of 40 theories in various ways to explain the cause-effect relationship(s) under investigation. The majority of these use theory in the experimental design to justify research questions and hypotheses, some use theory to provide post hoc explanations of their results, and a few test or modify theory. A third of the theories are proposed by authors of the reviewed articles. The interdisciplinary nature of the theories used is greater than that of research in software engineering in general. We found that theory use and awareness of theoretical issues are present, but that theory-driven research is, as yet, not a major issue in empirical software engineering. Several articles comment explicitly on the lack of relevant theory. We call for an increased awareness of the potential benefits of involving theory, when feasible. To support software engineering researchers who wish to use theory, we show which of the reviewed articles on which topics use which theories for what purposes, as well as details of the theories' characteristics.

Index Terms—Theory, experiments, research methodology, empirical software engineering.

1 Introduction

There are many arguments in favor of theory use. Theories offer common conceptual frameworks that allow the structuring of knowledge in a concise and precise manner, thus facilitating the communication of ideas and knowledge. Their level of abstraction enables the generalization of knowledge independently of a specific time and place [5], [66], [91], [95], [98]. Theory is the means through which one may generalize *analytically* [85], [103], thus enabling generalization from situations in which statistical generalization is not desirable or possible, such as from case studies [103], across populations [64], and indeed, often from experiments, especially those in social and behavioral sciences [85], with which experiments in empirical software engineering share essential features.

Such arguments have been voiced in the software engineering community as well, e.g., [6], [27], [43], [57], [89]. A theory provides explanations and understanding in terms of basic concepts and underlying mechanisms, which constitute an important counterpart to knowledge of passing trends and their manifestations. When developing better software engineering technology for long-lived industrial needs, building theories is a means to go beyond the mere observation of phenomena and to try to understand *why* and *how* these phenomena occur.

Thus, theories are of potential use to both researchers and practitioners. However, the usefulness of theories for software engineering is a subject of discussion, and the actual use of theory in empirical studies of software engineering is not well known. Decisions and discussions regarding issues of theory in empirical software engineering must be founded not only on an understanding of what a theory is and how it can be useful, but also on knowledge of the actual use of theories. The main motivation of this article is to contribute to the latter by reviewing the use of explanatory theories for software engineering.

An important undertaking in empirical software engineering is to determine what development technology to deploy and what developers to use in what situations; in other words, interest lies in comparing various technologies and skills and in determining their effects on software development. For this, the experiment is one choice of research method in that it may be applied more or less directly to make such comparisons and to measure such effects.

According to Shadish et al., experiments are suitable for *causal description*, that is, "describing the consequences attributable to deliberately varying a treatment" [85, p. 9], but do not in themselves provide *causal explanation*, that is, "clarifying the mechanisms through which and the conditions under which [the cause-effect relationship] holds" [85, p. 9]. The desire for causal explanation motivates the development of theory. Thus, the focus of this review is on theories that are used to explain, in one way or another, the cause-effect relationships investigated by experiments.

A challenge facing software engineering researchers who are considering using theories to support their research, is

- *J.E. Hannay and D.I.K. Sjøberg are with Simula Research Laboratory, Department of Software Engineering, Pb. 134, NO-1325 Lysaker, Norway. E-mail: {johannay, dagsj}@simula.no.*
- *T. Dybå is with Simula Research Laboratory and with SINTEF ICT, Department of Software Engineering, Safety, and Security, NO-7465 Trondheim, Norway. E-mail: tore.dyba@sintef.no.*

Manuscript received 2 June 2006; revised 6 Sept. 2006; accepted 3 Nov. 2006; published online 28 Dec. 2006.
Recommended for acceptance by D. Rombach.
For information on obtaining reprints of this article, please send e-mail to: tse@computer.org, and reference IEEECS Log Number TSE-0124-0606.

to find theories that are *relevant* for that research. Theory that offers causal explanation is hardly a topic in software engineering textbooks, although a comprehensive compilation of laws and beginnings of theory can be found in [27]. The main contribution of this review is therefore an *overview* of which reviewed articles use which theories and the way they use them, together with characteristics of the theories found. This overview indicates the state of affairs in the different areas of research within software engineering and should provide entry points for those who wish to utilize theory. Although it may be said that all (scientific) thinking involves abstraction and implicit theorizing in one way or another, our focus is on explicit and manifest uses of theory, to which other researchers may relate.

The analysis leading to our findings is firstly in terms of *theory identity*, that is, the determination of something as a theory and its characteristics. By examining theory identity, we wish to give a picture of the types of theory used and from which disciplines theories are taken. Furthermore, we review *theory role*, that is, the various positions (*design*, *post hoc explanation*, *tested*, *modified*, *proposed*, *basis*) theories take on when used to explain the investigated cause-effect relationships in software engineering experiments. Theory and empirical studies must interact if practical knowledge is to be acquired. Although this may seem obvious, there is a lack of consensus in many disciplines on how much weight to place on theory or empirical research [30], [52]. Furthermore, such interactions may take a variety of forms [22], [46], [50], [54], [75], [98]. By examining the role of theory, we illustrate the interaction between software engineering experiments and relevant theory.

Section 2 introduces relevant concepts. The method by which our analysis was conducted is described in Section 3. Section 4 reports the findings of the review. Section 5 discusses the implications of our findings. Section 6 summarizes and concludes.

2 Background

In more mature sciences, the use of theory tends to be taken for granted, and discussions tend to focus on *how*, rather than *whether*, to use theory [23], [24], [88], [100]. Nevertheless, contrasting views as to the aptness of theories abound. For instance, Lindblom [63] argues that less effort should be expended on the validation of theories and more effort used on extending commonsense reasoning. By contrast, Markovsky [66] discusses virtues that theories have, and that commonsense knowledge lacks. Weick [99] refers to [34], [42], [63] and argues that theory construction is "disciplined imagination" for making sense of the world, rather than for problem solving in the form of theory validation.

Thus, even assuming agreement as to what theories are, the question of when and whether, and if so, to what extent, to use them remains a source of debate in most sciences. Further, there is neither any uniform terminology, nor any universally agreed upon definition of what constitutes a theory. This section presents our approach to coming to grips with this challenge.

2.1 Theory Identity

It is no trivial task to determine what constitutes a theory, that is, to give necessary and sufficient conditions that define what a theory is. The way in which one defines and uses the concept "theory" rests on fundamental philosophical questions, as well as on practical considerations. For example, there are ontological questions pertaining to what kinds of entity and what relationships between them exist, and the sense in which they exist (e.g., in which sense do "cognitive processes" for, say, program comprehension exist [174]). There are also epistemological[1] questions that pertain to the nature and scope of knowledge and beliefs, and the justification for holding beliefs (e.g., what are the elements of an explanation of the anchoring effect in software estimation [53]). Further, we must consider to which uses we wish knowledge to be put (e.g., to develop better software engineering methods for industry). All these issues have a bearing on the nature and components of prospective theories.

2.1.1 Types of Theory

Gregor [38] has summarized the nature of theory from the standpoint of information systems research. Her classifications can be adopted to software engineering, on the assumption that information systems research focuses on user-technology relationships, while empirical software engineering focuses on developer-technology relationships.

Gregor describes five types of theory:

I. Analysis,
II. Explanation,
III. Prediction,
IV. Explanation and prediction, and
V. Design and action.

Type I theories include descriptions and conceptualizations of "what is." Also included are taxonomies, classifications, and ontologies in the sense of Gruber [39]. The lack of explicit explanation, appeal to causal relationships, and prediction disqualifies this class as theory for many scholars [5], [72], [88].

Type II theories explicitly explain why phenomena occur, but lack predictive power. Some scholars discount such structures as theories. For example, Homans [45] claims that Marx' statement that the economic organization of society determines its other institutions does not form part of a theory, because it is not possible to derive any logical consequences from it, even if the particular mode of economic organization of a society is specified.

Type III theories predict without providing explanations. Mathematical and probabilistic models in social science [48], and predictive models in software engineering such as COCOMO [9] fall under this category. Many view physical theories as belonging to this category. For example, Hawking states that "a physical theory is just a mathematical model and ... it is meaningless to ask whether it

1. Epistemological statements pertain to explanation, while ontological statements pertain to existence. There is a difference between saying that the constructs of one's explanations do not exist in the real world and claiming that what one is offering an explanation *about* does not exist. The distinction is foundational for issues summarized in Section 2.1.2.

corresponds to reality. All that one can ask is that its predictions should be in agreement with observation" [40, pp. 3-4], a sentiment also expressed by Feynman [28].

Type IV represents the structures that perhaps most scholars would agree to as being theories. For example, Belbin's model of management teams [114], [115] explains why certain teams are successful and, additionally, has instruments for predicting team success as well.

Finally, Type V theories describe "how to do" things and include design principles. Although there is usually an implicit prediction that following the design principles will be beneficial, this type of theory is reluctantly acknowledged by many [38].

A special note on Type V theories: Their implicit prediction of benefit would seem to be relevant to the research questions in software engineering experiments. For example, in a study of the effects of the use of design patterns, the methodology behind design patterns could be taken as a Type V theory implicitly predicting the outcome of the experiment. In our context, however, Type V theories usually do not constitute structures pretending to explain the cause-effect relationships under investigation. Instead, they merely postulate the existence of the relationship, often not accounting for the human factor, which is an essential part in software engineering experiments as investigated here.

2.1.2 Issues of Explanation and Epistemology

Any classification of theories such as the above is subject to interpretation. For example, the categories rely on an implicit understanding of what an "explanation" is, but this is a nontrivial issue. A common view is that an explanation is an answer to a predefined question of *why* something is—or happens (rather than *what* happens), and Van Fraassen [93] and others have formalized this in various ways. Sandborg [83] elaborates further and states that one must also admit as explanations answers that are corrective in that they introduce concepts that are more pertinent than those of the original question, thus, in effect, answering a different (and better) question. Current views also suggest that explanations should include notions of causality and asymmetry (if A explains B, then B should not also be a viable explanation of A), and that empirical generalizations produced purely from data through inductive and covering-law approaches do not produce explanations [38].

However, the explanatory *function* of a theory depends also on how the theory interacts with other theories and the current level of knowledge. A theory can be seen to explain at one level of abstraction, but not at other levels; the answer to *why* may be the *what* of another level—thus one may also argue that Type I and Type III theories provide explanations [32], [83]. For example, if a mathematical model fits data where no other explanation is available, then such a model may indeed embody explanation relative to prior knowledge.

What one is willing to admit as an explanation is also linked to one's epistemological stance with respect to realism and the existence of a plurality of theories—topics discussed in many disciplines relevant to software engineering.

Very roughly, *logical positivism* and *logical empiricism* hold scientific explanations as meaningless unless their terms in the last instance relate directly to sensory experience in the material world. *Scientific realism* on the other hand, accepts terms that approximate reality, such as for things that cannot be directly observed, e.g., gravity, but maintains that the approximations indeed stand for real things. The existence of postulated unobservable identities (e.g., cognitive processes and software developer motivation) is seen as verified by virtue of the resulting theory's predictive power for phenomena that *are* observable (e.g., behavior and programming performance). Some aspects of *antirealism* and *instrumentalism* hold that scientific explanations and theories need not describe reality at all. Instead, theories are intellectual structures for organizing or modeling our scientific conception of the world. Hawking's statement (Section 2.1.1) is instrumentalist. Some of this spirit is related to that of certain directions in *pragmatism*. Rorty states that "the pragmatist drops the notion of truth as correspondence with reality altogether, and says that modern science does not enable us to cope because it corresponds, it just plain enables us to cope" [79]; see [68].

Furthermore, the *semantic view* sees scientific theories as sets of *models*, along with claims about how things in the world satisfy the model's constructs [80]. This replaces "traditional" hypothetico-deductive approaches to theory [82]. Thus, models are often integral parts of theory, or constitute theory itself—or the two terms are used synonymously [26], [50], [80], [101]. Models are more or less formal structures that allow the building of best approximations "as substitutes for a complete understanding that science may not be able to attain" [80, p. 70]. Models accommodate the instrumentalist view of scientific theories being *useful* devices. Examples are the Bohrean atom model and rational choice theory in economics. Although empirical evidence refutes the existence of key constructs in these models (atoms do not really have electrons in orbits around them, and software project managers do not show consistently rational behavior when making decisions), they remain apt for explanation, prediction, and education in many situations. This independence from "truth" allows for the coexistence of several models describing various aspects of the same phenomenon (*theoretical pluralism*); see also [16], [74].

There are many debates concerning what is the most appropriate epistemological stance for modern-day empirical science [10], [15], [16]. One philosophical stance need not necessarily exclude another. For example, Cacioppo et al. [16] argue for a symbiosis of scientific realism and instrumentalism in the making of psychological theories. Although realism and instrumentalism in some interpretations are incompatible, one may take alternate views in the quest to gain useful knowledge [16], thus counteracting weaknesses and drawing upon the strengths, of both approaches. Also, even though observable phenomena (e.g., those investigated in empirical software engineering) may be bound to realism, they may be explained or

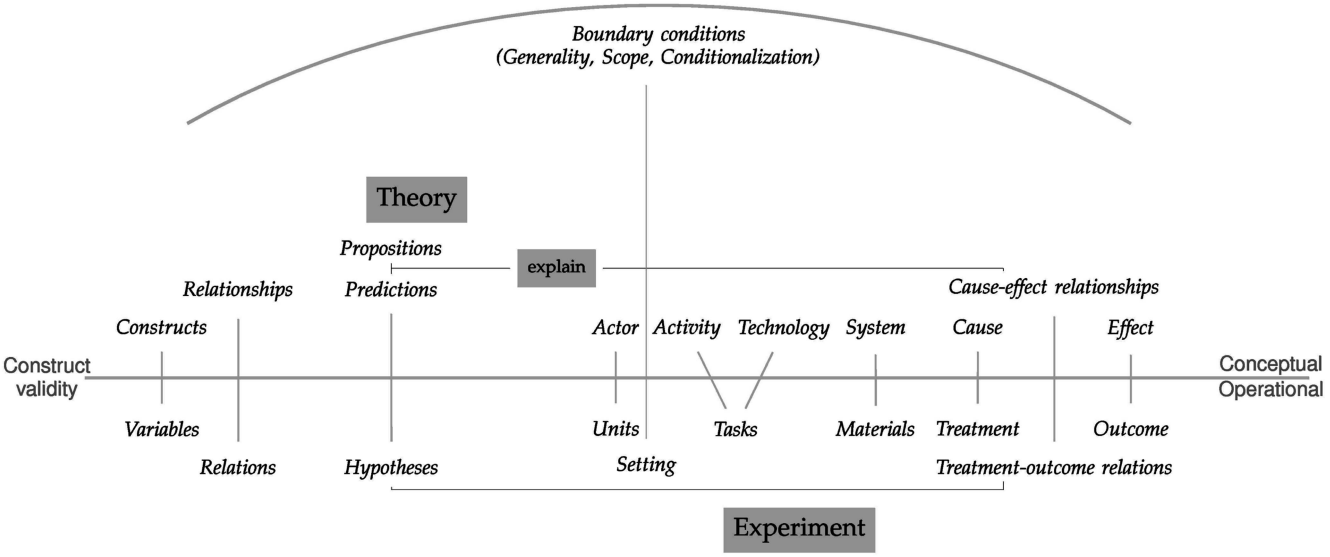

Fig. 1. Components of theories and experiments (adaptation of figures in [5], [90], and [102]). Constructs and relationships are operationalized to variables and relations. Relevant variables for experiments in empirical software engineering are units, settings, tasks, materials, treatments, and outcomes. These variables are operationalizations of relevant constructs. The relationships under investigation are cause-effect relationships, which are translated into hypothesized treatment-outcome relations (covariations) in terms of already-operationalized constructs. Theories play explanatory roles to the cause-effect relationships. These roles are determined by how theoretical propositions (theoretical statements), and predictions derived from propositions, associate to the investigated cause-effect relationships, both on the conceptual level and on the operational level through the translation of predictions into hypotheses. Boundary conditions define the circumstances in which the theoretical statements are meant to apply and should, to a large part, determine the experimental setting.

predicted by theories that speak in terms of entities whose existence is unresolved (entities, some may argue, such as the cognitive structures of cognitive psychology, the superstrings and 26 dimensions of physics, etc.). Causal relationships in such theories may be (instrumentalist) internal workings, and may be independent of the particular cause-effect relationships under investigation in an empirical study. (Causality may be seen as a conceptual rather than existent entity in many situations.) Correspondence with observation is then argued through the process of operationalization (Section 2.1.3 below).

2.1.3 Components of Theories and Experiments

The typology of theory types (I-V) above helps us to come to terms with the diversity of what theories are. The challenge is then to describe theories of such diversity in a uniform schematic way. Gregor provides a schema of structural components for describing theories in information systems research: *Means of representation*, *Constructs*, *Relationships*, and *Scope*. (In addition, there are components for discriminating according to theory type (I-V): *Causal explanations*, *Testable propositions*, and *Prescriptive statements*.) These components are adequate for describing theories as such, but here we also need to relate theories to experiments. In the following, we discuss the three most central structural components of theory (*Constructs*, *Relationships*, and *Boundary conditions*) in association to experiments.

Conceptual level versus operational level. First, in the experiment methodology usually adopted in empirical software engineering, it is useful to divide the domain of discourse into a *conceptual level*, where concepts, theorizing, and theories reside, and an *operational level*, where observations, measurements, and experiments are conducted; see Fig. 1.

Constructs and relationships versus variables and relations. The basic conceptual components are *constructs* and *relationships* between constructs. Bacharach refers to Schwab [84] and Kaplan [55] and states that "a construct may be viewed as a broad mental configuration of a given phenomenon, while a variable may be viewed as an operational configuration derived from a construct" [5]. Shadish et al. state that research cannot be done without using constructs: "Constructs are the central means we have for connecting the operations used in an experiment to pertinent theory, and to the language communities that will use the results to inform practical action" [85, p. 65].

Cronbach et al. [18], [19] and Shadish et al. [85] group the operational variables of an experiment into *units*, i.e., the specific (groups of) subjects of an experiment, *treatments*, *outcomes*, and experimental *settings*. In addition, since software engineering can be said to be a *design science* concerning artifacts [86], we include the operational variables of *tasks* and *materials* (Fig. 1). The conceptual counterparts to these variables are the construct types *actor* (e.g., "experts," "project teams," or "software developers in general") for units; *software process activity* and *software development technology* (e.g., "design using UML" or "validation using functional testing") for tasks; and *software system* (e.g., "safety-critical systems" or "object-oriented artifact") for materials. Experimental settings should be operationalizations of the relevant scope (e.g., "the software industry" or "short time to market strategies"). The conceptual counterparts for treatments and outcomes are *causes* (e.g., "familiarity of design patterns" or "perspective-based reading") and *effects* (e.g., the concepts of "software quality," "developer performance," or "reliability"); both of which

are composed using relevant constructs of the types just mentioned.

Cause-effect relationships versus treatment-outcome relations. What is under investigation in an experiment is one or several *cause-effect relationships* (e.g., "expert familiarity of design patterns when designing safety-critical systems in a short time to market strategy leads to better reliability"), which are operationalized in tests of *treatment-outcome relations* or *covariations* (e.g., the suite of design patterns used for designing this flight controller under induced time pressure by these senior developers who have used patterns regularly is related to fewer exceptions when running the resulting piece of code). Whether the operationalizations actually reflect the constructs and relationships is the question of *construct validity* [85].

Theory and experiment. A theory seeking to explain a cause-effect relationship in an experiment would then relate to the constructs involved in this relationship and offer some additional insight into why or how the cause-effect relationship occurs. For example, a theory explaining the cause-effect relationship exemplified above might use elements from learning theory and cognitive psychology to describe mechanisms for how experts' familiarity of design patterns triggers the recognition of previously encountered abstract problems. How directly a theory relates to an investigated cause-effect relationship depends, among other things, on how conceptually close the theory's constructs and relationships are to those associated with the experiment.

Predictions versus hypotheses. A theory and its predictions are conceptual [5], [90], [102]. In experimental terms, hypotheses are then the operationalizations of a theory's predictions.[2] Hypotheses state in *operational* terms exactly what you think will happen in a particular study [90]. What role a theory plays in an experiment is determined by the way in which the theory's predictions (and the propositions from which they are derived) relate to the cause-effect relationships.

Boundary conditions. Boundary conditions delineate the expanse of a theory's statements. That a theory is *general* means that it is independent of time and place and is applicable in a potentially infinite number of instances [66], [98]. In this sense, a general theory is not meant to encompass all domains. The *scope* of a theory is the domain of phenomena to which it applies [11], [66]. Furthermore, *conditionalization* adds precision to theoretical claims and prevents their undiscerning application to phenomena that may be related but have opposite effects. For example, "anchoring" predicts that the level of initial judgments will influence later judgments [3]. However, in software bidding, empirical research suggests that anchoring is neutralized or reversed in conditions of high perceived risk [53]. Thus, a way of refining the theory of anchoring is to conditionalize it by "perceived risk." A theory with scope conditions and conditionalizations remains general [66].

2. Several accounts place hypotheses on the conceptual level and predictions on the operational level. This is mainly a matter of semantics. For example, Bunge states: "A scientific theory is a system of hypotheses that is supposed to give a partial and approximate account of a bit of reality [14, p. 391].

Although structural schemes such as the above help to systematize a subject that is far from being homogeneous, views as to what a theory's components should be are still diverse, and many authors attempt to set boundaries for what does and does not belong in a theory [5], [17], [25], [45], [71], [97], [101]. Sutton and Staw [88] and Bacharach [5] describe components (data, lists, graphs, predictions, etc.) that do not, on their own, constitute theory. Weick [99], [100] on the other hand, paraphrases [70], [81]: "*Theory* is more a dimension than a category," suggesting that the boundary between theory and more modest acts of theorizing (producing lists, graphs, predictions, etc.) is less clear.

2.2 Theory Roles

For the purpose of this review, we developed the (possibly overlapping) *theory role* categories described below. These roles are targeted specifically toward explaining cause-effect relationships investigated in experiments. Theory used for other purposes is not included in the review but is illustrated at the end of this section.

Design. A theory is said to be used in the *design* of an experiment if the research questions and hypotheses are justified or motivated by the theory. The link from the theory need not be formal, but a clear argumentative link should be apparent. For example, consider a program comprehension theory [118], [174], where the comprehension process is driven by building a hierarchy of so-called mental hypotheses. Building this hierarchy in a top-down fashion, rather than bottom-up, is quicker. In an experiment on design pattern comment lines [78] a particular cause-effect relationship was postulated, namely that comment lines describing design pattern usage will increase programmer performance (when compared with regular comments). The link from the theory is provided by the added assumptions that design pattern comments lead to large-grain mental hypotheses that aid top-down hierarchy construction, and that increased comprehension leads to better performance.

Post hoc explanation. A theory is used as a *post hoc explanation* if is it used as an explanation of observations pertaining to the cause-effect relationship(s) after the experiment has been conducted. An example is a study in which the authors, after having presented the experiment analysis, refer to a model of query writing [156] and state that the results of the experiment (that subjects using the graphical query language QBE perform better than those using SQL) supports the assertions of the model (that a syntactic form requiring fewer transformations from English sentences is easier to learn) [182].

Tested. A theory, or an instance or derivation thereof, is *tested* if clear attempts are made to validate any of the theory's predictions that are directly related to the investigated cause-effect relationship(s). This may happen if predictions of the theory pose directly as research questions or find operational expression in experiment hypotheses. As an example, consider a study that investigated the predictions of media effect theory, media richness theory and social presence theory [123], [146], [162] in a

specific context: "Most theories claim that group performance on negotiation tasks decreases when such leaner media [computer conferencing] are used because of a mismatch between the task needs and the medium's information richness. In our study, we sought to test these assumptions in the context of requirements negotiations" [21]. The theory was therefore tested with respect to a specific setting. A theory that is said to be tested is used directly, as opposed to its use in design, where additional justification is necessary. A theory used for post hoc explanation may also be tested by reusing data from the experiment.

Modified. A theory is *modified* if there is a constructive effort to enhance, refine, conditionalize, etc., an existing theory based on results from the experiment.

Proposed. A theory is *proposed* if the author(s) 1) present their own theory in the article (for example, based on existing literature or analytic deliberations) and the theory pertains to explaining the cause-effect relationship under investigation in one of the roles above or 2) the theory is proposed on the basis of the experiment's treatment-outcome relations. For example, the purpose of an explorative experiment may be to propose an initial theory.

Basis. A theory is referred to as constituting a *basis* if it transitively entails or provides structural elements for another theory in the roles above.

In addition, there might be uses of theory in experiments that do not pertain to explaining the investigated cause-effect relationship(s) and which are, therefore, not included in this review. For instance, theory may be used for definition, that is, in defining operationalizations of the cause construct to treatment variables or the effect construct to outcome variables. For example, when testing the impact of structured versus unstructured function definitions on programmer performance, the cause construct "structured" may be operationalized by employing a control flow model for functional languages [92]. Theory may also be used to define methods under test (without explaining why the method is better), an example being the use of a human-computer interaction model [73], [104] for defining perspectives in perspective-based inspection for evaluating usability [104]. Definitional uses of theory such as these pertain to construct validity.

Theories may also be used in other validity arguments. For example, a behavioral theory of group performance [159] stating that task expertise is the dominant determinant, was used in arguments of external validity (representative subjects) [8].

Other subsidiary roles are circumstantial in that theory is merely mentioned; for example, when a theory of debugging [4], [94] is mentioned in passing in an experiment designed to assess the impact of recursive and iterative constructs on debugging performance [7], but not used elsewhere.

Models can play yet other roles in experiments. For example, statistical models and formal measures can, depending on perspective, take on the role of theory. However, if they are used only for analyzing data, they are not relevant here. More precisely, if a statistical model is formulated to fit existing data without the explicit intention to validate the model on new sets of data, then it is not theory. This is related to the problem of overfitting models to particular data. Gigerenzer refers to Feynman [29] and Hoffrage et al. [44] and states that "the true test of a model is to fix its parameters on one sample and to test it on a new sample" [35].

Thus, what passes as theory depends on one's perspective and the context of usage. From a Type V-theory perspective, principles underlying software engineering technology also constitute part of theory, and for a systems analyst, a UML diagram models a part of reality and can, in a design situation, be said to function as part of a Type II theory. Our perspective in this article, however, is quite clear: The object of explanation is the cause-effect relationships investigated in experiments, and only theories that concern this directly, i.e., found in the roles *design*, *post hoc explanation*, *tested*, *modified*, *proposed*, or *basis*, are considered relevant.

3 Research Method

This section describes how we identified articles reporting experiments, and subsequently, how we extracted theories from those articles.

3.1 Extraction of Experiments

We assessed all the 103 articles describing experiments (of a total of 5,453 articles) identified by Sjøberg et al. [87], published in nine leading software engineering journals and three conferences from the decade 1993-2002 (Table 1). These journals and conferences were chosen because they were considered to be leaders in software engineering in general and empirical software engineering in particular.

Since the term "experiment" is used in an inconsistent manner in the software engineering community (often being used synonymously with "empirical study"), Sjøberg et al. defined *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 treatments—different populations, processes, methods, techniques, languages, or tools. Randomized experiments (random assignment of units to treatments) and quasi-experiments (nonrandom assignment of units to treatments) in the sense of [85] were included alike, because both experiment designs are widely used in empirical software engineering [61]. In this article, we consistently use the term "experiment" in the above-mentioned sense of "controlled experiment."

Excluded are several types of study that share certain characteristics with experiments because, while these may be highly relevant for the field, they do not contain the deliberate intervention or control essential to experiments. Thus, excluded are correlation studies, studies that are solely based on calculations on existing data (e.g., from data mining), and evaluations of simulated teams based on data for individuals. Studies that use projects or companies as treatment groups, in which data is collected at several levels (treatment defined, but no experimental unit defined) are also excluded because these are considered to be multiple case studies [103]. The focus is on articles (not editorials,

TABLE 1
Distribution of Articles Describing Experiments, January 1993-December 2002

Journal/Conference proceeding	*N*	%
Journal of Systems and Software (JSS)	24	23.3
Empirical Software Engineering (EMSE)	22	21.4
IEEE Transactions on Software Engineering (TSE)	17	16.5
International Conference on Software Engineering (ICSE)	12	11.7
IEEE International Symposium on Software Metrics (METRICS)	10	9.7
Information and Software Technology (IST)	8	7.8
IEEE Software	4	3.9
IEEE International Symposium on Empirical Software Engineering (ISESE)	3	2.9
Software Maintenance and Evolution (SME)	2	1.9
ACM Transactions on Software Engineering Methodology (TOSEM)	1	1.0
Software: Practice and Experience (SP&E)	-	-
IEEE Computer	-	-
Total	103	100

prefaces, article summaries, etc.) in which the reporting of experiments is the principal element.

The article selection process was determined from predefined criteria as suggested in [56], see [87] for full details.

3.2 Extraction of Theories

We wish to be as inclusive as possible with regards to the concepts of theory. In our context, viewing theories through the categories of Gregor [38] is a good starting point. In addition, we do not exclude theories on the grounds of their epistemological mode. An insistence on, say, scientific realism or correspondences with "reality" would be too exclusive in the present scientific landscape. We adhere to the maxim that a theory *explains* if it answers a question of *why*, but we accept that such an answer may be indirect or may answer a directly related question (Section 2.1.2).

Theories are in the conceptual domain (Section 2.1.3 and Fig. 1). However, although most researchers probably think and write in terms of concepts (constructs and relationships), not all concepts form a part of theory, and it is nontrivial to draw the dividing line between theory and nontheory. This issue is all the more pertinent here, in that we are sampling theory not from primary explications of theory, but from uses of theory in articles reporting a certain type of empirical study. Therefore, it is necessary to search for explicit statements regarding theory in these articles' discussions.

A review should give an operational definition of the concept it is sampling. However, another consequence of the fact that we are sampling theory from secondary sources is that an operational definition of theory is not useful, because theories are usually not described comprehensively enough in these sources to match such a definition. Instead, the judgment that something is a theory has to be based, not only on explicit renderings of elements of theory identity (Section 2.1), but also by whether it is, in fact, used in roles for explaining cause-effect relationships (Section 2.2). Thus, we give the following two-part inclusion criterion.[3]

3. This approach would not suffice in disciplines such as management and sociology, where the use of theory and models in one form or another is abundant and often implicit. Since our field is less developed in this respect, we postulate that uses of theory stand out more clearly.

Theory inclusion criterion. A theory is identified by

1. *Candidacy for theory:*
   - the mention of the terms "theory" or "model" or grammatical derivatives thereof, together with at least one reference, or, alternatively,
   - the identification of constructs and relationships in a body of conceptual argumentation delineated by diagrams, words, etc., and
2. *Explanation of cause-effect relationship:*
   - the use in the roles of *design*, *post hoc explanation*, *tested*, *modified*, *proposed*, or *basis*.

Thus, the decision to include something as a theory in our setting is based on two equally important factors.

First, the detection of explicit terminology or essential components pertaining to theory gives rise to a *theory candidate*. The reason for the "or" clause is that identifying theories solely on the basis of the authors' explicit use of the terms "theory," "model," or derivatives thereof, is not reliable, because these terms, especially "model," are not used consistently. A theory may be being presented, discussed or used even when it is not referred to by "theory" or "model." Therefore, the identification of constructs and relationships, which are essential structural components of theory (Section 2.1.3), also justify a candidacy for theory, if these occur within a body of conceptual argumentation. For example, in [155], extensive text and diagrams in a designated section constitute a body of conceptual argumentation, in which constructs and relationships are easily identified (e.g., "short-/buffer-/long-term memory" and interactions between these). This yields a candidate for a theory, even though the terms "theory" or "model" are not used for the body of argumentation itself. (The argumentation does, however, refer to other "models.") Conversely, several articles refer to "theories" and "models" ("learning theories," "self-efficiency theory") in their discussions but give no literature references nor any constructs or relationships, rendering the theories too unspecified for consideration as a theory candidate.

The second factor is the determination that a theory candidate is used in explaining cause-effect relationships in

TABLE 2
Data Extraction Attributes for Theories

Metadata
- *Name*. The name given for the theory by the authors of the reviewed article, or by us if no explicit name is given.
- *References*. The literary references given for the theory.
- *Terminology* (*theory, model, none*). Indicates explicit use of the terms "theory," "model" or their derivatives in referring to the theory.
- *Reference discipline*. The discipline(s) of an article's literary sources to the theory.
- *Topic*. The topic of the article in which the theory is used.

Structural components—Generic (adapted from [38])
- *Means of representation* (*words, tables, diagrams, mathematics, logic*). The way in which the theory is presented.
- *Constructs and relationships*. Examples of main constructs and relationships found for the theory.
- *Boundary conditions*. Indications given of the theory's boundary conditions.[a]

Structural components—Contingent on theory type (adapted from [38])
- *Causal explanations*. Indications that the theory gives statements of relationships among phenomena that show causal reasoning (not covering law or probabilistic reasoning alone).
- *Predictions*. Indications that the theory gives statements of relationships between constructs in such a form that their operationalizations can be tested empirically.[b]
- *Prescriptive statements*. Indications that the theory gives statements that specify how people can accomplish something in practice, *e.g.*, construct an artifact or develop a strategy.

Theory role: The values of this attribute are
- *design*: the experiment's research questions and hypotheses are justified or motivated by the theory
- *post hoc explanation*: the theory is used after the experiment to explain observed phenomena
- *tested*: the theory is tested by the experiment – *derivation*: derivation of theory, *instance*: instance of theory
- *modified*: the theory is enhanced, refined, conditionalized, *etc.* based on the experimental findings
- *proposed*: a major part of the theory is proposed by the author(s) in the current reviewed article, and the theory is used in one of the preceding roles or is based on treatment-outcome relations of the experiment
- *basis*: the theory is used as a basis for other theory used in one of the preceding roles.

Calls for Theory: Records any calls for theory or comments on the lack of theory being problematic.

a. The name of this category in Gregor's structural schema is "Scope."
b. This deviates from Gregor's formulation of Testable propositions *[38] in that we insist that the theory be conceptual (Section 2.1.3). A theory may predict, but the predictions are only indirectly testable via their operationilizations.*

an experiment. Such determination concerns two matters: First, neither explicit terminology nor the presence of essential components is sufficient to warrant calling something a theory; the use of the terms "theory" and "model" may be used in ways that deviate from our understanding in this review, and constructs and relationships do not in themselves constitute theory. For example, the mention of theory in "the concept of recursion is essential to tree searching and traversal which in turn are important in areas such as graph theory, artificial intelligence ..." from a study of the effect of recursion versus iteration on comprehension, is clearly not relevant here, nor are mentions of data or design models, and, although constructs and relationships are identifiable for the candidate from [155] exemplified above, these in themselves do not constitute a theory. Second, even if a candidate clearly is a theory, it should not be included unless it pertains to explaining cause-effect relationships in experiments in the reviewed articles. Determining that a candidate takes on one or several of the indicated roles establishes that the candidate is a theory as understood here and that it pertains to explaining cause-effect relationships. For example, the human-computer-interaction model used in [104] is clearly a theory according to our discussion in Section 2.1, but it is used for defining a treatment, not for explaining the cause-effect relationship of the experiment, and is therefore not included; the candidate from [155], exemplified above, remains a candidate until one is able to determine that it is used, in fact, in the role of *design*. Note that role determination is not only a classification of an already-included theory; it is also a means to determine whether something *is* a theory.

Theory role is, at the outset, independent of theory type, and theories of all types (I-V) are admissible. However, as explanation is the focus, it is expected that most of the extracted theories will be of Type II or IV, but some may be of Type I or III, since these may arguably offer explanation (Section 2.1.1). Note, however, that the focus on the investigated cause-effect relationships is crucial. For example, software estimation models may be viewed as Type III theories, but, if an experiment studies the effect of, say, COCOMO versus other estimation models, then the type of theory we are sampling is a theory that explains why one model outperforms another. At this level of inquiry, COCOMO or other estimation models do not offer explanation.

3.3 Extraction of Theory Attribute Data

Table 2 gives the attributes that we use for extracting data about theories from the reviewed articles. The *Metadata* and *Structural components* attributes pertain to the identity of theories. The structural components are adaptations of Gregor's structural components described in Section 2.1.3. Theory use is classified in the *Theory role* attribute using the categories described in Section 2.2. We here describe the data extraction procedure for all attributes that are not self-explanatory.

Metadata. Theory *names* and *references* are compiled from the reviewed articles, when possible. Otherwise, theories are given descriptive names by us. Moreover, we group references according to argumentation in an article: An article may list several sources of, say, program comprehension theory, and each source might give a different theory or theory fragment within the subject. If all these references are used together as a unit in the discussion, we treat them as references to one and the same theory.

Software engineering is recognized as being multidisciplinary, and knowledge from the managerial, psychological, and social sciences is frequently used together with technological knowledge. The *Reference Discipline* attribute records the disciplines of the literary sources to the theory, given in an article. The discipline categories are those of Glass et al. [36]. The *Topic* attribute holds information on the topic of the article in which a theory is used. All reviewed articles have been classified according to topic [87]. Topics are defined according to a classification scheme devised from the IEEE Keyword Taxonomy [49], which is an extended version of the ACM Computing Classification System [1].

Structural components—Generic. *Means of representation* signifies how a theory is presented typographically. The value *words* indicates descriptions beyond references and names of theories. Values for the *Constructs and relationships* and *Boundary conditions* attributes are determined based on the understanding given in Section 2.1.3 regardless of whether or not they are referred to explicitly as such components. Only what we perceive to be main constructs and relationships are recorded. Boundary conditions do not include obvious boundaries, such as the implicit fact that a software control model has software development as scope.

Structural components—Contingent on theory-type. Gregor [38] used these attributes to determine theory type (I-V). We record the presence of these components based on our understanding in Section 2.1.1.

Theory role. We categorize theories to roles according to what we perceive to be the main intent in an article's argumentation, and a theory may be seen to belong to several role categories. A *proposed* theory is taken to be such if major parts of it are presented as elaborated by the author(s) and there are no references to these parts of the theory in the article. A theory is categorized as a *basis* theory if it has this role relative to a theory already assigned to a role.

As mentioned, the determination of theory role is based on its argumentative position in an article. For this, but also for the identity of theories, the cause-effect relationships under investigation must be identified, as well as possible predictions, research questions, and (operational) hypotheses. Evaluations of the (logical) links from theory to research questions and hypotheses help to determine theory use. However, the details are not reported here.

As suggested by [56], we wished to determine the extraction process from predefined criteria. The sampling from nonprimary sources and the lack of clear definitions of theory precluded this somewhat. We therefore performed an initial exploratory analysis of the 103 articles and, after some adjustments, compiled the attributes listed in Table 2. The final analysis with respect to theory identification and extraction were done by three reviewers in such a way that all articles were analyzed by at least two authors, with the first author analyzing all articles. Conflicts were resolved by discussion or, in the last instance, by majority vote, and reasons for inclusion/exclusion were documented. For the remaining attributes, one reviewer extracted data and two other reviewers subsequently checked this data, with the same conflict resolution strategy as above.

4 Findings

This section presents our findings. First, we report on who uses which theories for what purposes. Then, we report theory-specific data in order to characterize the kinds of theories involved.

4.1 Articles and Theories

The 103 articles in this review describe 113 experiments. An article may describe several experiments, but at our level of analysis, the use of theory is uniform per article. Using the theory inclusion criterion, we extracted a total of 40 theories in 24 articles. Fig. 2 lists the articles, sorted by topic, in relation to the theories they use and the roles in which they are used. The theory identifiers (T1-T40) are used in the text and relate Fig. 2 to Table 4, which gives references to the theories' literary sources cited in the article(s). (All extracted theory references are included in the latter part of the References section ([105] through [182]).[4] The reviewed articles that propose theories appear among the theory references. Otherwise, they appear among the regular part of the References section ([1] through [104].)

Only two of the 40 theories are used in more than one article: "Probabilistic model of PBR and CBR inspection" (T1) is used in three articles, [33], [59], [140] authored partly by the same people, and "Theory of cognitive fit" (T7) is used in two articles, [2] and [47].

The two topics with the largest amount of articles using theory are *Code inspections and walkthroughs* with five articles using six distinct theories and *Object-oriented design methods* with three articles using five distinct theories. Roughly, the number of articles using theory within each topic varies proportionally with the total number of articles in each topic.

As can also be seen from Fig. 2, the article with the most (four) theories used [105] is classified to topic *Productivity*. Three other articles use three theories each, [144] with topic *Design notations and documentation* and [119] and [21] with topic *Software psychology*. In all, 14 articles use more than one theory.

There are, in total, 83 citations to 78 literary sources associated to the 40 theories. The sources are cited in one article each, except Laitenberger et al. [140], which is cited in three articles, and Brooks [117], Pennington [153], and Miller [148], which are cited in two articles. The number of

4. These references are extracted from the reviewed articles and are not necessarily complete or up-to-date.

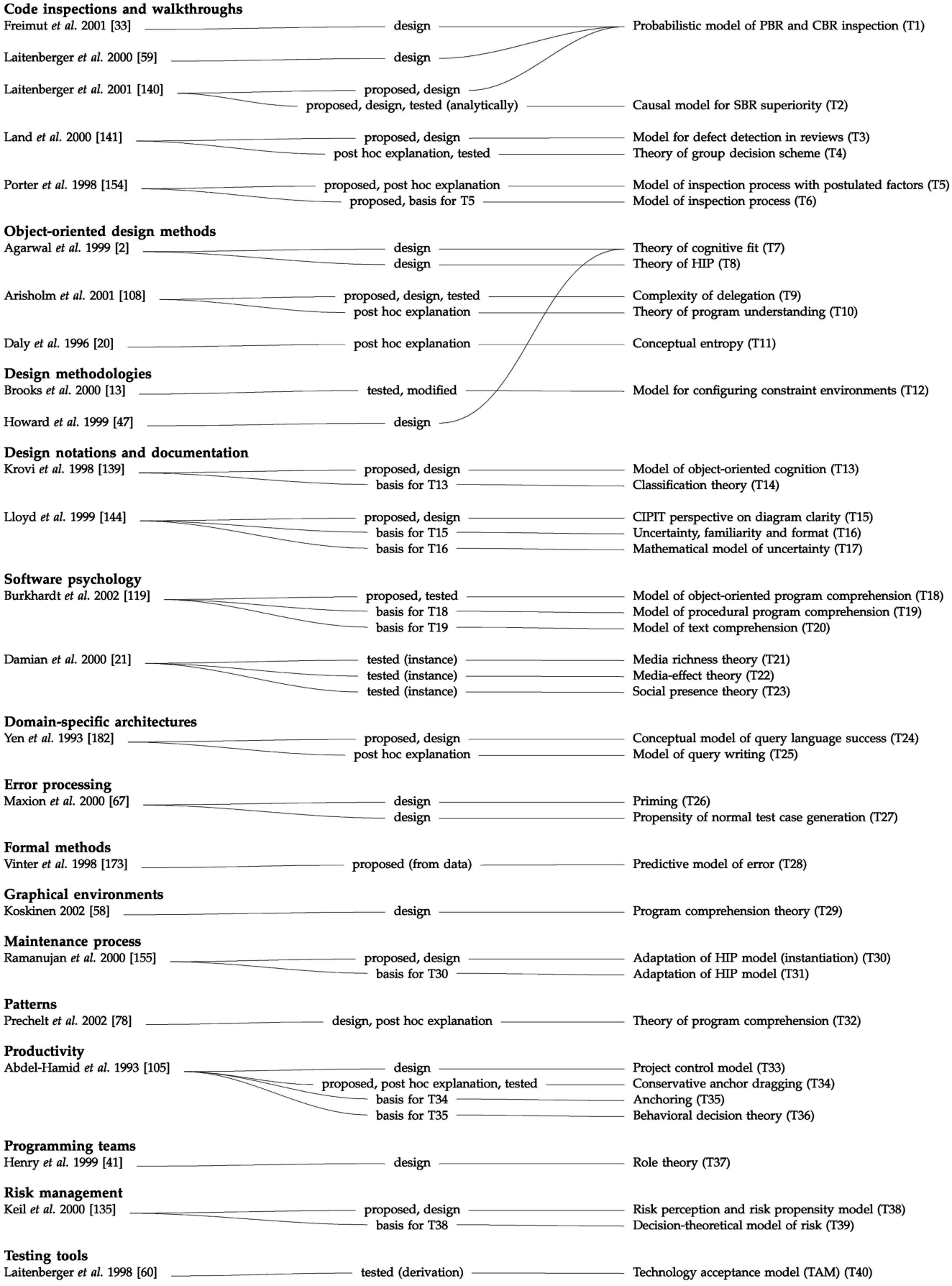

Fig. 2. Articles using theory.

distinct theory references per article ranges from 1, to 14 in [155]. Note that a reference may occur in several of the theories.

Concerning terminology, 15 theories are referred to explicitly as "theory," 15 are referred to explicitly as "model," one is referred to as "theory" or "model" in different articles, and nine are not referred to as either. These nine are referred to as "perspective," "framework," "idea," "concept," "cause-effect diagram," "factors," "conceptualization," "process," "strategy," "phenomenon," and "rationale." According to our inclusion criterion, none of these words suffice to qualify something as a theory. However, constructs and relationships (as well as roles) were identified for all these nine theories.

4.2 The Theories' Roles

In accordance with our inclusion criterion, the theories extracted are used for explaining cause-effect relationships in the roles of *design* (17 articles), *post hoc explanation* (7 articles), *tested* (8 articles), *modified* (1 article), *proposed* (12 articles), and *basis* (7 articles).

Theory as motivation rather than incentive. The most common use of theory is in the *design* role. When theory influences the design of an experiment, one might expect that the theory would be tested as a matter of course. However, in the reviewed articles we found that the *design* role does not usually imply that the theory is tested. Instead, these theories and models are used for forming conceptual frameworks and motivation for research questions and hypotheses. A pertinent example is [2], where the theory of cognitive fit (T7) is used in rationalizing the experiment's hypotheses. However, since the theory of cognitive fit is more general and the article does not formulate a subtheory to describe more specifically the issues investigated, we found that the intention was to motivate the design rather than to test.

That a theory is used for design only, rather than for theory testing, may be due to the theory having low specificity. It might predict the presence of a relationship without specifying what exactly that relationship is (often indicated diagrammatically by a line connecting two boxes, but leaving the line unspecified). For example, in one article [182], the authors set up a conceptual model of query language success (T24) postulating that various constructs may affect the construct of "query language success." Constructs and relationships are present but do not in themselves entail the more specific hypotheses of the experiment. Some scholars would hesitate to call anything with such low specificity an explanation. We hold that this depends on the present level of knowledge. In our review, we found that models with low specificity often gave valuable conceptualizations and explanations at more exploratory stages.

Ten theories (in eight articles) do get tested, however, and some of these are contradicted. The three theories on media richness (T21), media effect (T22) and social presence (T23) mentioned in [21] are not supported by the empirical evidence of the experiment. Theory T2 is analytically refuted on the grounds of calculations and simulations in a probabilistic model framework in which both Theories T1 and T2 are modeled [140]. Also, one test of theory results in modification rather than refutation; see below.

Theory generation. With regards to constructive efforts, 12 articles propose theories, three of them (T5, T28, and T34) on the basis of observations from the experiment. Theory T28 is a mathematical model based on experimental data. In [105], the proposed theory (T34) is the result of deliberations about the nonconformance of observations with existing theory (T35). The proposed theory is then tested on data from the experiment. The article also uses theory in the *design* role. Also a constructive effort, one article modifies an existing theory: In [13], an exploratory experiment with no stated null hypotheses is described. The observed phenomena, however, allowed the experimenters to test a model (T12) for configuring constraint environments for a tools-user response project. The results are in accordance with the model, but the authors conclude that the model needs more constructs (in order to account for HCI-imposed constraints and subject speed and ability). Theory role is here classified as *tested* and *modified*.

Theory structuring. Several articles present theories that are based on other theories in a hierarchic manner. An example is [119] where the foundation is a comprehension-integration model of text comprehension (T20). This forms the basis of a mental model of procedural program comprehension (T19), which in the current study is extended to a mental model for comprehension of object-oriented programs (T18). This model directs the hypotheses, and the observations are discussed relative to the proposed model. Another example [114] is the hierarchical buildup of the cognitive information processing/information theory perspective (CIPIT) (T15) from the basis of a mathematical model of uncertainty (T17) and a psychological model of uncertainty (T16). One theory is the result of conditioning other theory. "Conservative anchor dragging" (T34) is proposed as an explanation of experimental observations that do not support Hogarth's predictions on anchoring in dynamic environments (T35).

Theory in several roles. Thirteen articles use several theories in different roles (apart from *proposed*). For example, in [141], theory plays the roles of *design*, *post hoc explanation*, and *tested*. The study presents a model (T3) for predicting defect detection performance in groups. The experimental hypotheses are directly structured and formulated according to this model. A group decision scheme (plurality, or majority-rules) (T4) is used afterward to explain observations, and this yields a deeper level of explanation. The group decision scheme is then tested by reusing data from the experiment.

Theory building. Theory building in large demands collaborative efforts over long periods of time and would, in practice, involve all of the above roles. The reviewed articles give no indication that this is taking place to any great extent. The intention is, nevertheless, present in one article: "This study is a small step in the direction toward building a cognitive psychology based theory of software maintenance effort that not only predicts 'what' factors affect software maintenance effort but also explains 'why'

TABLE 3
Theory Characteristics—Structural Components

Component	N	%
Generic		
Means of Representation (other than name or references)	39	97.5
Constructs and Relationships	37	92.5
Boundary conditions	5	12.5
Contingent on Theory Type		
Causal explanations	32	80.0
Predictions	34	85.0
Prescriptive statements	0	0

and 'how' these factors influence software maintenance effort" [155]. Another article gives at least an indication of future work involving theory: "Future research should study the behavior of query construction and processing by making use of concepts from the theories of information processing and problem solving" [182].

4.3 The Theories' Structural Components

The amount of identifiable structural components gives indications of the explicitness with which theories are presented. Table 3 shows the number of theories exhibiting each component. Note, however, that no articles report constructs and relationships or boundary conditions explicitly by name. Examples of constructs and relationships are given in Table 4. (For three theories, constructs or relationships were not found. These theories were all denoted by the term "theory" in the articles.)

Means of representation vary from extensive presentations with words, references, logic, mathematics, diagrams, and tables, e.g., Theories T1 and T2 as described in [140], to mere references and name, e.g., Theory T1 as mentioned in [59] and Theory T14 as mentioned in [139].

Five theories are described with what amount to boundary conditions. Theories T3, T18, T20, and T38 have scope conditions. An example is: "Our findings relate to reviews which conform to the model ... that is where there is individual defect detection followed by a group meeting. Since the focus of such meetings is on collection of defects, it is impossible to predict whether procedural roles would have a similar effect on Fagan-style inspections where the focus of the meeting is on defect detection" [141]. Theory T34 has conditionalization: "We will label this decision strategy 'conservative anchor dragging,' i.e., continuous anchoring moderated by conservatism" [105]. Here, the moderation of continuous anchoring under uncertainty yields conservative anchor dragging. In addition, one article makes the point that conditionalization and scoping are important: "Other studies, however, have found risk propensity to be a situationally specific variable, meaning that an individual's risk propensity will not be the same in every situation ... This suggests, for example, that if one is interested in predicting decision-making in an IS project context, then it is necessary to examine risk propensity in situations concerning IS project decision-making" [135].

Since our samples are not from primary sources of theory, we did not expect to find sufficient information to determine theory type (I-V). Nonetheless, we did find structural components contingent on theory type for 36 of the 40 theories. The findings suggest that 34 of these 36 theories are Type IV (Explanation and Prediction), one theory, T24, is Type II (Explanation), and one theory, T28, is Type III (Prediction). We found no theories that we could positively classify as Type I (Analysis) because, unlike in a review from primary sources, the absence of explanation and prediction (and prescription) does not imply that a theory is Type I. Also, we found no theories (matching our inclusion criterion) that we could classify positively as Type V. This distribution of theories to type is similar to Gregor's findings for articles describing theories in the information systems field [38]: She found that 33 of 50 articles presented Type IV theories, four presented Type II theories, and one presented a Type III theory. In addition, Gregor found three articles presenting Type I (Analysis) and nine presenting Type V (Design and action).

4.4 The Theories' Disciplines

Table 4 shows the disciplines of the referenced literature for the identified theories. (In order to comply with Glass et al.'s classification (Section 3.3) four cases of references belonging to HCI literature have been included in the software engineering category.) Fifteen of the 40 theories have references belonging to software engineering literature alone. Seven theories have references belonging to both software engineering and various combinations of cognitive psychology, social and behavioral science, and information systems. Eighteen theories have no references belonging to the software engineering literature in the respective articles.

Theories referenced in the software engineering literature (including theories proposed in the reviewed articles) may be adaptations of theories from other fields. Examples of this from our review are Theories T30 and T38; the former is an instantiation, within a software engineering context, of an adaptation of a human information processing model from cognitive psychology and social and behavioral science, and the latter is a risk perception and risk propensity model for decision-making in software engineering, based on a decision-theoretical model of risk from information systems, cognitive psychology, management, social and behavioral science, and economics.

Thus, in total, 22 (55 percent) of the extracted theories were classified as coming from the discipline of software engineering, on the basis of the theories' references. A similar proportion of the theories are imported from other disciplines. Glass et al. [36], [37] concluded that on the whole, software engineering research relies very little on other disciplines for its thinking, a trait it shares with research in computer science, whereas information systems research relies on other disciplines to a much greater extent. However, the theories in this review show a much higher degree of interdisciplinarity than does software engineering research in general. Table 5 shows the results of Glass et al. [36] for research in general in computer science (CS), software engineering (SE), and information systems (IS), together with our findings for theories in empirical software engineering.

TABLE 4
Theory Characteristics

	Theory	Constructs and Relationships	Reference Discipline
T1.	Probabilistic model of PBR and CBR inspection [140]	subset focus, defect overlap, inspection effectiveness, probabilistic associations	SE
T2.	Causal model for SBR superiority [140]	subset focus, defect overlap, inspection effectiveness, probabilistic associations	SE
T3.	Model for defect detection in reviews [141], [145], [167]	defect groups (process loss, emergent defects), *etc.* , expertise, process, *etc.*	SE/Social and Behavioral Science
T4.	Theory of group decision scheme [159]	group decision scheme, plurality dominance, group performance, process skills	SE
T5.	Model of inspection process with postulated factors [154]	process inputs, collection output, inspection output, code unit factors, author factors, reviewer factors, cause-effect relationships	SE
T6.	Model of inspection process [154]	process inputs, collection output, inspection output, cause-effect relationships	SE
T7.	Theory of cognitive fit [106], [171], [172]	problem representation, problem solving task	IS/Management
T8.	Theory of human information processing (HIP) [149]	cognitive processes, problem space	Cognitive Psychology
T9.	Complexity of delegation [108]	good design, bad design, delegation, complexity, effort, correctness, stability	SE
T10.	Theory of program understanding [176]	control-flow abstraction	SE
T11.	Conceptual entropy [127]	entropy, conceptual inconsistency, class hierarchy	SE
T12.	Model for configuring constraint environments [125]	task characteristics, constraint characteristics, perceptual filter, attitude toward constraint, quality, productivity, ...	IS
T13.	Model of object-oriented cognition [139], [157], [164]	inheritance, polymorphism, message passing, cognitive economy	SE/Cognitive Psychology/Social and Behavioral Science
T14.	Classification theory [165]	-	Cognitive Psychology
T15.	Cognitive information processing/information theory perspective on diagram clarity [144]	uncertainty, format, familiarity	SE
T16.	Uncertainty, familiarity and format [120], [132], [134], [148], [151], [177], [181]	chunks (of info.), familiarity, choice reaction time, information, noise, uncertainty	SE/Cognitive Psychology/Social and Behavioral Science
T17.	Mathematical model of uncertainty [160]	information, noise, uncertainty	CS
T18.	Model of object-oriented program comprehension [119]	text relations, knowledge structures, mental representation, model	SE
T19.	Model of procedural program comprehension [152], [153]	text relations, knowledge structures, mental representation, model	SE/Cognitive Psychology
T20.	Model of text comprehension [170]	surface form representation, propositional textbase representation, situation model	Social and Behavioral Science
T21.	Media richness theory [123]	information-carrying capacity, feed back, channel, source, language	Management Science
T22.	Media-effect theory [146]	information-carrying capacity, feed back, channel, source, language, group interaction	Social and Behavioral Science
T23.	Social presence theory [162]	media, individuals, types of interaction	Social and Behavioral Science
T24.	Conceptual model of query language success [182]	situational variables, experiential variables, DBMS characteristics, QL success	SE
T25.	Model of query writing [156]	template, lexical items	CS
T26.	Priming [180]	categories, initial recall, semantic extension	Social and Behavioral Science
T27.	Propensity of normal test case generation [136]	normal test cases, fault test cases, propensity	Social and Behavioral Science
T28.	Predictive model of error [173]	logical form, inference task, expertise, correctness, probability of correctness	SE
T29.	Program comprehension theory [117], [143], [175]	-	SE
T30.	Adaptation of human information processing (HIP) model (instantiation) [126], [155], [179]	short-term/long-term/buffer memory, chunking, (non-)episodic, internal semantics, given state, desired state, funneling	SE/Cognitive Psychology/Social and Behavioral Science
T31.	Adaptation of human information processing (HIP) model [110], [111], [121], [122], [129], [130], [148], [150], [161], [168], [178]	short-term/long-term/buffer memory, chunking, (non-)episodic, internal semantics	SE/Cognitive Psychology/Social and Behavioral Science
T32.	Theory of program comprehension [117], [118], [142], [153], [166], [174]	beacons, knowledge domains, delocalized plans, hierarchy of hypotheses	SE/IS/Cognitive Psychology
T33.	Project control model [107], [112]	planning, control, measurement, comparison, deviation, information, communication and feedback between constructs	Management
T34.	Conservative anchor dragging [105]	initial estimates, final estimates, bias, dynamic environments, conservatism	SE
T35.	Anchoring [133], [169]	initial estimates, final estimates, bias, static decision tasks, dynamic environments	Social and Behavioral Science/Economics
T36.	Behavioral decision theory [128]	-	Cognitive Psychology
T37.	Role theory [114]–[116], [137], [158]	unsure position, behaviors, personality characteristics, functions, functional requisite, role complement	IS/Management/Social and Behavioral Science
T38.	Risk perception and risk propensity model [135]	risk perception, risk propensity, probability of failure, magnitude of loss, decision	SE
T39.	Decision-theoretical model of risk [109], [113], [131], [138], [147], [163]	risk perception, risk propensity, probability of failure, magnitude of loss, decision	IS/Cognitive Psychology/Management/Social and Behavioral Science/Economics
T40.	Technology acceptance model (TAM) [124]	ease of use, usefulness, tool-acceptance behavior, self-predicted future usage	Management Science

TABLE 5
Reference Discipline (Extension of Table 3 in [36])

Reference Discipline	CS	IS	SE	Theory in ESE Experiments
Cognitive Psychology	0.80%	10.7%	0.54%	25.0%
Social and Behavioral Science	-	9.0%	0.27%	32.5%
Science	0.96%	-	0.27%	-
Management	-	18.0%	0.27%	10.0%
Management Science	-	6.6%	0.27%	5.0%
Economics	-	11.1%	-	5.0%
Mathematics	8.60%	-	-	-
CS	**89.33%**	-	-	5.0%
IS	-	**27.2%**	-	12.5%
SE	-	-	**98.1%**	**55.0%**
Other	0.32%	12.5%	0.27%	-
Not applicable	-	4.9%	-	-
Review Base	628 articles	488 articles	369 articles	40 theories

The CS, IS, and SE columns show the percentage of literature from various disciplines cited in articles from, respectively, computer science, information systems, and software engineering and gives an indication of the interdisciplinarity in these three fields. The last column shows the distribution of theory references to disciplines for the 40 theories extracted in this review. (A theory may have several references, so the percentages add up to more than 100.) The interdisciplinarity for theories is higher for these theories than for software engineering in general.

Relating to bodies of knowledge in other disciplines may increase explanatory power. Some of the reviewed articles also make this point: "Therefore we also recognize that theories of software maintenance effort can be based on other theoretical perspectives from fields such as sociology and social psychology" [155]. Another states: "The major reference theories for examining the efficacy of alternative systems analysis and design methods come from the cognitive psychology and human factors literatures" [2].

Conversely, there seems to be potential for empirical software engineering to contribute theoretically to other disciplines. For example, in [105], the static and dynamic anchoring concepts from management, economics, and psychology are refined to conservative anchor dragging. Although the theory is derived from an experiment in a software engineering context, it is not formulated specifically for this context, and the theory should be relevant in a context wider than software engineering.

4.5 Calls for Theory

Explicit calls for theory indicates the awareness of theory among the authors of the reviewed articles. Nine of the 103 articles either comment on the fact that there is a lack of any relevant theory or express a desire for relevant theory. In five of these, the authors comment that such a lack hinders an explanation of the phenomena observed. For example, "As yet, there exists no coherent theory that would explain these advantages specifically in the context of design patterns" [78]. An example from an article that does not use theory: "This work suggests that there are general and identifiable mechanisms, driving the costs and benefits of inspections. However, we lack a comprehensive theory bringing these principles together. We are currently exploring this issue" [76]. Yet another example: "There is currently no psychological theory that allows these differences to be predicted based on known attributes of subjects. This is an interesting area for long-term basic research" [31]. In one of the articles, the comment is made that theory is necessary for generalization: "Furthermore, without an explicit theory of SW maintenance, it is difficult to predict what effect other design patterns (and alternatives) than the five specific ones used in the experiment may have" [77].

The importance of a solid body of empirical work prior to building theory is emphasized in two articles: "In mature scientific disciplines, this is a standard procedure before any theory will be considered valid. The present work provides such empirical evidence" [77], and "Multiple independent studies of the same hypothesis are essential if software engineering is going to produce empirically evaluated theories and procedures" [69].

5 Discussion

This section discusses the implications of our findings for empirical software engineering, and relates our review to the discussion about theory in general.

5.1 Extent of Theory Use

Our review suggests that most experiments do not relate to theory. Instead, they are "searches for empirical regularities" in the terminology of [22]. About 23 percent of the reviewed articles use theory in explanatory roles pertaining to the investigated cause-effect relationships. It is not a straightforward matter to assess quantitatively whether or not this is a lot. To our knowledge, no comparable reviews exist for other fields, but it seems unlikely that such reviews exist in any large number.[5] Thus, we have no quantifiable data on theory use with which to compare our field with others.

However, it is still possible to make qualitative judgments on the extent of theory use. First, the fact that only two of the extracted theories are used in more than one article (and only one of these is used in articles by different authors) indicates that there is little sharing of theories, even within topics. Theories provide common conceptual frameworks to which researchers may relate, and this is a

5. The status of theory use in other disciplines might be difficult to quantify because theory use may be more integral and implicit than that which we experienced in doing our review.

prerequisite for building larger cumulative bodies of knowledge. Our findings show that this use of theory is negligible.

Second, although we found that theories are used in important explanatory roles, we cannot say overall that theories are used for providing a theoretical framework or paradigm within which studies are conducted and interpreted. Rather, most of the theories we extracted are used somewhat locally for supporting and motivating the study, rather than the study being a result of theory. Only two articles hint that the experiments are steps in larger efforts at building theory. Even the theories proposed within the topics of *Code inspections and walkthroughs* and *Object-oriented design methods* were not tested or further developed in other experiments.

From these two points of view, our findings constitute empirical support for prior claims (e.g., [43]) that theory-driven investigations and theory building are rare in empirical software engineering.

On a more modest level of expectations, however, this review suggests that empirical software engineering is in no way devoid of theory. Although large-scale theory building may not be present, constructive efforts are being made, in that theory is proposed in almost 12 percent of the 103 reviewed articles.

5.2 The Usefulness of Theory

Does our review suggest that empirical software engineering should engage in more theoretical deliberations than is now the case? Our review reports the *use* of theories and does not assess the *usefulness* of theory. Indeed, given that an objective attribute-based assessment of usefulness is even possible, it is probably too early to make any such claims. However, after working extensively with the 103 reviewed articles, we do, nonetheless, have opinions on issues related to the usefulness of theory.

Our opinion is that, had theory been omitted from the articles that do use theory, they would have been less *interesting* because the theories used provide a conceptual framework for explaining observed phenomena. Such explanations often provide links to other disciplines, in which a closely related problem may have been researched extensively. For example, in [155], the human information processing theory (T30) that is used provides a conceptual framework in which a number of phenomena may be explained and predicted. Phenomena that may seem intuitively obvious, such as that a decrease in program control flow complexity leads to higher maintainability and that this difference is greater for large programs than for small programs, are explained in a framework that links the underlying mechanisms for this phenomenon to the mechanisms of other (perhaps less obvious) phenomena.

It is also our opinion that, had theory not been used, some of the *less obvious* research questions investigated would have been harder to rationalize or even come by. (Davis [23] discusses a suite of ways in which "good" theories give rise to nonobvious research questions.) For example, in [67], the theorized propensity in people for generating normal test cases rather than fault test cases (T27) is used to predict the relative performance of various exception coverage strategies according to how well the strategies address this propensity. Without this, or a similar theory, it would seem hard to argue that one strategy is better than the other. In [155], the human information processing theory (T30) is used again to argue that increased concentration leads to more efficient maintenance, and more so for novices (who have low semantic knowledge) than for experts (who have high semantic knowledge). Without this theory, it would seem difficult to argue that the impact of concentration should be greater for novices, rather than the impact being greater for experts.

Finally, in our opinion, several of the theories provide paths for *new research* in that they often speak in terms of underlying mechanisms that have not yet been investigated. An example is the framework in which the Theories T1 and T2 are formulated [140]. The framework simulates two posited mechanisms—subset focus and defect overlap—for perspective/scenario-based reading. Calculations and Monte Carlo simulations in the framework demonstrate that the intuitively posited relationships between subset focus, defect overlap, and the resulting defect detection rate do not always hold. Thus, theory may generate deeper research questions, and corresponding empirical studies that in turn would provide insights for devising even better inspection methods.

Based on the fact alone that nine of the surveyed articles themselves state explicitly that the lack of theory is an obstacle, we think that efforts should be made to develop and use theory to a greater extent.

5.3 Obstacles for Using Theory

It is, perhaps, a common conception that a massive body of empirical evidence must be accumulated prior to building theory. However, this is just one of many ways to generate theory. Theory may be generated on only modest empirical evidence, it may be derived mathematically (and even on principles of aesthetics), and it may be derived as adaptations of theories from other disciplines. Indeed, this review shows several examples of theory being generated in these ways. Additionally, some areas of software engineering have already accumulated considerable empirical evidence, and many scholars state quite clearly that "the experimenter must begin with theory" [98] and not with observation.

Another obstacle to using theory may arise from difficulties in relating theories to empirical research. This review should be helpful in this respect, since it summarizes how theory may be used in experiments. There are also a number of general suggestions as to how theoretical and empirical research should interact. Based on the nature of empirical software engineering, Pfleeger suggests an iterative cycle of "study a little, theorize a little" [75]. Theory and experimentation can interact in a variety of ways; for example, Davis and Holt [22] describe various types of experiment that relate to theory in different ways, Waller and Zimbelman [96] present a bridging strategy for generalizing theoretical propositions to field settings via the laboratory, and Lynham [65] suggests a general method for theory-building research in applied disciplines that

consists of five iterative phases involving both conceptual development and application.

Theory is abstract and therefore, one might argue, it is of no use to the practice of software industry.[6] However, other practical disciplines, such as pedagogy, psychology, nursing, and management are theory-based, and standard textbooks in these subjects present relevant theory as a matter of course. Theory purports to explain underlying mechanisms of whatever its subject matter is. Whether this subject matter is austere multidimensional string theory or practical software engineering, is not in itself the point. For example, understanding the underlying mechanisms of software effort estimation in terms of variants of anchoring (theories T34 and T35) [105] has a direct impact on how project managers may improve their estimates. Understanding underlying mechanisms means understanding more of "what really goes on"—in the famous words of Kurt Lewin: "There is nothing so practical as a good theory" [62]. Further, consequences derived from a sound theory can motivate the revision of practice in the interests of better performance.

5.4 Explicit Use of Theory

None of the reviewed articles state reasons for using theory (or reasons for abstaining from using theory). Although it may not be realistic to expect such reporting, we think that researchers should decide explicitly whether they wish to offer conceptual explanations to their observations or not. For example, Houdek takes an explicit stance in his framework for conducting laboratory experiments in the direct interests of industry: "Generalization of results to other environments [than] the original target environment ... is not covered by [this] approach at all" [46]. Herein lies an explicit decision that renders theoretical deliberations less meaningful.

Explicit theoretical terminology concerning the types of theory, structural components, roles, and epistemological issues are all but absent from the reviewed articles. (One article [154] does explicitly discuss criteria for causality and explanatory versus predictive traits of model fitting.) However, as the use of any research method requires the reporting of how the method's principles are applied, so also should the use of theory as an explanatory device be made explicit—a concern also voiced in other relevant disciplines [88], [97]. For example, it is important to be explicit about boundary conditions, because these free the researcher from false obligations to formulate theories that are "general" in ways that are not useful, and because boundary conditions make explicit how theories are adapted to the context of the study.

Furthermore, in established disciplines, epistemological issues are often resolved implicitly according to in which school of thought one is operating. The situation is more delicate in empirical software engineering because there are no such established theoretical directions, and when theories from other disciplines are used for software engineering, the epistemological context of these theories may easily be lost. For example, in [139], a scientific-realist stance would seem evident in that one postulates the actual presence of object-oriented properties in human cognition, based on the agreement of observation with predictions of an object-oriented cognitive model (T13). On the other hand, Bourne et al. state that the view of the human brain as an information processing unit, is "an evolving framework or domain of discourse that permits cognitive psychologists ... to exchange cogent ideas regarding cognitive phenomenon" [12, pp. 11-12], which suggests an instrumentalist perspective. Logical positivism as expressed in some modes of psychological behaviorism does not admit explanation in terms of the constructs of cognitive psychology at all; unless these constructs stand for observable movement in the brain (localized by disciplines such as cognitive psychophysiology, e.g., [51]). This pertains to many of the constructs found in this survey, such as "risk perception" in cost estimation, "mental stress" during coding, "attitudes" toward tools, "hierarchy of hypotheses," "knowledge domains," and "beacons." Also, in [105], notions of anchoring proposed by various authors are used. Some of these authors may hold that anchoring is a consequence of cognitive structures present in the brain, others may hold that "anchoring" is merely a useful conceptual device, while yet others may use "anchoring" in a behaviorist way as the name of an observed effect. It is not obvious that ignoring epistemological mode is possible when relating to other disciplines, when combining explanations, or when asking others to interpret one's results.

Finally, it is important to be explicit about how one proposes to use theory: for explaining the cause-effect relationship(s) (e.g., *design*, *post hoc explanation*, *tested*, *modified*, *proposed*, or *basis*), or for subsidiary or other uses. In the reviewed articles (including some of our own), there were several references to theory whose uses it was difficult to determine. We think it will benefit others who wish to build on a study if uses of theory are described clearly.

5.5 Software Engineering Theory

What is a software engineering-specific theory? From our results and from our experiences with this review, the answer to this question is not straightforward. In only a few instances could we claim by looking at the theory itself that it was specific to software engineering. By and large, the theories that we classified to software engineering incorporate constructs that are endemic to other disciplines, and, sometimes, an article's argumentation was done entirely in terms of constructs that are not software engineering specific. (Indeed, in determining the discipline of theories, we relied on the disciplines of publications referenced in the reviewed articles, rather than on characteristics of the theories themselves.) The multidisciplinary nature of software engineering and the uses of interdisciplinary theories uncovered in this review suggest that it may be difficult to devise theories and larger theoretical frameworks that are entirely endemic to software engineering, even when they refer to the peculiar traits of software development problem solving.

6. Lynham [65] discusses further such claims in a general discussion concerning the relationship between theory and applied disciplines.

Other theory-based disciplines have theoretical frameworks that encompass and define the entire discipline, and one may criticize software engineering for lacking this. However, since software engineering is as many-faceted as it is, it might be likely that smaller units of theory will evolve, each seeking to explain different aspects of a phenomenon according to existing theory from other disciplines. The semantic approach to theories accommodates this, as does theoretical pluralism (Section 2.1.2). The incentive to build theory might be a refinement of what characterizes the uses of theory uncovered in this review: rather than large concerted efforts, local efforts to model phenomena in different ways that are useful for answering the various practical questions at hand.

5.6 Limitations

The main limitations of this study are publication selection bias, inaccuracy in data extraction, and misclassification. Publication selection bias is addressed in [87].

The extraction of theories from nonprimary sources is a challenging task. First, there are no uniformly accepted criteria for identifying theories, which hampers the generation of predefined selection criteria. Second, the amount of information given about theories is often sparse. Our approach was to choose an epistemological stance that is relatively inclusive and to use the relatively generic structural components used for theory identification in a closely related field [38]. The multireviewer examination resulted in disagreements regarding the inclusion of six theories, of which four were finally excluded.

5.7 Future Work

Although our findings may be indicative for the field as a whole, further reviews are needed to establish their status for other types of empirical study for which theory is highly relevant, such as case studies, correlation/regression studies, simulations, etc. We focused on theories used in explanatory roles pertaining to the cause-effect relationships investigated in experiments. We did not include theories used in subsidiary and other roles in the results of this review. However, we did find a substantial amount of such uses of theory. Although these uses of theory are secondary when it comes to causal explanation, they are of great value for other aspects of empirical studies and for the field as a whole. Research is currently in progress on these issues.

6 Conclusion

This systematic review investigated the use of theories in software engineering experiments. Our observations reveal that about a quarter of the surveyed articles involve theory in explanatory roles pertaining to the cause-effect relationships under investigation. Most of the articles use theory to justify or motivate experimental research questions. However, we found no evidence of theory-driven research, in the sense of empirically based theories that encompass and define the research questions of empirical software engineering. In particular, the theories that are used do not, to any extent, function as frameworks in which issues are discussed across articles by different authors.

Nevertheless, theories *are* being used, and disciplines with strong theoretical traditions are being consulted when seeking explanation in empirical studies for software engineering. Even among the reviewed articles that do not use theory, there are explicit calls for such use. We concur with this view; to advance the body of knowledge, theory should be an integral part of empirical studies in software engineering. We hope that the information provided in this study about theories that are used in software engineering experiments will contribute to achieving this goal.

Acknowledgments

The authors are grateful to Bente Anda, Alastair Hannay, Magne Jørgensen, Vigdis By Kampenes, Amela Karahasanović, Barbara Kitchenham, and Chris Wright for useful feedback and enlightening discussions. The authors are also grateful to the anonymous referees for pertinent and insightful comments. Thanks to Ove Hansen, Nils-Kristian Liborg, Anette Rekdal, and several of the above for their work in extracting articles describing experiments, to Jørgen Busvold and Magnar Martinsen for assistance in compiling data, and to Chris Wright for proofreading the paper.

References

[1] "ACM Computing Classification System," http://www.acm.org/class, 2004.
[2] 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.
[3] *Principles of Forecasting: A Handbook for Researchers and Practitioners,* J.S. Armstrong, ed. Kluwer Academic, 2001.
[4] M.E. Atwood and H.R. Ramsey, "Cognitive Structures in the Comprehension and Memory of Computer Programs: An Investigation of Computer Program Debugging," Technical Report TR-78-A21, U.S. Army Research Inst. for the Behavioral and Social Sciences, 1978.
[5] S.B. Bacharach, "Organizational Theories: Some Criteria for Evaluation," *Academy of Management Rev.,* vol. 14, no. 4, pp. 496-515, 1989.
[6] V.R. Basili, "Editorial," *Empirical Software Eng.,* vol. 1, no. 2, pp. 105-108, Jan. 1996.
[7] A.C. Benander, B.A. Benander, and J. Sang, "An Empirical Analysis of Debugging Performance—Differences between Iterative and Recursive Constructs," *J. Systems and Software,* vol. 54, no. 1, pp. 17-28, Sept. 2000.
[8] A. Bianchi, F. Lanubile, and G. Visaggio, "A Controlled Experiment to Assess the Effectiveness of Inspection Meetings," *Proc. Seventh Int'l Symp. Software Metrics,* pp. 42-50, 2001.
[9] B.W. Boehm and B. Clark, "Cost Models for Future Life Cycle Processes: COCOMO 2," *Annals Software Eng.,* vol. 1, pp. 57-94, 1995.
[10] D. Borsboom, G.J. Mellenbergh, and J. Van Heerden, "The Theoretical Status of Latent Variables," *Psychological Rev.,* vol. 110, no. 2, pp. 203-219, 2003.
[11] T. Boswell and C. Brown, "The Scope of General Theory," *Sociological Methods & Research,* vol. 28, no. 2, pp. 154-185, 1999.
[12] L.F. Bourne, R.L. Dominowski, E.F. Loftus, and A.F. Healy, *Cognitive Processes.* Prentice Hall, 1986.
[13] A. Brooks, F. Utbult, C. Mulligan, and R. Jeffery, "Early Lifecycle Work: Influence of Individual Characteristics, Methodological Constraints, and Interface Constraints," *Empirical Software Eng.,* vol. 5, no. 3, pp. 269-285, Nov. 2000.
[14] M. Bunge, *Scientific Research I: The Search for a System.* Springer Verlag, 1967.

[15] M. Bunge, "Realism and Antirealism in Social Science," *Theory and Decision,* vol. 35, no. 3, pp. 207-235, 1993.
[16] J.T. Cacioppo, G.R. Semin, and G.G. Berntson, "Realism, Instrumentalism, and Scientific Symbiosis," *Am. Psychologist,* vol. 59, no. 4, pp. 214-223, 2004.
[17] B. Cohen, *Developing Sociological Knowledge: Theory and Method.* Prentice Hall, 1980.
[18] L.J. Cronbach, *Designing Evaluations of Social and Educational Programs.* Josey-Bass, 1982.
[19] 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.
[20] 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.
[21] D.E.H. Damian, A. Eberlein, M.L.G. Shaw, and B. Gaines, "Using Different Communication Media in Requirements Negotiation," *IEEE Software,* vol. 17, no. 3, pp. 28-36, May/June 2000.
[22] D.D. Davis and C.A. Holt, *Experimental Economics.* Princeton Univ. Press, 1993.
[23] M.S. Davis, "That's Interesting! Towards a Phenomenology of Sociology and a Sociology of Phenomenology," *Philosophy of the Social Sciences,* vol.1, pp. 309-344, 1971.
[24] P.J. Dimaggio, "Comments on 'What Theory Is *Not*," *Administrative Science Quarterly,* vol. 40, pp. 391-397, 1995.
[25] R. Dubin, *Theory Building.* Free Press, 1969.
[26] R. Dubin, *Theory Development.* Free Press, 1978.
[27] A. Endres and D. Rombach, *A Handbook of Software and Systems Engineering,* Fraunhofer IESE Series on Software Eng. Pearson Education Limited, 2003.
[28] R.P. Feynman, *QED—The Strange Theory of Light and Matter.* Penguin Science, 1985.
[29] R.P. Feynman, *The Meaning of It All: Thoughts of a Citizen-Scientist.* Perseus Books, 1998.
[30] P. Fonagy and M. Target, *Psychoanalytic Theories. Perspectives from Developmental Psychopathology.* Whurr, 2003.
[31] W.B. Frakes and T.P. Pole, "An Empirical Study of Representation Methods for Reusable Software Components," *IEEE Trans. Software Eng.,* vol. 20, no. 8, pp. 617-630, Aug. 1994.
[32] R. Franck, *The Explanatory Power of Models.* Kluwer Academic, 2002.
[33] B. Freimut, O. Laitenberger, and S. Biffl, "Investigating the Impact of Reading Techniques on the Accuracy of Different Defect Content Estimation Techniques," *Proc. Seventh Int'l Symp. Software Metrics,* pp. 51-62, 2001.
[34] K.J. Gergen, "Correspondence Versus Autonomy in the Language of Understanding Human Action," *Metatheory in Social Science,* D.W. Fiske and R.A. Schweder, eds. Univ. of Chicago Press, pp. 136-162, 1986.
[35] G. Gigerenzer, "Mindless Statistics," *J. Socio-Economics,* vol. 33, pp. 587-606, 2004.
[36] R.L. Glass, V. Ramesh, and I. Vessey, "An Analysis of Research in Computing Disciplines," *Comm. ACM,* vol. 47, no. 6, pp. 89-94, June 2004.
[37] 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.
[38] S. Gregor, "The Nature of Theory in Information Systems," *MIS Quarterly,* vol. 30, no. 3, pp. 491-506, Sept. 2006.
[39] T.R. Gruber, "A Translation Approach to Portable Ontology Specifications," *Knowledge Acquisition,* vol. 5, no. 2, pp. 199-220, 1993.
[40] S. Hawking and R. Penrose, *The Nature of Space and Time.* Princeton Univ. Press, 1996.
[41] S.M. Henry and K.T. 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.
[42] R.L. Henschel, "Sociology and Prediction," *Am. Sociologist,* vol. 6, pp. 213-220, 1971.
[43] J.D. Herbsleb and A. Mockus, "Formulation and Preliminary Test of an Empirical Theory of Coordination in Software Engineering," *Proc. European Software Eng. Conf./ACM SIGSOFT Symp. Foundations of Software Eng.,* pp. 112-121, 2003.
[44] U. Hoffrage, R. Hertwig, and G. Gigerenzer, "Hindsight Bias: A By-Product of Knowledge Updating?" *J. Experimental Psychology: Learning, Memory, and Cognition,* vol. 26, pp. 566-581, 2000.
[45] G.C. Homans, "Bringing Men Back In," *The Philosophy of Social Explanation,* A. Ryan, ed., Oxford Univ. Press, pp. 50-64, 1973.
[46] F. Houdek, "External Experiments—A Workable Paradigm for Collaboration between Industry and Academia," *Lecture Notes on Empirical Software Eng.,* vol. 12, N. Juristo and A.M. Moreno, eds., chapter 4, World Scientific, vol. 12, 2003.
[47] 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.
[48] P. Humphreys, "Mathematical Modeling in Social Sciences," *Philosophy of the Social Sciences,* S.P. Turner and P.A. Roth, eds. Blackwell, 2003.
[49] "IEEE Keyword Taxonomy," http://www.computer.org/mc/keywords/software.htm, 2004.
[50] R. Jeffery and L.G. Votta, "Guest Editor's Special Section Introduction," *IEEE Trans. Software Eng.,* vol. 25, no. 4, pp. 435-437, July/Aug. 1999.
[51] *Handbook of Cognitive Psychophysiology: Central and Autonomic Nervous System Approaches,* J.R. Jennings and M.G.H. Coles, eds. Wiley, 1991.
[52] E.E. Jones, "Major Developments in Social Psychology during the Five Past Decades," *The Handbook of Social Psychology,* third ed., G. Lindzey and E. Aronsen, eds., chapter 2, pp. 47-107, Random House, 1985.
[53] M. Jørgensen and G.J. Carelius, "An Empirical Study of Software Project Bidding," *IEEE Trans. Software Eng.,* vol. 30, no. 12, pp. 953-969, Dec. 2004.
[54] N. Juristo and A.M. Moreno, *Basics of Software Engineering Experimentation.* Kluwer Academic, 2003.
[55] A. Kaplan, *The Conduct of Inquiry,* Chandler, 1964.
[56] B.A. Kitchenham, "Procedures for Performing Systematic Reviews," Technical Report TR/SE-0401, Keele Univ., and Technical Report 0400011T.1, NICTA, 2004.
[57] 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.
[58] J. Koskinen, "Experimental Evaluation of Hypertext Access Structures," *Software Maintenance and Evolution,* vol. 14, no. 2, pp. 83-108, 2002.
[59] O. Laitenberger, C. Atkinson, M. Schlich, and K. El Emam, "An Experimental Comparison of Reading Techniques for Defect Detection in UML Design Documents," *J. Systems and Software,* vol. 53, no. 2, pp. 183-204, Aug. 2000.
[60] O. Laitenberger and H.M. Dreyer, "Evaluating the Usefulness and the Ease of Use of A Web-Based Inspection Data Collection Tool," *Proc. Fifth Int'l Symp. Software Metrics,* pp. 122-132, 1998.
[61] O. Laitenberger and H.D. Rombach, "(Quasi-)Experimental Studies in Industrial Settings," *Lecture Notes on Empirical Software Engineering,* vol. 12, N. Juristo and A.M. Moreno, eds., chapter 5, pp. 167-227, World Scientific, 2003.
[62] K. Lewin, "The Research Center for Group Dynamics at Massachusetts Institute of Technology," *Sociometry,* vol. 8, pp. 126-135, 1945.
[63] C.E. Lindblom, "Alternatives to Validity: Some Thoughts Suggested by Campbell's Guidelines," *Knowledge Creation, Diffusion, Utilization,* vol. 8, pp. 509-520, 1987.
[64] J.W. Lucas, "Theory-Testing, Generalization, and the Problem of External Validity," *Sociological Theory,* vol. 21, no. 3, pp. 236-253, 2003.
[65] S.A. Lynham, "The General Method of Theory-Building Research in Applied Disciplines," *Advances in Developing Human Resources,* vol. 4, no. 3, pp. 221-241, Aug. 2002.
[66] B. Markovsky, "The Structure of Theories," *Group Processes,* M. Foschi and E.J. Lawler, eds., pp. 3-24, Nelson-Hall, 1994.
[67] 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.
[68] E. McMullin, "A Case for Scientific Realism," *Scientific Realism,* J. Leplin, ed., Univ. of California Press, 1984.
[69] J. Miller, M. Wood, and M. Roper, "Further Experiences with Scenarios and Checklists," *Empirical Software Eng.,* vol. 3, no. 1, pp. 37-64, Mar. 1998.
[70] B. Mohr, *Explaining Organizational Behavior.* Josey Bass, 1982.

[71] E. Nagel, *The Structure of Science: Problems in the Logic of Scientific Explanation.* Harcourt, Brace and World, 1961.
[72] E. Nagel, *The Structure of Science.* Hackett, 1979.
[73] D.A. Norman, *The Design of Everyday Things.* Basic Books, 1988.
[74] D. Papineau, "Philosophy of Science," *The Blackwell Companion to Philosophy,* N. Bunnin and E.P. Tsui-James, eds. Blackwell, 1996.
[75] S.L. Pfleeger, "Albert Einstein and Empirical Software Engineering," *Computer,* vol. 32, no. 10, pp. 32-38, Oct. 1999.
[76] A.A. Porter, H.P. Siy, C.A. Toman, and L.G. Votta, "An Experiment to Assess the Cost-Benefits of Code Inspections in Large Scale Software Development," *IEEE Trans. Software Eng.,* vol. 23, no. 6, pp. 329-346, June 1997.
[77] L. Prechelt, B. Unger, W.F. Tichy, P. Brossler, and L.G. Votta, "A Controlled Experiment in Maintenance: Comparing Design Patterns to Simpler Solutions," *IEEE Trans. Software Eng.,* vol. 27, no. 12, pp. 1134-1144, Dec. 2001.
[78] L. Prechelt, B. Unger-Lamprecht, M. Philippsen, and W.F. Tichy, "Two Controlled Experiments Assessing the Usefulness of Design Pattern Documentation in Program Maintenance," *IEEE Trans. Software Eng.,* vol. 28, no. 6, pp. 595-606, June 2002.
[79] R. Rorty, *Consequences of Pragmatism.* Univ. of Minnesota Press, 1982.
[80] A. Rosenberg, *Philosophy of Science, A Contemporary Introduction.* Routledge, 2001.
[81] P.J. Runkel and M. Runkel, *A Guide to Usage for Writers and Students in the Social Sciences.* Rowman and Allanheld, 1984.
[82] M. Ruse, "Theory," *The Oxford Companion to Philosophy,* T. Honderich, ed., Oxford Univ. Press, pp. 870-871, 1995.
[83] D. Sandborg, "Mathematical Explanation and the Theory of Why-Questions," *British J. Philosophy of Science,* vol. 49, no. 4, pp. 603-624, Dec. 1998.
[84] D.P. Schwab, "Construct Validity in Organizational Behavior," *Research in Organizational Behavior,* vol. 2, B.M. Staw and L.L. Cummings, eds., pp. 3-43, AI Press, 1980.
[85] W.R. Shadish, T.D. Cook, and D.T. Campbell, *Experimental and Quasi-Experimental Designs for Generalized Causal Inference.* Houghton Mifflin, 2002.
[86] H.A. Simon, *The Sciences of the Artificial,* third ed., MIT Press, 1996.
[87] 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.
[88] R.I. Sutton and B.M. Staw, "What Theory Is *Not*," *Administrative Science Quarterly,* vol. 40, pp. 371-384, 1995.
[89] W.F. Tichy, "Should Computer Scientist Experiment More? 16 Excuses to Avoid Experimentation," *Computer,* vol. 31, no. 5, pp. 32-40, May 1998.
[90] W.M.K. Trochim, *The Research Methods Knowledge Base.* Atomic Dog, 2001.
[91] A.H. Van de Ven, "Nothing Is Quite So Practical as a Good Theory," *Academy of Management Rev.,* vol. 14, no. 4, pp. 486-489, 1989.
[92] K.G. van den Berg and P.M. van den Broek, "Programmers' Performance on Structured versus Nonstructured Function Definitions," *Information and Software Technology,* vol. 38, no. 7, pp. 477-492, July 1996.
[93] B. Van Fraassen, *The Scientific Image.* Oxford Univ. Press, 1980.
[94] I. Vessey, "Toward a Theory of Computer Program Bugs: An Empirical Test," *Int'l J. Man-Machine Studies,* vol. 30, no. 1, pp. 23-46, 1989.
[95] D.G. Wagner, "The Growth of Theories," *Group Processes,* M. Foschi and E.J. Lawler, eds., pp. 25-42, Nelson-Hall, 1994.
[96] 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.
[97] R. Weber, "Editor's Comments," *MIS Quarterly,* vol. 27, no. 3, pp. iii-xii, Sept. 2003.
[98] M. Webster Jr., "Experimental Methods," *Group Processes,* M. Foschi and E.J. Lawler, eds. Nelson-Hall, pp. 43-69, 1994.
[99] K.E. Weick, "Theory Construction as Disciplined Imagination," *Academy of Management Rev.,* vol. 14, no. 4, pp. 516-531, 1989.
[100] K.E. Weick, "What Theory Is *Not,* Theorizing *Is*," *Administrative Science Quarterly,* vol. 40, pp. 385-390, 1995.
[101] D.A. Whetten, "What Constitutes a Theoretical Contribution," *Academy Of Management Rev.,* vol. 14, no. 4, pp. 490-495, 1989.
[102] C. Wohlin, P. Runeson, M. Höst, M.C. Ohlsson, B. Regnell, and A. Wesslen, *Experimentation in Software Engineering: An Introduction.* Kluwer Academic, 1999.
[103] R.K. Yin, *Case Study Research: Design and Methods,* third ed., Applied Social Research Methods Series, vol. 5. Sage, 2003.
[104] Z. Zhang, V. Basili, and B. Shneiderman, "Perspective-Based Usability Inspection: An Empirical Validation of Efficacy," *Empirical Software Eng.,* vol. 4, no. 1, pp. 43-69, Mar. 1999.
[105] 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.
[106] 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.
[107] R.N. Anthony and J. Dearden, *Management Control Systems.* Richard D. Irwin, 1980.
[108] E. Arisholm, D.I.K. Sjøberg, and M. Jørgensen, "Assessing the Changeability of Two Object-Oriented Design Alternatives—A Controlled Experiment," *Empirical Software Eng.,* vol. 6, no. 3, pp. 231-237, Sept. 2001.
[109] K.J. Arrow, *Essays in the Theory of Risk-Bearing.* North-Holland, 1970.
[110] R. Atkinson, D. Herrmann, and K. Wescourt, "Search Processes in Recognition Memory," *Theories in Cognitive Psychology: The Loyola Symposium,* R.L. Solso, ed. Lawrence Earlbaum Assoc., 1974.
[111] R. Atkinson and R. Shiffrin, "Human Memory: A Proposed System and Its Control Process," *Advances of Psychological Theory of Learning and Motivation Research and Theory,* vol. 2, K.W. Spence and J.D. Spence, eds., Academic, 1968.
[112] A.O. Awani, *Data Processing Project Management,* Petrocelli, 1986.
[113] H. Barki, S. Rivard, and J. Talbot, "Toward an Assessment of Software Development Risk," *J. Management Information Systems,* vol. 10, no. 2, pp. 203-225, 1993.
[114] R.M. Belbin, *Management Teams.* Wiley, 1981.
[115] R.M. Belbin, *Team Roles at Work.* Butterworth-Heinemann, 1993.
[116] B.J. Biddle, *Role Theory: Expectation, Identities, and Behaviors.* Academic Press, 1979.
[117] R. Brooks, "Towards a Theory of the Comprehension of Computer Programs," *Int'l J. Man-Machine Studies,* vol. 18, no. 6, pp. 543-554, June 1983.
[118] R. Brooks, "Using a Behavioral Theory of Program Comprehension in Software Engineering," *Proc. Third Int'l Conf. Software Eng.,* pp. 196-201, 1998.
[119] J.M. Burkhardt, F. Détienne, and S. Wiedenbeck, "Object-Oriented Program Comprehension: Effect of Expertise, Task and Phase," *Empirical Software Eng.,* vol. 7, no. 2, pp. 115-156, June 2002.
[120] W.G. Chase and H.A. Simon, "The Mind's Eye in Chess," *Visual Information Processing,* W.G. Chase, ed., Academic, 1973.
[121] N. Chomsky, "Aspects of Theory of Syntax," technical report, MIT Press, 1965.
[122] A. Collins and E. Loftus, "A Spreading Activation Theory of Semantic Processing," *Psychology Rev.,* vol. 82, pp. 407-428, 1975.
[123] R.L. Daft and R.K. Lengel, "Organizational Information Requirements, Media Richness and Structural Design," *Management Science,* vol. 32, no. 5, pp. 554-571, 1986.
[124] F.D. Davis, R.P. Bagozzi, and P.R. Warshaw, "User Acceptance of Computer Technology: A Comparison of Two Theoretical Models," *Management Science,* vol. 35, no. 8, pp. 982-1003, Aug. 1989.
[125] D. Day, M. Ahuja, and L. Scott, "Constraints in Design Engineering: A Report of Research in Progress," *Proc. Eighth Australian Conf. Information Systems,* pp. 509-516, 1997.
[126] K. Duncker, "On Problem Solving," *Psychology Monograph,* vol. 58, chapter 5, Am. Psychological Assoc., 1945.
[127] J. Dvorak, "Conceptual Entropy and Its Effect on Class Hierarchies," *IEEE Computer,* vol. 27, no. 6, pp. 59-63, June 1994.
[128] W. Edwards, "Conservatism in Human Information Processing," *Formal Representation of Human Judgment,* B. Kleinmuntz, ed., Wiley, 1968.
[129] E. Feigenbaum, "Information Processing and Memory," *Models of Memory,* D.A. Norman, ed. Academic, 1970.
[130] J. Greeno, "The Structure of Memory and the Process of Solving Problems," *Contemporary Issues in Cognitive Psychology: The Loyola Symposium,* R.L. Solso, ed., Wiley, 1973.
[131] D.L. Harnett and L.L. Cummings, *Bargaining Behavior: An International Study.* Dame, 1980.

[132] W.E. Hick, "On the Rate of Gain of Information," *Quarterly J. Experimental Psychology,* vol. 4, pp. 11-26, 1952.
[133] R. Hogarth, "Beyond Discrete Biases: Functional and Dysfunctional Aspects of Judgmental Heuristics," *Psychological Bull.,* vol. 90, no. 2, pp. 197-217, 1981.
[134] R. Hyman, "Stimulus Information as a Determinant of Reaction Time," *J. Experimental Psychology,* vol. 45, pp. 188-196, 1953.
[135] 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.
[136] J. Klayman and Y.W. Ha, "Confirmation, Disconfirmation, and Information In Hypothesis Testing," *Psychological Rev.,* vol. 94, no. 2, pp. 221-228, 1987.
[137] E.S. Knowles, "From Individuals to Group Members: A Dialectic for the Social Sciences," *Personality, Roles and Social Behavior,* W. Ickes and E.S. Knowles, eds., Springer, 1982.
[138] N. Kogan and M.A. Wallach, *Risk Taking: A Study in Cognition and Personality.* Holt, Rinehart, and Winston, 1964.
[139] R. Krovi and A. Chandra, "User Cognitive Representations: The Case for an Object-Oriented Model," *J. Systems and Software,* vol. 43, no. 3, pp. 165-176, Nov. 1998.
[140] 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.
[141] L.P.W. Land, C. Sauer, and R. Jeffery, "The Use of Procedural Roles in Code Inspections: An Experimental Study," *Empirical Software Eng.,* vol. 5, no. 1, pp. 11-34, Mar. 2000.
[142] S. Letovsky, "Cognitive Processes in Program Comprehension," *Proc. First Workshop Empirical Studies of Programmers,* pp. 58-79, 1986.
[143] S. Letovsky and E. Soloway, "Delocalized Plans and Program Comprehension," *IEEE Software,* vol. 3, no. 3, pp. 41-49, May 1986.
[144] 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.
[145] I. Lorge, D. Fox, J. Davitz, and M. Brenner, "A Survey of Studies Contrasting the Quality of Group Performance and Individual Performance, 1920-1957," *Psychological Bull.,* vol. 55, no. 6, pp. 337-371, 1958.
[146] J.E. McGrath and A.B. Hollingshead, *Groups Interacting with Technology: Ideas, Evidence, Issues and an Agenda.* Sage, 1994.
[147] B.A. Mellers and S. Chang, "Representations of Risk Judgments," *Organizational Behavior and Human Decision Processes,* vol. 57, no. 2, pp. 167-184, 1994.
[148] G.A. Miller, "The Magical Number Seven, Plus or Minus Two: Some Limits on Our Capacity for Processing Information," *Psychological Rev.,* vol. 63, pp. 81-97, 1956.
[149] A. Newell and H.A. Simon, *Human Problem Solving.* Prentice Hall, 1972.
[150] A. Paivio, *Imagery and Verbal Processes.* Holt, Rinehart, and Winston, 1971.
[151] R. Palmer and I. Rock, "Rethinking Perceptual Organization: The Role of Uniform Connectedness," *Psychometric Bull. and Rev.,* vol. 1, no. 1, pp. 29-55, 1994.
[152] N. Pennington, "Comprehension Strategies in Programming," *Proc. Second Workshop Empirical Studies of Programmers,* pp. 100-113, 1987.
[153] N. Pennington, "Stimulus Structures and Mental Representations in Expert Comprehension of Computer Programs," *Cognitive Psychology,* vol. 19, pp. 295-341, 1987.
[154] 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.
[155] 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.
[156] P. Reisner, "Human Factors Studies of Database Query Languages: A Survey and Assessment," *ACM Computing Surveys,* vol. 13, pp. 13-31, 1981.
[157] E. Rosch, "Principles of Categorization," *Cognition and Categorization,* Lawrence Erlbaum, 1978.
[158] T.R. Sarbin, "Role Theory," *Handbook of Social Psychology,* G. Lindzey, ed., Addison-Wesley, 1954.
[159] C. Sauer, R. Jeffery, L.P.W. Land, and P. Yetton, "Understanding and Improving the Effectiveness of Software Development Technical Reviews: A Behaviourally Motivated Programme of Research," *IEEE Trans. Software Eng.,* vol. 26, no. 1, pp. 1-14, Jan. 2000.
[160] C.E. Shannon, "A Mathematical Theory of Communication," *Bell System Technical J.,* vol. 27, pp. 379-423, 1948.
[161] B. Shneiderman, "Measuring Computer Program Quality and Comprehension," *Int'l J. Man-Machine Studies,* vol. 9, no. 1, pp. 465-478, 1977.
[162] L. Short, E. Williams, and B. Christie, *The Social Psychology of Telecommunications.* John Wiley & Sons, 1976.
[163] S.B. Sitkin and A.L. Pablo, "Reconceptualizing the Determinants of Risk Behavior," *Academy of Management Rev.,* vol. 17, no. 1, pp. 9-38, 1992.
[164] E.E. Smith, "Concepts and Thought," *The Psychology of Human Thought,* Cambridge Univ. Press, 1988.
[165] E.E. Smith and D.L. Medin, *Categories and Concepts.* Harvard Univ. Press, 1981.
[166] E. Soloway, J. Pinto, S. Letovsky, D. Littman, and R. Lampert, "Designing Documentation to Compensate for Delocalized Plans," *Comm. ACM,* vol. 31, no. 11, pp. 1259-1267, Nov. 1988.
[167] I.D. Steiner, *Group Process and Productivity,* Academic, 1972.
[168] E. Tulving, "Episodic and Semantic Memory," *Organization and Memory,* E. Tulving and W. Donaldson, eds., Academic, 1972.
[169] A. Tversky and D. Kahneman, "Judgement under Uncertainty: Heuristics and Biases," *Science,* vol. 185, no. 27, pp. 1124-1131, Sept. 1974.
[170] T.A. van Dijk and W. Kintsch, *Strategies of Discourse Comprehension.* Academic, 1983.
[171] I. Vessey, "Cognitive Fit: A Theory-Based Analysis of the Graphs versus Tables Literature," *Decision Sciences,* vol. 22, no. 2, pp. 219-240, 1991.
[172] I. Vessey and D. Galletta, "Cognitive Fit: An Empirical Study of Information Acquisition," *Information Systems Research,* vol. 2, pp. 63-84, Mar. 1991.
[173] R. Vinter, M. Loomes, and D. Kornbrot, "Applying Software Metrics to Formal Specifications: A Cognitive Approach," *Proc. Fifth Int'l Symp. Software Metrics,* pp. 216-223, 1998.
[174] A. von Mayrhauser and S. Lang, "A Coding Scheme to Support Systematic Analysis of Software Comprehension," *IEEE Trans. Software Eng.,* vol. 25, no. 4, pp. 526-540, July/Aug. 1999.
[175] A. von Mayrhauser and A.M. Vans, "Industrial Experience with an Integrated Code Comprehension Model," *Software Eng. J.,* vol. 10, no. 5, pp. 171-182, Sep. 1995.
[176] A. von Mayrhauser, A.M. Vans, and A.E. Howe, "Program Understanding Behaviour during Enhancement of Large-Scale Software," *Software Maintenance: Research and Practice,* vol. 9, pp. 299-327, 1997.
[177] M. Wertheimer, "Untersuchungen Zurlehre von der Gestalt: II," *Psychologische Forschung,* vol. 4, pp. 301-350, 1923.
[178] M. Wertheimer, *Productive Thinking.* Harper & Row, 1959.
[179] W.A. Wickelgren, *How to Solve Problems.* Freeman, 1974.
[180] W.A. Wickelgren, *Learning and Memory,* Personality, Roles and Social Behavior. Prentice Hall, 1977.
[181] S. Wiedenbeck, "Novice/Expert Differences in Programming Skills," *Int'l J. Man-Machine Studies,* vol. 23, no. 4, pp. 383-390, 1985.
[182] M.Y.M. Yen and R.W. Scamell, "A Human Factors Experimental Comparison of SQL and QBE," *IEEE Trans. Software Eng.,* vol. 19, no. 4, pp. 390-409, Apr. 1993.

Jo E. Hannay received the MSc degree in computer science from the University of Oslo in 1995 and the PhD degree in specification and data refinement using type theory and logic from the University of Edinburgh in 2001. He has two years of experience as a software developer in the insurance industry. He is currently a post-doctoral fellow at Simula Research Laboratory. His interests include the use and development of theories in empirical software engineering, the nature of knowledge useful to software engineering, and logics and formalisms for describing knowledge.

Dag I.K. Sjøberg received the MSc degree in computer science from the University of Oslo in 1987 and the PhD degree in computing science from the University of Glasgow in 1993. He has five years of industry experience as a consultant and group leader. He is now the research director of the Department of Software Engineering at Simula Research Laboratory, and a professor of software engineering in the Department of Informatics at the University of Oslo. Among his research interests are research methods in empirical software engineering, software processes, software process improvement, software effort estimation, and object-oriented analysis and design. He is a member of the IEEE and the IEEE Computer Society.

Tore Dybå received the MSc degree in electrical engineering and computer science from the Norwegian Institute of Technology in 1986 and the PhD degree in computer and information science from the Norwegian University of Science and Technology in 2001. He is the chief scientist at SINTEF ICT and a visiting scientist at the Simula Research Laboratory. Dr. Dybå worked as a consultant for eight years in Norway and Saudi Arabia before he joined SINTEF in 1994. His research interests include empirical and evidence-based software engineering, software process improvement, and organizational learning. Dr. Dybå is the author and coauthor of more than 50 publications appearing in international journals, books, and conference proceedings in the fields of software engineering and knowledge management. He is the principal author of the book *Process Improvement in Practice: A Handbook for IT Companies*, published as part of the Kluwer International Series in Software Engineering. He is a member of the International Software Engineering Research Network, the IEEE, the IEEE Computer Society, and the editorial board of *Empirical Software Engineering*.