Construct Validity in Software Engineering

Dag I. K. Sjøberg, *Member, IEEE*, and Gunnar R. Bergersen

Abstract—Empirical research aims to establish generalizable claims from data. Such claims involve concepts that often must be measured indirectly by using indicators. Construct validity is concerned with whether one can justifiably make claims at the conceptual level that are supported by results at the operational level. We report a quantitative analysis of the awareness of construct validity in the software engineering literature between 2000 and 2019 and a qualitative review of 83 articles about human-centric experiments published in five high-quality journals between 2015 and 2019. Over the two decades, the appearance in the literature of the term construct validity increased sevenfold. Some of the reviewed articles we reviewed employed various ways to ensure that the indicators span the concept in an unbiased manner. We also found articles that reuse formerly validated constructs. However, the articles disagree about how to define construct validity. Several interpret construct validity excessively by including threats to internal, external, or statistical conclusion validity. A few articles also include fundamental challenges of a study, such as cheating and misunderstandings of experiment material. The diversity of topics discussed makes us recommend a minimalist approach to construct validity. We propose seven guidelines to establish a common ground for addressing construct validity in software engineering.

Index Terms—Measurement, research quality, empirical research, systematic review, guidelines

1 INTRODUCTION

IN all sciences, one attempts to measure “things” that are not directly measurable. A set of indicators that are directly measurable may serve as a substitute for a single measure of the phenomenon or characteristic of interest. For example, a collection of key performance indicators (KPIs) is supposed express how a company performs.

In software engineering, studies typically investigate the effect of using various processes, methods, technologies, or practices for software development. Results from single studies need to be confirmed in replications. One premise for meaningful replications is that the individual studies are comparable; that is, the objects studied and the way they are measured should be the same or closely related. For example, a software development method may be evaluated with respect to the maintainability of the developed software. At the *conceptual level*, meaningful comparisons of alternative development methods require the concept of maintainability to be consistently defined. When differences between study results occur, the cause may be different definitions of maintainability rather than differences in the effects of the investigated methods. At the *operational level*, it is not sufficient that a concept has an adequate definition; one must also ensure that the concept is measured validly.

A concept that is not directly measurable, and therefore is represented by indicators at the operational level, is called a *construct*. The extent to which a concept definition is adequate and the indicators represent the concept is the issue of *construct validity* (CV) [1], [2]. The development and use of well-defined concepts are at the core of all sciences. Einstein [3, p. 674] stated: “... thinking without the positing of categories and of concepts in general would be as impossible as is breathing in a vacuum.”

CV is a prerequisite to making valid inferences from particular observations at the operational level in a study to general claims at the conceptual level. What is suggested in a paper may, therefore, not work in practice. Moreover, an unsatisfactory CV may be one reason why meta-analyses and systematic reviews often give inconclusive results.

Several papers have addressed the notion of CV in software engineering [4], [5], [6], [7]. However, the state of awareness and practice has not been systematically investigated before. In this article, we report a quantitative analysis of how the awareness of CV in software engineering has changed over time. The primary contribution of this article, however, is a qualitative review of CV discussions in 83 human-centric experiments published in high-quality journals. Based on the findings in the review, we propose a set of guidelines to improve the understanding and practice of CV in software engineering.

The remainder of this article is organized as follows. Section 2 defines and describes CV and the threats to it. Section 3 reports the analysis of CV awareness. Section 4 reports the review. Section 5 describes the guidelines and discusses limitations and related work. Section 6 concludes.

2 CONSTRUCT VALIDITY

2.1 Definition of Construct

Concepts are at the core of any science. Creating and understanding concepts are fundamental in the development and acquisition of knowledge. We use concepts to categorize physical and abstract phenomena. Moreover, concepts allow us to generalize the particulars and abstract the details, making complexity simpler to manage.

A concept has a name, an intension, and an extension, three components that are well described in the classical lit-

• D.I.K. Sjøberg, and G.R. Bergersen are with the Department of Informatics, University of Oslo, PO Box 1080 Blindern, NO-0316 Oslo, Norway
E-mail: [dagsj, gunnab]@ifi.uio.no.

Fig. 1. Elements of a construct.

erature of logic, linguistics, and philosophy.¹ The intension is the definition (meaning) or characteristics of the concept. The extension is the set of instances that the concept denotes. In computer science, this understanding of a concept has been directly applied: Apart from the name, the notion of *class* in object-oriented programming [8] is identical to the notion of concept described above.²

In an empirical science, concepts need to be measured. For some concepts, measurement is straightforward for practical purposes because the concept is sufficiently understood. For example, time can be directly measured using a time-keeping device. However, for concepts that are not directly measurable, one needs to use one or more *indicators* [2] to represent the concept at an observational level (Fig. 1). The process of determining such indicators is called *operationalization*. A concept that is operationalized into a set of indicators is called a *construct*. As such, a construct consists of a concept and its indicators.³

When the definition of the concept is the starting point for choosing indicators that reflect the concept, one has a reflective measurement model, which is the focus in this article. Conversely, when one chooses one or more indicators to form (define) a concept, one has a formative measurement model.

2.2 Validity and Validation of Constructs

CV is broadly about how well a set of indicators represent a concept. CV is a prerequisite for making valid inferences on relationships among concepts when the concepts are measured via indicators. For example, if an experiment claims a cause-effect relationship but the CV of the studied concepts is problematic, the relationship claimed may not be valid. Thus, the validity of constructs affects whether the use of the results of a research study to make decisions (policy changes, etc.) is justified.

1. "Intension and extension, in logic, correlative words that indicate the reference of a term or concept: "intension" indicates the internal content of a term or concept that constitutes its formal definition; and "extension" indicates its range of applicability by naming the particular objects that it denotes. For instance, the intension of "ship" as a substantive is "vehicle for conveyance on water", whereas its extension embraces such things as cargo ships, passenger ships, battleships, and sailing ships", <https://www.britannica.com/topic/intension>.

2. The intention is a description of its attributes, which are variables or methods that can be performed on those variables. The extension is the set of instances of the class that are created during program executions, called objects.

3. Note that some literature uses the term "abstract construct" to refer to what we choose to name "concept."

Fig. 2. Examples of construct underrepresentation.

Fig. 3. Examples of construct-representation bias.

A valid construct requires an unambiguous definition or understanding of the concept involved in the construct. Without such a basis, it is difficult to evaluate how well a set of indicators represent the concept.

Convergent validation [9] is a process used to investigate the extent to which a set of indicators span a concept. A traditional bottom-up approach is to use several variables with high, positive correlations as potential indicators of the same concept. Three frequently used techniques are internal consistency reliability (e.g., Cronbach's alpha), principal components analysis (PCA) and exploratory factor analysis [1]. A technique that involves several concepts simultaneously is the Multitrait-Multimethod Matrix (MTMM), which Campbell and Fiske [9] recommend for investigating convergent and discriminant validity. Each concept (trait) is measured by multiple methods, and each method is used to measure multiple concepts.

Confirmatory factor analysis was introduced by Jöreskog [10]. This technique requires an explicit, *a priori* model of the relationship between a concept (factor) and its indicators (observable variables). One tests whether the variables load on the same factor according to the model. Confirmatory factor analysis is used in structural equation modeling (SEM), which investigates relationships between concepts (called *latent variables* in SEM) based on observable variables.

Discriminant validation [9] is a process used to investigate the extent to which a set of indicators describes a specific concept only and not other concepts. One would like to determine whether two constructs that differ according to expectations (as specified in the intension of the concept) in fact diverge⁴ according to available data. Convergent and discriminant validation often use the same correlation-based techniques.

2.3 Threats to Construct Validity

One major category of threats to CV is a missing or inadequate definition of the concept. Two other major categories concern the operationalization of the concept: the indicators

4. The term *divergent validation* is often used synonymously with *discriminant validation*.

may underrepresent the concept or there may be bias in the representation.

2.3.1 Inadequate Definition of Concept

The threat to CV, referred to by Cook and Campbell [11] as “inadequate preoperational explication of constructs”, is informally explained by Trochim et al. [12]: “... *what this phrase means is that you didn’t do a good enough job defining what you meant by the construct [concept] before you tried to translate it into a measure ..*”. For example, the concept of lead time may have different definitions in a software engineering context [13]. One definition is that it denotes the amount of time between the proposal for a new feature or another request and its deployment in the customer’s environment. Another definition is that it denotes the amount of time that passes from the moment that the development team receives a request to the moment that it completes the work item. How well a set of indicators represent the concept of lead time will depend on which definition is chosen.

Mark [14] proposes four categories of errors that researchers might make when identifying constructs: (1) the identified concept⁵ may be at too general a level; (2) the concept may be at too specific a level; (3) the identified concept is not the right one; (4) the concept may consist of several subconcepts where only one of them is included in the operationalization, or vice versa; the concept is defined in terms of two subconcepts, when in fact there is only one.

A theoretical definition of a concept is based on an underlying theory that describes the concept. Theoretical definitions guide how to operationalize concepts. In the presence of a well-established substantive theory, previously validated measures are often available and should be used. For example, a theoretical definition of the concept of skill, a concept much used in the software engineering literature, was originally proposed by Fitts and Posner [15]. Unfortunately, most of the concepts involved in software engineering research are not theoretically defined, which is on a par with the fact that theories in general in the field are relatively little used [16].

2.3.2 Construct Underrepresentation

Construct underrepresentation [17] occurs when the set of indicators resulting from the operationalizations are too narrow to include all important aspects or dimensions of the concept that should be included.⁶ What these important aspects are, is relative to the purpose of the concept in the present setting. Such a consideration should be addressed from both a theoretical and an empirical perspective. To determine whether underrepresentation is present, one should confer the research literature that reports earlier use of the construct under investigation. It may also be useful to confer with practitioners that are domain experts to obtain additional viewpoints on central aspects of the involved concept that may be missing from the research literature. Successful operationalizations also require measurement expertise.

5. Mark [14] uses the terms *type* and *mechanism* instead of *concept*.

6. Underrepresentation has also been referred to as “incomplete representation” [18].

Fig. 2 (left) shows the result of a single (mono) operationalization; that is, only one indicator represents the concept. For example, when the number of bugs is the only indicator of software quality, the construct is underrepresented because additional aspects are needed to span the concept of software quality. In fact, the standard ISO25010 defines eight overall characteristics of software quality. Reliability is one the aspects, for which the number of bugs is only one of several possible indicators of reliability.

Detecting single operationalizations is trivial (counting to one). However, determining the quality of a single operationalization is not trivial. For example, measuring the concept “program length” in terms of lines of code (LOC) would not be a serious case of underrepresentation, but using LOC to measure the concept “system complexity” would be serious because important aspects of system complexity are not captured by LOC alone. A single operationalization as described above is also referred to as *mono-operation* in the CV literature [11].

Fig. 2 (right) shows another example of construct underrepresentation where there are multiple indicators, but they do not span the concept sufficiently because the indicators are clustered and unbalanced. An example is if software complexity is measured in terms of both LOC and function points. Even though there are two indicators that represent the concept of size well, they do not capture other aspects of software complexity.

A special case of imbalance occurs when two indicators overlap; that is, when they essentially measure the same thing. For example, answers to the questions “are you a parent?” and “do you have children?” would have a correlation that approaches unity, and only one of the questions should be used. So, although high values of Cronbach’s alpha of a questionnaire indicate high internal consistency reliability, values that are too high (typically from $> .80$ to $> .95$) indicate that the questions do not span the concept being measured and thus cause construct underrepresentation [19].

2.3.3 Construct-Representation Bias

The second major threat to CV is construct-representation bias.⁷ Even though an obvious goal is to obtain an “ideal” representation of a concept through its indicators, they will in practice misrepresent the concept to some extent; that is, the representation will be somewhat biased. (If the indicators fully represent a concept, they are actually not *indicators*. Instead, they are measures or variables that uniquely determine the concept. Evaluating CV would then be meaningless because there is nothing more to the concept than what the indicators define.) However, there is no way to measure the extent of a bias because there is no direct way of measuring the concept itself; that is why we need indicators.

We consider two kinds of construct-representation bias. First, one or more indicators may be outside the scope of the indicators (Fig. 3, left); that is, the indicator does not follow “naturally” from the definition of the concept. The

7. Messick [17] uses the term *construct-irrelevant variance*. However, we consider that term to be more appropriate as part of measurement theory, while *construct-representation bias* describes the issue better at the same conceptual level as *construct underrepresentation*.

Fig. 4. Types of constructs in experiments.

indicator may only indirectly represent the concept and may be a better indicator of another concept. For example, LOC would be an inappropriate measure of usability, which is one of the aspects of quality included in ISO25010. There is some relationship between usability and LOC, of course; if there is hardly any LOC, the usability would certainly be unsatisfactory, but LOC is nevertheless a weak indicator of usability.

Another kind of bias occurs when multiple indicators systematically misrepresent the concept (Fig. 3, right), which is the case when the error variance of the indicators is systematic. For example, if the values of several indicators are collected using the same method, any error caused by that method will be systematic, which is referred to as *common-method variance*. The operationalization will then face *mono-method bias* [11]. For example, in investigating the effect of education on the quality of developed code, a mono-method bias would be in play if the data on education is collected only through self-reports. Such a mono-method bias would not exist if data on education were also collected from university transcripts. In addition, even when several indicators are used to represent education (i.e., there is no mono-operation), a mono-method bias would exist if the same method is used for all the indicators. To avoid mono-method bias, the quality of code, for example, may be evaluated by using objective code metrics provided by tools and the subjective opinions of experts.

A construct with only one indicator has *mono-operation bias* [11] and thus faces two threats. Underrepresentation is present because there is only a single indicator. Construct-representation bias is also present because any measurement error in the indicator will be included in the representation of the construct.

2.4 Construct Validity in Experiments

In this article, we analyze the understanding of CV in experiments because they are usually more uniformly reported and thus easier to investigate and compare than other kinds of study. According to Shadish et al. [20], an experiment is a study in which an intervention (often called

treatment) is deliberately introduced to observe its effects. Research questions that involve experiments express cause-effect relationships at the conceptual level, while the data analyses used to investigate such relationships occur at the operational level. Through the design of the experiment, plausible inferences can be made about causal relationships from observational data by using correlations.

Fig. 4 shows a model of the cause-effect relationship and what affects that relationship. The figure also shows the terminology that we will use when reporting on CV in the investigated articles. Instead of the term indicator, we will use term variable, more specifically, independent and dependent variables, because those terms are well established in the software engineering experiment literature. In our terminology, indicators are variables, but not vice versa: an indicator is a particular variable that requires a concept, whereas a variable is an operational term that may or may not indicate anything other than what is observed. Thus, a variable is a named placeholder associated with a value, which may vary. In software engineering, the terms metrics and measures are also used at the operational level, although they are not the same: while "... every measure is a 'metric', the converse is certainly not true" [21].⁸

Fig. 4 also shows a *moderator* and *mediator* construct appearing between the cause and effect constructs. A moderator changes (moderates) the *strength* and/or *direction* of a causal relationship [22]. At the operational level, a moderator variable affects the size or direction of a correlation or adds an interaction effect in regression and variance analyses. A mediator concept is on the chain from the cause concept to the effect concept. For example, the effect of programming experience on programming skill has been found to be mediated by programming knowledge [23]; that is, programming experience affects programming knowl-

8. The particular terms used in this area vary across different disciplines; for example, among synonyms to "dependent variable" are "criterion", "experimental variable", "explained variable", "measured variable", "outcome variable", "output variable", "predicted variable", "regressand", "response variable", and "responding variable". Furthermore, in many disciplines, researchers may use the terms "independent variable" and "dependent variable" at the conceptual level.

Fig. 5. The construct "CV awareness in SE literature"

edge, which in turn affects programming skill. Note that in practice, both concepts and variables may be defined at several levels.

Operationalization also relates to research questions and hypotheses. Research questions concern relationships between concepts, while hypotheses investigate relationships among variables at the operational level. The weaker the CV, the less would the hypothesis testing answer the research questions.

3 AWARENESS OF CONSTRUCT VALIDITY IN SOFTWARE ENGINEERING

This section reports on the awareness of CV as reflected in the software engineering research literature. The construct is illustrated in Figure 5. This section also reports the awareness of CV in a particular sample of journals and experiments. Underrepresentation and other kinds of threats to CV of this construct are discussed in Section 5.2.

3.1 Data Collection

We identified the proportion of documents that include "construct validity" in the general research literature of software engineering, more specifically in five leading software engineering journals, and even more specifically in articles that report human-centric experiments in these five journals.

3.1.1 CV in the General Software Engineering Literature

As a source for the general literature, we used the documents in Google Scholar produced between 2000 and 2019. The search was conducted in February 2021. As a coarse-grained indicator of awareness, we used the proportion of software engineering documents that contains the term "construct validity"; that is, the number of documents containing both the terms "construct validity" and "software engineering" divided by those documents containing only the term "software engineering". Further details, including the Python search script, can be found in the supplemental material at <http://...>⁹

3.1.2 CV in Software Engineering Journals

We wanted to investigate the awareness of CV in the software engineering research literature that report empirical studies and where one is likely to find discussions on CV. We selected five journals with a high journal impact factor [24]: *IEEE Transactions on Software Engineering* (TSE),

9. To editor/reviewers: to be added

TABLE 1
Number of articles identified in the steps of the review

Journal	Articles #	Search #	Exp. #	Discuss CV # (%)
EMSE	387	171	28	27 (96)
IST	540	146	31	26 (84)
JSS	947	181	24	16 (67)
TOSEM	103	18	8	5 (63)
TSE	288	55	11	9 (82)
Total	2265	571	102	83 (81)

Empirical Software Engineering (EMSE), *Information and Software Technology* (IST), *ACM Transactions on Software Engineering and Methodology* (TOSEM), and *Journal of Systems and Software* (JSS). We found the proportion of research articles that included the term "construct validity" for each year in the period 2000 to 2019. For this investigation, we used ACM Digital Library for TOSEM, IEEE Explore for TSE, Springer's internal search engine for EMSE, and Elsevier's internal search engine for IST and JSS. Further details on how the search was conducted can be found in the supplemental materials at <http://...>¹⁰

3.1.3 CV in Software Engineering Experiments

In our academic roles as both reviewers and editors, we have noticed that most discussions on CV appear in articles that report human-centric experiments, which focus on "software engineering methods, techniques and procedures that depend on human expertise" [25]. The concepts studied in such experiments tend to be at a level of higher abstraction (e.g., usability, maintainability, competence, understanding, difficulty) than those studied in experiments that do not involve humans (e.g., time, defect). Thus, the challenge of operationalization is generally greater in human-centric experiments.

We identified the human-centric experiments published in the five journals listed in Section 3.1.2 for the period 2015 to 2019¹¹ using the procedure described in Appendix A. As shown in Table 1, the five journals published 2265 research articles in the given period. Our search script yielded 571 articles. After a manual exclusion process, we ended up with 102 articles that reported human-centric experiments, of which 83 discuss CV.

3.2 Results

3.2.1 CV in the General Software Engineering Literature

The use of the term "construct validity" in the general software engineering literature has greatly increased over the past two decades. Figure 6 shows an increase from 0.4 percent in 2000 to 3.1 percent in 2019; that is, seven times more in 2019. Most of the increase occurred in the last decade. To compare the awareness of CV in software engineering with

10. To editor/reviewers: to be added.

11. In TOSEM, Volume 24 covers parts of 2014 and parts of 2015, Volume 25 parts of 2015 and parts of 2016, etc. The five-year period for the analysis of TOSEM includes Volumes 24 to 28.

that of other computing disciplines, we conducted searches on “information systems” and “hardware” that were similar to those for “software engineering”; see Figure 6. The use of the term “construct validity” has also increased substantially in recent years in the disciplines of information systems and hardware. In that period, a slightly higher proportion of documents with CV were published in information systems than in software engineering, which in turn, has had a higher proportion than hardware. This ordering of disciplines is plausible because it reflects the extent to which human aspects are studied. Human-centric studies tend to involve concepts, such as performance, ability, skill, and motivation of individuals and teams. Naturally, software engineering’s focus on humans is somewhere between information systems and hardware.

The four types of validity described in [20] – construct validity, external validity, internal validity, and conclusion validity – are commonly discussed under “threats to validity” in papers in empirical software engineering. Figure 7 shows that in software engineering, “external validity” and “internal validity” consistently appear more frequently than “construct validity”, which in turn appears much more than “conclusion validity”. All the validity types have increased substantially since the year 2000.

3.2.2 CV in Specific Software Engineering Journals

For the five specific software engineering journals, there is a similar pattern of growth (Figure 8) as in the general software engineering literature. The average proportion of the use of the CV term increased from 4 percent in 2000 to 44 percent in 2019. The variation for 2019 is from 36 percent for TOSEM and JSS to 53 and 56 percent for IST and EMSE, respectively. It is plausible that EMSE has a higher proportion than the other journals because it is tailored to report empirical studies, which typically involve measurements.

3.2.3 CV in Experiments

Of the 102 published experiments, the 83 articles that discuss CV constitute 81 percent; see Table 1. EMSE is on top among the journals, with a CV discussion in 96 percent of the experiment articles. In an earlier study, we investigated human-centric experiments published in the period from 1993 to 2002 [26]. In the sample of 72 experiment articles that were published in the same five journals as examined here, only six discussed CV, that is, 8 percent. The proportion of human-centric experiments published in these journals doubled from the period 1993–2002 to the period 2015–2019, from 2.8 to 5.7 percent. Neto and Conte [27] reported a study of experiments published in nine journals (including the five just mentioned) and four conferences in the period 2003–2011. In their sample of 46 papers, 10 discussed CV (22 percent).

4 CONSTRUCT VALIDITY DISCUSSIONS IN SOFTWARE ENGINEERING

This section reports how the 83 experiment articles identified in Section 3.2.3 define CV and discuss threats to CV.

Fig. 6. Percentage of documents in Google Scholar that contain “construct validity” in three subdisciplines.

Fig. 7. Percentage of software engineering documents that contain “external validity”, “internal validity”, “construct validity”, and “conclusion validity”, respectively.

Fig. 8. Percentage of articles that contain “construct validity” in five journals. Weighted/unweighted average in dashed black/grey.

TABLE 2
Definitions or explanations of CV

Definition category	#	% of total	% of def
Relationship between theory and observation	15	18	42
Relationship between experiment goals and results	6	7	17
Measure of intention	4	5	11
Represents the concepts of study	3	4	8
Miscellaneous definitions	8	10	22
No definition	47	56	
Total	83	100	100

4.1 Analysis Method

The analysis of each article proceeded as follows. Both authors read the article independently to obtain an overview of the reported experiment. Along the way, we extracted text related to CV, which was primarily found under threats to validity but occasionally also in sections on experiment design or research method. In case we had missed CV-related text through our reading, we searched the article automatically for the term “construct” and read further to see whether it was related to CV.

We then worked alongside on the extracted CV text. We categorized how CV was defined and how threats to CV were described according to the three predefined categories of threats to CV given in Section 2.3. To ensure that we did not miss relevant information, we searched the whole article for terms such as “underrepresentation”, “bias”, “mono-operation” and “mono-method”, “operationalization”, and “indicator”.

The sentence(s) or sections that could not be placed in the predefined categories were encoded into new categories. One category that emerged from the texts is “imported constructs”; see Section 4.3.4. Another group of categories was seven types of threats to CV that Cook and Campbell defined in 1979 [11]: see Section 4.3.5. Finally, a large group of issues that we consider outside CV are described in Section 4.3.6.

In most cases, we quickly agreed on the encoding. In a few cases, however, interpreting the meaning of the CV text was so difficult for both of us that we decided to exclude the text from the reported analysis.

4.2 Results: Definition of CV

This section discusses the definitions of CV found in the analyzed articles; see Table 2.

The most common definition of CV is that it is concerned with the relationship between theory and observation (Table 2, row 1) [A6], [A9], [A15], [A33], [A40], [A49], [A53], [A56], [A60], [A61], [A63], [A65], [A66], [A81], [A82]. Seven of those 15 articles cite the textbook by Wohlin et al. [6], [28], which provides the statement, “Construct validity ... is concerned with the relation between theory and observation” [6, p. 103]. However, apart from [A81], none of the articles explains what “theory” and “observation” mean in the context of CV.

If the articles had referred to or developed named scientific theories, the “relation between theory and observation” could have meant the relationship between the concepts

of such theories and the variables that contain observed data in the experiments. Such an approach would have been consistent with the notions of “theoretical vocabulary” and “observational vocabulary” used by, e.g., Suppe [29] in his discussion of theoretical and observational terms in scientific theories. Theories are formulated at the conceptual level, while they are tested in hypotheses at the operational level. Theories include validated constructs. However, in the articles with this definition, there are no explicit references to theories that include the constructs of the reported experiments. And vice versa: those articles that use named theories, for example, the Technology Acceptance Model (TAM) [30], which is used by [A1], [A10], [A35], [A76], do not adhere to this definition of CV.

The authors of the articles with this definition may not have intended to use “theory” in the meaning of scientific theories [16], [31] but only as something that resides at the conceptual (abstract) level and in contrast to the operational (concrete) level, as shown in Figure 4. A similar figure, which distinguishes between the levels of “theory” and “observation”, is shown in Trochim [32] and adapted in [6]. Nevertheless, none of the articles is explicit about what is at the conceptual (theory) level, what is at the observational level, and what the relationship is. Thus, the articles themselves do not seem to apply the definition.

Six articles [A2], [A13], [A14], [A71], [A72], [A73] define CV as concerning the relationship between the goals of an experiment and its results, or the cause-effect investigated in the experiment (Table 2, row 2). However, CV does not relate to cause and effect per se, which represents a horizontal relationship in Fig. 4, whereas CV represents a vertical relationship.

Four articles [A7], [A51], [A67], [A75] define CV as the extent to which the variables (or measures) measure what they intend (or claim) to measure (Table 2, row 3). One article [A67] states that CV may also be the extent to which the measures of the experiment test what they claim to test, which is consistent with a definition of test validity: “the extent to which a test (such as a chemical, physical, or scholastic test) accurately measures what it is supposed to measure” (Wikipedia).

At an overall level, it is correct that CV is about the degree to which something measures what it “intends”, “claims”, or “is supposed” to measure. However, a problem with such a definition is that it is difficult to establish and describe accurately what the intention or claim is. None of the articles is explicit about what that intention is. Consequently, judging how well the variables span the (intention of) the concept is problematic.

Defining CV as to how well the variables measure what they intend to measure has support in the classical CV literature [33], [34]. However, more recent literature questions the usefulness of such a definition: “It does seem that if one knows exactly what one intends to measure, then one will probably know how to measure it, and little if any validation research will be necessary. If this is correct, then the problem of validation research is not that it is difficult to find out what is measured; the problem is that it is difficult to find out what one intends to measure” [35, p. 1067].

Furthermore, an intention is subjective, which means that people will differ on the interpretation of the intention

of a measure. Even if people agree (for example, if measured by internal consistency tests like Cronbach's alpha), a subjective fundament makes it challenging to evaluate the operationalizations. If a concept is vaguely and subjectively defined through an intention or claim, a possible consequence is that any set of (arbitrary chosen) indicators then uniquely determine the concept. Evaluating CV would then be meaningless; see Section 2.3.3. An example is when intelligence is defined as what is measured by an IQ test, which is an example used by Sakhnini et al. [A67]. They discuss the challenge of operationalizing the concept of creativity: "The shakiest measure used in the experiment is the Williams test of individual creativity. For any psychometric test, such as the Williams test and the standard IQ tests, there is always the question of whether the test measures what its designers say it measures".

Examples in software engineering are when researchers equate software complexity with McCabe's cyclomatic complexity metric [36] and when they equate software maintainability with the maintainability index [37]. Software complexity and maintainability are certainly more comprehensive concepts than can be fully represented by these metrics [38].

Three articles [A11], [A35], [A39] define CV as the extent to which variables (or measures) represent the concepts under study (Table 2, row 4), which is consistent with our definition in Section 2.

Other explanations are that CV is about the construction or design of an experiment, whether the experiment setting reflects the research objective or the method used to evaluate the outcome of the tasks of an experiment (Table 2, row 5).¹²

4.3 Results: Threats to CV

This section reports what the analyzed articles discuss as threats to CV. The first three subsections discuss the core threats to CV that we introduced in Section 2.3. Table 3 summarizes the findings related to these subsections. The remaining subsections report on import of constructs, peripheral threats to CV, and other kinds of validity than CV.

4.3.1 Inadequate Definition of Concept

Ten articles discuss threats related to inadequate definitions of concepts, which we categorize as follows.

Different definitions. If the definition of a concept under investigation is specific to a research context, the findings may be less relevant to an industry context that uses another definition [A8]. Furthermore, different definitions of core concepts may lead to results that are not comparable across experiments [A37].

Redefinition. Lenberg et al. [A41] wish to form a psychological concept that relates to the understanding of a subject that comprises knowledge and experience. In the absence of a suitable new term, they redefine knowledge to mean the combination of knowledge and experience. They acknowledge that such a redefinition is questionable.

Imported definition. A few articles refer to definitions provided by frameworks [A73] or prior studies [A18], [A28]

to avoid or mitigate the threat that their concepts are inadequately defined.

Definition by examples. Two articles discuss how a concept is formed from concrete examples. Arnaoudova et al. [A6] discuss the risk that developers' perception of linguistic antipattern is bound to particular examples (instances) instead of the *concept* of linguistic antipattern. Bernárdez et al. [A9] investigate the effect of mindfulness in the performance of conceptual modeling. They state that to avoid "inadequate pre-operational explication of construct", they developed reference "conceptual models", which served as examples of the concept of "conceptual model".

Disagreement about extension. Even if a concept has a commonly agreed definition, its extension may be in question. Kosar et al. [A38] describe the threat of people disagreeing about whether a given tool is a build tool, editor, find and replace tool, or some other tool. Palomba et al. [A53] discuss the threat of tools for detecting code smells identifying smells that are not perceived as smells by developers.

4.3.2 Construct Underrepresentation

Twenty-eight of the articles address threats to construct underrepresentation, which we categorize as follows.

Mono-operation. Twenty-two articles consider the mono-operation threat, where most of them discuss it for moderator concepts. Twelve articles use the term "mono-operation" explicitly; the other articles describe this threat using other terms such as "single" or by explaining that a concept was measured using only one variable. Five articles use the term "mono-method bias" to describe mono-operation of measures; they do not discuss methods at all. Their view is consistent with that of Trochim [12], who confines the use of mono-operationalization to the side of cause and mono-method to the side of effect. However, such a view confuses two separate issues. One concerns the number of operationalizations, the other the number of methods used for data collection, independently of whether the methods apply to the cause or effect side [11], [20].

Multiple indicators. Seven articles report underrepresentation of a concept even though multiple indicators are used: six indicators are used in [A64], four in [A57], three in [A81] and [A41], two in [A35] and [A67], and an unspecified number of indicators are used in [A51].

No operation. An extreme case of underrepresentation is when no indicator represents the concept. Krein et al. [A39] did use an indicator of motivation in the analysis but acknowledge it as a threat that they did not formally operationalize or measure, but instead used "ad hoc, largely qualitative data" to represent the concept. Consequently, they state, "our conclusions about motivation should not be accepted as established fact, but rather should be considered as justification for future work."

4.3.3 Construct-Representation Bias

Twenty-two of the articles address threats to construct-representation bias, which we divide into three categories.

Mono-method. Three articles address threats related to mono-methods bias. Hassan et al. [A32] use two methods for measuring the completeness of definitions of software project scope: calculation of a software project index and expert judgement. Sharif et al. [A74] use a developer's gaze

12. To editor/reviewers: We have deliberately not referenced those who provided these explanations because they have misunderstood what CV is about.

TABLE 3
Threats to CV that are present, mitigated, or not present in the experiments

Threats to CV	Role of Construct	Threat	Mitigated	No threat	#
Inadequate definition of concept	Independent	Different definitions [A37] Redefinition [A41]	Definition by examples [A6]		3
	Moderator	Disagreement about extension [A38]	Disagreement about extension [A53]	Imported definition [A28]	3
	Dependent		Different definitions [A8] Imported definition [A73]	Imported definition [A18] Definition by examples [A9]	4
Underrepresentation	Independent	Mono-operation [A54], [A55], [A58] Multiple indicators [A41]	Mono-operation [A9], [A16]		6
	Moderator	Mono-operation [A8], [A28], [A65], [A69] No operation [A39]	Mono-operation [A2], [A12], [A17], [A36], [A74]	Mono-operation [A11], [A39], [A78]	13
	Dependent	Mono-operation [A28], [A35], [A60], [A65], [A78] Multiple indicators [A35], [A57], [A64], [A67], [A81]	Mono-operation [A9], [A29], [A32], [A54], [A63] Multiple indicators [A51]	Mono-operation [A54]	17
Construct-Representation Bias	Independent				0
	Moderator	Subjectivity [A40]	Subjectivity [A5]		2
	Dependent	Subjectivity [A77], [A81]	Mono-method [A18] Subjectivity [A33], [A49], [A53], [A56], [A65], [A79] Indicators outside scope [A30]	Mono-method [A32], [A74] Subjectivity [A1], [A4], [A7], [A15], [A18], [A43], [A75], [A76], [A78]	21
Total		25	26	18	69

as the only method for data collection but suggest using a tool that tracks how a developer interacts with an IDE as an additional method. In the experiment of Czepa et al. [A18], keeping time records was the personal responsibility of the participant. They suggest using an online tool as an additional measurement method.

Subjectivity. Data based on a person’s opinions, feelings, views, or desires is subjective. If the purpose of a study is to investigate a subjective phenomenon, subjectivity does not lead to construct-representation bias. For example, the goal of the experiment reported in Alves et al. [A4] was “to see the practical impact that each inspection strategy has on developers’ opinions.” However, 18 articles discuss subjectivity as a source of bias, that is, as an unintended side effect. The articles distinguish between *evaluator bias* (called “rater bias” in educational research and other disciplines) and *participant bias*. Evaluator bias was discussed in twelve articles and concerned how evaluators rated participants’ performance on the dependent variables, which included the number of correctly fixed faults [A15], usefulness of business value modeling methods [A76], quality of models of stakeholders’ prioritization in requirements specification [A1], architectural design decisions [A43], release notes [A49], maintenance tasks [A56], comments on scenarios [A65], conceptual models [A75], use case specifications [A77], system functionality [A78], architectural changes [A79], and requirement documents [A81]. Among reasons for reporting no threat for evaluator bias were the use of automated techniques [A78], automatically extracted

objective variables [A43], and high inter-rater reliability between two experts [A75]. Participant bias was reported in nine articles. This bias was discussed for the self-reported moderator variables skill [A40] and knowledge [A5]. The dependent variables discussed were developers’ opinion on an inspection strategy [A4], usability of visualization and interaction techniques [A7], understandability of graphical and textual constraint representations [A18], usefulness of feature diagrams and textual requirements [A33], quality of release notes [A49], code smell identification [A53], and understanding of requirement documents [A81].

Indicators outside scope. An extreme case of construction-representation bias is when indicators are outside the scope of the concept; see Figure 3 (left). Hannebauer et al. [A30] report an example of this kind of threat: “the tasks might not measure program comprehension but rather a completely different cognitive process. We tried to avoid this pitfall commonly known as construct validity by using different types of tasks, so it becomes more unlikely that all tasks measure something different than program comprehension.”

4.3.4 Import of Constructs

Reuse of formerly defined and validated constructs in a study increases quality and saves effort, similarly to the reuse of software components. Sixteen of the articles describe constructs imported from various sources.

Models, theories, and frameworks. Well-founded models and theories include validated constructs. Five articles [A1],

[A7], [A10], [A35], [A76] import the constructs *perceived usefulness* and *perceived ease of use* from the Technology Acceptance Model (TAM) [30] to evaluate technologies. Two articles [A23], [A31] use a framework for evaluating comprehensibility of conceptual models described in [39]. This framework is, in turn, based on several theories.

Psychometric tests. The fields of psychology and education have developed many complex constructs to describe skills, knowledge, and learning achievements, which are also used in software engineering. Baum et al. [A8] reuse an instrument for measuring working memory capacity.

Questionnaires. A construct may be measured using questionnaires, where the responses to the questions (items) are indicators. Caivano et al. [A13] reuse the System Usability Scale questionnaire [40] and the Net Promoter Score questionnaire [41], which have been used and validated over years and across disciplines. Four other articles [A8], [A40], [A47], [A48] reuse questionnaires used in other studies but do not cite validation studies of those questionnaires.

Other studies. The models, theories, instruments, etc. described above have been the subjects of comprehensive validation studies where the sole purpose is to understand and validate the constructs of interest. The investigated articles referred to above cite these validation studies. In contrast, some of the articles [A1], [A18], [A25] reuse measures and variables of constructs used by others, but without citing studies that have explicitly attempted to validate those constructs. Thus, the threats to the involved constructs are unknown.

4.3.5 Peripheral Threats to CV

Sections 4.3.1–4.3.3 discussed the three core types of threats to CV. They are also the first three of ten types defined by Cook and Campbell [11]. The remaining seven types are listed in Table 4. Almost half of the articles discuss at least one of those types. All of the articles with citations regarding these types cite Wohlin et al. [28], [6], who brought forward the CV types introduced by Cook and Campbell [11].

These seven types describe aspects of validity, but it is questionable whether they should primarily be categorized as belonging to CV or to internal, external, or statistical conclusion validity. Cook and Campbell themselves, together with Shadish, removed the first four types in Table 4 from their updated list of types of threats to CV in 2002 [20, p. 73]. The fifth type, “evaluation apprehension”, was subsumed in a broader CV type called “reactivity to the experimental situation” in [20]. Furthermore, four types that were categorized as internal validity in [11], were later categorized as CV in [20]. Nevertheless, the articles that report on these four types, which are “resentful demoralization” [A9], [A18], [A32], [A37], [A54], [A63], [A69], [A80], “compensatory equalization” [A9], [A32], [A69], “compensatory rivalry” [A9], [A18], [A32], [A37], [A54], [A63], [A69], [A78], and “treatment diffusion” [A9], [A18], [A32], [A35], [A37], [A59], [A61], [A72], [A73], [A82] categorize them as internal validity. The change of categories over the years by the same scholars illustrates the challenges of understanding and conceptualizing CV. The place to which things belong is a matter of perspective and is subject to argumentation. In the subsections below, we present arguments for why

TABLE 4
Peripheral Named CV Threats

Named threat	Threat	Mitigated Threat	No Threat	#
Hypothesis guessing		[A6], [A8], [A12], [A22], [A43], [A54], [A55], [A59], [A63], [A74]	[A1], [A9], [A21], [A28], [A32], [A34], [A69], [A71], [A77]	19
Interaction of different treatments	[A54], [A61]	[A32], [A72], [A73], [A80], [A83]	[A9]	8
Interaction of testing and treatment	[A28]		[A9], [A54], [A69]	4
Restricted Generalizability Across Constructs	[A28], [A32]		[A9], [A55]	4
Evaluation apprehension	[A19]	[A31], [A54], [A61], [A62], [A63], [A73]	[A1], [A9], [A16], [A17], [A23], [A24], [A25], [A28], [A32], [A72]	17
Experimenter expectancies	[A32], [A51], [A55]	[A1], [A56], [A77], [A80]	[A21], [A40], [A61], [A68], [A71], [A72], [A73], [A75]	15
Confounding constructs and levels of constructs	[A61], [A73]	[A52], [A72]	[A9]	5
Total	11	27	34	72

the types in Table 4 should not be primarily categorized as threats to CV.

Independently of category, the types in Table 4 may be considered peripheral in software engineering also because of the frequency of no threats versus (real) threats in the investigated articles. The types in Table 4 have three times as many “no threats” as “threats” (34 versus 11), whereas for the core CV types in Table 3, there are fewer “no threats” than “threats” (18 versus 25).

Hypothesis guessing. If the participants try to guess the purpose of the experiment, they may change their behavior to meet the purpose. Hypothesis guessing may be confounded with the treatment, and is thus better categorized as a threat to internal validity rather than to CV. All of the 16 articles that mention this kind of threat consider it as mitigated or not present, indicating that it is viewed as of little importance in software engineering.

Interaction of different treatments. If several treatments in a study are present, “we would not be able to unconfound the effects of the treatment from the effects of the context of several treatments” [11, p. 68]. The solution to the problem is either to give only one treatment or to conduct separate analyses of the effect of the different treatments. Because the primary concern of this threat is related to cause and effect, the interaction of different treatments is also better categorized as a threat to internal validity than to CV.

Interaction of testing and treatment. A pretest in an experiment may “condition the reception of the experimental stimulus” [11]. A threat to validity is then whether the treatment will have the same effect on people who did not conduct the pretest; that is, can the results of the experiment be generalized to a non-pretested population? Generalizations from populations that were studied to those that were not studied is an issue of external validity [20].

Restricted generalizability across constructs. Cook and Campbell [11] state, “it is useful to explore . . . how a treatment might influence constructs other than those that first come to mind in the original formulation of the research questions.” The treatment may have unintended negative side effects. One should be careful about generalizing the outcome of a treatment to other dependent constructs than those specified, but this challenge is more related to general experimental design than to the operationalization of a particular concept, which is the core of CV.

Evaluation apprehension. Participants in an experiment may be worried that performance that may be poor might become visible to others, which is termed evaluation apprehension [42]. Dalpiaz et al. [A19] give an example: “The tagging activity was not graded, but this is still part of a course assignment, and this may have influenced the performance of some participants.” A threat is that the effect of such an evaluation is confounded with the effect of the treatment, and is thus better categorized as concerning internal validity rather than CV.

Experimenter expectancies. The “experimenter expectancies” effect [11], also referred to as the observer-expectancy effect [43], is that the experimenter intentionally or unintentionally influences the behavior of the participants in a study. This kind of experimenter bias may then distort the effect of the treatment. We thus consider it as a threat to internal validity. To deal with this effect, a double-blind study is often designed, so that neither the experimenter nor the participant knows whether the participant is in the treatment or the control group.

Confounding constructs and levels of constructs. The amount, magnitude, or quantity of a construct may be divided into levels. The use of different levels may affect the inferences from a study. Trochim [12] gives an example from medicine: A certain dosage of a drug may have no effect against cancer, but a different level of the dosage (increase or decrease) may have an effect. Examples from the articles include levels of the treatment UML diagram (low, high) [A24], code comment (none, bad, good) [A11], domain knowledge awareness (ignorant, aware) [A5], and mindfulness (number, duration of mindfulness training sessions) [A9]. None of the analyzed articles that discuss this threat [A9], [A52], [A61], [A72], [A73] consider levels of treatments. They only discuss characteristics of the experiment participants as moderators, such as ability, expertise, and experience. Variations in levels of constructs may create problems related to cause and effect (internal validity), generalizations from what was sampled in a study to other populations of people, tasks, settings, etc. (external validity), and how concepts are represented by indicators (CV). However, the term “confounding” is used in the combined terms “confounding factor” and “confounding variable”, and is strongly connected to the terms “confounder” or “con-

found”, which are all mostly used in descriptions of threats to internal validity in the general literature; see Section 4.3.6. Accordingly, three of the articles investigated [A61], [A72], [A73] describe problems of internal validity when they discuss this type of validity. One article [A52] describes problems of external validity. The remaining article [A9] just stated that this type was not relevant for the reported experiment. We consider the type “confounding constructs and levels of constructs” as unnecessary because it is already included in the three core threats to CV, as described in Section 2.

4.3.6 Quality Aspects outside CV

This section describes aspects of the quality of a study that the articles include in their section on threats to CV but that are not part of CV, according to how we describe the concept in Section 2. We recommend that these aspects are discussed under threats to internal, external, or statistical conclusion validity, or as general limitations of a study.

Internal validity. The extent to which an observed covariation between A (treatment) and B (outcome) reflects a causal relationship from A to B is referred to as internal validity [20, p. 53]. Threats to internal validity are factors that affect (threaten) the inferences about such a causal relationship. Confounding is one of several threats to internal validity. A confounder is a factor that is not included in an experiment design but still affects both the cause and effect concepts. Thus, when confounders are present, spurious relationships between the treatment and outcome will be obtained, which in turn may lead to mistakes in the inference about cause and effect. In contrast, a moderator or mediator (Section 2.4) is explicitly modeled in the experiment design. The articles discuss threats to validity caused by inequality between treatment groups with respect to a range of confounders:

- Participants’ general expertise [A77], skill [A77], knowledge [A82], experience [A70], [A82], ability [A10], and capability [A21].
- Task difficulty [A14] and tool support [A38].
- Familiarity or understanding of tools [A16], [A19], [A36], questions [A10], [A40], scenarios [A77], and application domain [A20], [A37].

In our opinion, threats related to an experiment as it proceeds should also be considered as threats to internal validity, not to CV. Examples are ordering effects [A21], sharing of solutions [A39], collaboration among participants [A21], and learning effects [A4], [A23]. Most of the articles do not discuss learning effects as part of threats to CV but more appropriately as part of threats to internal validity [A7], [A11], [A17], [A18], [A27], [A34], [A44], [A50], [A51], [A61], [A70], [A73], [A74], [A78], [A82], [A83] or experiment design [A15], [A21], [A22], [A24], [A25], [A26], [A35], [A39], [A40], [A47], [A48], [A54], [A56], [A57], [A58], [A66].

External validity. Whether causal inferences from a particular study generalize to elements that were not studied is referred to as external validity [20]. The types of elements discussed in the articles are participants, tasks, and material used in the experiments. The validity of using students is discussed in [A64], [A83], which is a subject of recurring debate in software engineering [44]. Whether the experimental

tasks used in the experiments are representative for tasks in industry with respect to size and complexity is a topic in [A24], [A26], [A60], [A64]. The discussion about experimental material is also about whether it is representative for material used in industry, which concerns system requirements [A65], code [A56], [A62], code comments [A11], and seeded faults [A2], [A27].

Measurement scales. Four articles [A8], [A21], [A22], [A51] discuss the use of a binary nominal variable versus a more fine-grained scale. Three articles discuss the use of an ordinal scale with respect to whether a more fine-grained scale would be suitable [A65], whether a five-point ordinal scale can be treated as an interval scale [A39], and whether an ordinal scale with three levels is pseudo, i.e., a binary scale, when one level is hardly used [A70]. One article [A7] states that the limited number of options of a Likert scale prevents the accurate measurement of opinions, while another one [A18] justifies using a 5-point Likert scale instead of a scale with 7 or 11 points. Two other articles [A6], [A33] simply state that they use a Likert scale. Core literature and textbooks describe measurement scales as separate from CV [17], [20], [45]. We share the view of Kaner and Bond [5]: “Both in the historical development and logical structure of scientific knowledge, the formulation of a theoretical concept or construct ... precedes the development of measurement procedures and scales.” Measurements scales and statistical aspects such as ceiling/floor effects, discretization of continuous variables, measurement reliability, and heterogeneity of units may be better discussed under statistical conclusion validity, as recommended by Shadish et al. [20].

Fundamental study requirements. Empirical research assumes willing participants who follow the procedure of a study. The articles discuss limitations of a study as follows:

- Participants receive task instructions that have unclear wording [A81] or are otherwise confusing [A50], [A52]. One reason may be unsatisfactory translation from the original language of the instructions to the native language of the participants [A25].
- Participants misunderstand task descriptions because they are written in another language than the native language of the participants [A8], [A19], [A43], [A73].
- Experimental material is of unsatisfactory quality because of lack of testing [A8], [A31], [A75].
- Participants are not allocated sufficient time to finish the study [A81].
- Participants do not use the treatments as specified [A25], [A42], [A43], [A75].
- Participants cheat [A17], [A78].

5 DISCUSSION AND GUIDELINES

This section proposes a set of guidelines and discusses limitations and related work.

5.1 Guidelines

The review showed discrepancies in how CV is defined (theory-observation, goals-results, intentions, etc.). A lack of consensus in the definition may not be a concern if the

articles otherwise share an understanding of how to address CV in practice. However, the articles vary substantially regarding the topics included in the discussion of CV and their relevance to CV, and the extent to which they discuss real versus mitigated or non-present threats to CV. To achieve a more shared understanding and reporting of CV in software engineering, we propose a set of guidelines that are based on a combination of core CV literature and the findings in the analyzed articles; see Table 5. Although we analyzed only experiments, we consider the guidelines to be applicable to any empirical study, except where otherwise specified.

5.1.1 Create model of constructs

For some of the articles, it is difficult to obtain an overview of the constructs and how they are used. We recommend creating a model of the constructs upfront in the form of a figure, as illustrated in Fig. 4. Such a model will help ensure that the constructs are referred to consistently throughout an article, which, in turn, will alleviate the confusion that occurs when terms differ (such as synonyms used that may not be universally defined) in the abstract, introduction, method, results, and discussion sections, even though they are intended to refer to the same constructs.

Another benefit of such a model is that other researchers can more easily judge whether a paper is relevant to their research. Although most constructs are on the dependent side in the analyzed articles, the model should also include independent and moderator/mediator constructs when reporting experiments.

5.1.2 Import constructs

One-fifth of the articles studied import constructs. The import of explicitly validated constructs saves effort and improves research quality. Meta-studies also benefit from such reuse of constructs because it is easier to identify and synthesize primary studies that use the same concepts and operationalizations. If relevant constructs are not already available in the literature, the research question could be adjusted to reuse formerly validated constructs.

5.1.3 Ensure adequate concept definitions

One-seventh of the articles address threats regarding concept definitions, which include different definitions of the same concept, concept redefinitions, and disagreement about the extension of a concept. To avoid or mitigate such threats, definitions of concepts should be primarily imported from established theories, standards, or frameworks, or from other articles that explicitly and thoroughly define the concepts. If appropriate definitions are not available, new ones must be made. A recommendation is that a definition should be attempted that is consistent with a common perception of the concept or with at least one of the meanings given in contemporary dictionaries or encyclopedias. If a unique definition is required, a reason for a deviation from a common definition should be given. A particular challenge when defining concepts is the *jingle-jangle* fallacy [34], which is the erroneous assumption that two concepts have the same definition (intension) and extension because the concepts have the same name (*jingle*) or that two concepts have different definitions and extensions because

TABLE 5
Guidelines for CV

#	Guideline	Description
G1	Create model of constructs	Create a model of the constructs under study, preferably represented as a figure, that includes the concepts and their indicators. Experiments would typically include independent, moderator/mediator, and dependent constructs. While creating the model, for each construct, do G2 if possible, else do G3–G5.
G2	Import constructs	If available, import already validated constructs from established theories, frameworks, standards, tests, instruments, or questionnaires, or refer to studies that have formerly validated the constructs.
G3	Ensure adequate concept definitions	If available, import definitions of core study concepts from theories, frameworks, standards, or refer to studies that have formerly defined the concepts. If not available, carefully define the concepts.
G4	Avoid construct underrepresentation	Select indicators that span the concept. In general, the more indicators that are non-overlapping, the less underrepresentation.
G5	Avoid construct-representation bias	Avoid systematic bias in the representation of the concept; for example, avoid using only one data collection method (mono-method) or collecting only subjective data.
G6	Avoid excessive interpretation of CV	Focus on the core CV aspects given by G3–G5. Many aspects of study quality are important but do not naturally belong to a discussion of CV.
G7	Report threats and constructs explicitly	When reporting threats to CV in a paper, be explicit about what the threats are and which constructs are subject to the threats.

the concepts have different names (*jangle*). An example of a jangle is given by Dieste et al. [A20], who discuss two different dictionary definitions of *experience*¹³: “(1) skill or knowledge that you get by doing something, and (2) the length of time that you have spent doing something (such as a particular job).” An example of a jangle is if *experience* and *expertise* are both defined by (2).

5.1.4 Avoid construct underrepresentation

One-third of the articles address the threat of construct underrepresentation, primarily that only one indicator is used (mono-operation). In most cases, several indicators are necessary to span the concept sufficiently. However, even if more indicators are used, the concept may still be underrepresented, as noted in seven of the articles; in two of the articles, as many as six indicators were still not considered to be sufficient to avoid underrepresentation. Therefore, we cannot advise any concrete number of indicators, but in general, the broader a concept is, the more indicators are needed. As long as new indicators do not fully overlap existing ones (resulting in redundancy), adding new indicators will reduce underrepresentation.

Within the constraints of a study, it is often difficult to achieve an acceptable representation of a broad concept. A practical approach is then to use a narrower concept, as we have illustrated previously [46]: When the available indicator is “faults per line of code”, it would be better to use the narrow concept of “fault density” rather than the broad concept of “software quality”, which would be severely underrepresented.

5.1.5 Avoid construct-representation bias

One-fourth of the articles address the threat of construct-representation bias. Bias means here that there is a systematic error in the measurements (as opposed to random

error). The most common bias discussed in the articles was subjectivity of evaluators or experiment participants when they rated a variety of quality-related aspects (Section 4.3.3). In software engineering, several quality concepts, such as usability and maintainability, may benefit from indicators where both subjective and objective data are collected. Dybå et al. [47] compared subjective and objective measures of software methodology usage. Generally, to avoid mono-method bias, one should use several methods, that is, method triangulation [9], [48].

5.1.6 Avoid excessive interpretation of CV

The analyzed articles reveal a variety of definitions of CV and an extensive set of what is considered threats to CV. If the software engineering community achieves a common ground on what CV is, how to improve it, and how to evaluate the threats to it, it would be easier to learn from one another and review others’ work and, thus, generally raise the quality of studies in our field. We believe that achieving a common ground would be more obtainable with a narrower interpretation of CV, which is reflected through guidelines G3 to G5 (below). Many of the issues discussed as threats to CV in the articles should, in our view, be considered as threats to other types of validity or as more general limitations of the studies; see Section 4.3.6. In particular, we recommend that the set of threats to CV proposed by Cook and Campbell [11] more than 40 years ago (Table 4) should actually not be regarded as threats to CV. Measurement scales are central in measurements, but we argue that they should be included under statistical conclusion validity.

5.1.7 Report threats and constructs explicitly

Several of the articles discuss threats to CV where it is unclear what the threat is or which construct is concerned. Sometimes the experimental design or material is described without indicating any threat. Other articles state a problem

13. <http://www.merriam-webster.com/dictionary/experience>. Retrieved 7 Oct 2015.

with a particular threat (e.g., mono-operation) without stating which construct faces that threat. In addition to being explicit about threats and constructs, we also recommend that the authors should primarily describe threats that are present in the study, and to a lesser extent describe non-threats. Following G1 may make it easier to be explicit about which construct is the object of discussion.

5.2 Applying the Guidelines

We illustrate the use of the guidelines by applying them to the construct “CV awareness in software engineering literature” investigated in Section 3.

- G1 Figure 5 shows a model of the construct.
- G2 We found no previously validated theories, frameworks, standards, tests, instruments, or questionnaires from which we could import the awareness construct.
- G3 We found no definition of the concept of “CV awareness in the software engineering literature” either. (Note that a concept may be defined even though a corresponding construct, which also includes the operationalizations, is not defined.) We define “CV awareness in the software engineering literature” as the extent to which the term “construct validity” is used in the software engineering research literature. We could have investigated CV awareness among software engineering researchers more comprehensively using methods such as surveys and interviews.
- G4 We have a mono-operation threat because we use only one indicator. Better would have been to add indicators such as (1) density of the term “construct validity” in the documents; that is, frequency relative to the length of the document, not only a binary indicator; (2) length of discussions of CV in the documents; and (3) quality of the text that concerns CV, cf. our discussion in Section 4.
- G5 Our approach is facing mono-method bias because we use only text analysis to collect data. If we had had the extra indicator (item (3) in G4 above), we could have evaluated the quality of the CV text by interviewing experts who read the text. Moreover, if we had investigated the broader construct of “CV awareness among software engineering researchers” mentioned above, it would have been natural to use both surveys and interviews as methods. The present data collection is objective. If we had included the indicator (3) based on interviews with experts, we would have faced the bias of subjectivity.
- G6 By following the guidelines, we have only discussed the core aspect of CV. Other quality aspects are discussed under limitations; see Section 5.3.
- G7 We explicitly stated which construct we discussed above.

5.3 Limitations

This section discusses limitations of the studies reported in Sections 3 and 4, respectively.

5.3.1 Quantitative Study of Awareness

To identify the CV awareness in the software engineering literature, we first sampled documents from Google Scholar. Software engineering documents were identified by whether they contained the term “software engineering”. Indeed, a document may contain that term even though the document is not part of the software engineering literature. The same is the case for the hardware and information systems literature, which we also investigated. However, given that we investigated thousands of documents, the overall effect of random errors (noise) is likely to be very small. Furthermore, the five journals from which we sampled articles in general and those that reported experiments in particular have relatively high impact factors and publish many human-centric studies. The absolute proportion of documents that include the CV term differs in our three samples, but the increase is evident in all the samples, which strengthens the overall result of a substantial increase in the awareness in the field of software engineering.

5.3.2 Qualitative Study of CV Discussions

Some of the CV analysis reported in Section 4 is objective, for example, the frequency of articles with specific terms (e.g., CV types reported in [11]). However, because the texts are often not explicit regarding the topics we investigate, much analysis required interpretation. Also, our choice of and, emphasis on, the various CV topics in this article is subjective. Our perspective reflects our former work related to validity in general in software engineering [26] and specifically on CV when we constructed and validated an instrument for measuring programming skill [49], [50].

5.4 Related Work

Already in 1980, Curtis [4] imported discussions of CV from the general psychological literature to the software engineering literature. In a framework for evaluating software metrics, Kaner and Bond [5] included one of ten questions as “the CV question”: “What is the relationship of the attribute to the metric value?”. The discussion of CV by Wohlin et al. [28], [6] is primarily based on Cook and Campbell [11], whose work then became better known within software engineering.

Ralph and Tempero [7] proposed a set of guidelines explicitly for assessing CV. We share the philosophical view of scientific realism as the underpinning of CV.¹⁴ Their guideline on generating a nomological network¹⁵ corresponds to our guideline on creating a model of the constructs under study (G1). Furthermore, their guideline on assessing *content validity* (“the degree to which an operationalization

14. We have discussed CV at a practical level to support software engineering researchers in evaluating CV in the studies they undertake. We have deliberately refrained from a more in-depth philosophical discussion. Readers who would be interested in understanding more about CV in the context of the philosophy of science may start reading, for example, Cherryholmes [51], who discusses CV in relation to *deconstruction* (Derrida), *critical theory* (Habermas), and *interpretive analytics* (Foucault).

15. The term “nomological network” was coined by Cronbach and Meehl [1]. It indicates a lawlike network, which is difficult to establish in the human-centric parts of software engineering. The term is never used in the analyzed articles, which signifies that it is an unfamiliar term in software engineering.

encompasses all aspects of a construct”) corresponds to our guideline on avoiding construct underrepresentation (G4). We also agree on their convergent/discriminant approach for CV, which is reflected in our guidelines G3 to G5. Ralph and Tempero include a guideline on assessing *face validity*. Despite the appeal of intuitively reasonable operationalizations, we do not support such a guideline because of the problems acknowledged by Ralph and Tempero themselves: “intuition is subjective and reality is often counterintuitive”. Note that we already included subjectivity as one category of construct-representation bias in Section 4.3.3. Moreover, Messick [17] state: “Face validity, which refers to what a test appears to measure to the untrained eye, is discounted as not really being validity at all in the technical sense”. Ralph and Tempero also provide a guideline on assessing *predictive validity*¹⁶ as part of CV. In our opinion, CV should be restricted to how well a set of indicators represent a single concept, not to how well a construct predicts another construct.

What to subsume under CV has historically been a challenge. From its conception in the 1950s, the notion of CV continued to expand into the 1980s and 1990s. Cronbach stated in 1984 [52, p. 126] that “the profession is coming around to the view that all validation is construct validation”. This broad interpretation of CV also underlies many of the articles we analyzed that were published between 2015 and 2019, where a wide range of topics is included under the CV umbrella. Messick advocated an even broader view in 1989 [17], which included topics such as the social consequences of using measurements in particular ways.

More recently, Borsboom et al. [35] called for a conceptualization of validity that has been “stripped of all excess baggage”. Our guidelines are a step in the direction of stripping the excess baggage of CV in particular.

6 CONCLUSIONS

Awareness of CV in software engineering has increased substantially over the last 20 years. CV is now a topic in four of five experiments published in the top-tier journals that we investigated. Many of the articles provided insightful discussions of CV. However, a large proportion of the articles continue to discuss CV inconsistently, and in some cases, incoherently.

Based on our analysis of 83 reported experiments, we propose a set of guidelines to improve how CV is handled and reported in software engineering. In several of the articles, it was challenging to identify the core constructs of the reported study and the constructs that were the objects of the discussion of threats to CV. Therefore, we advise researchers to create and present a model of the principal constructs in a study, one that includes the concepts, their indicators, and the relationships among the constructs. We also advise researchers to be explicit about which constructs face which threats when discussing threats to CV.

One in five of the articles reviewed reuse formerly defined and validated constructs. To save effort and improve quality, we recommend that, when available, researchers

should import available, relevant constructs from established theories, frameworks, standards, tests, instruments, or questionnaires.

The articles discuss a wide range of topics under the notion of CV. Some topics are clearly outside CV. Other topics may be included if one takes the broad view that “all validation is CV”. However, a concept that is too wide is less useful as a means to establish common ground. We observed that many types of validity classified as CV in the articles are better described as internal, external, or statistical conclusion validity. Therefore, we encourage a minimal approach, which is reflected through the three core CV guidelines brought forward from the classical literature: state how the concept is defined, how indicators span the concept, and how bias is handled.

APPENDIX A: IDENTIFYING HUMAN-CENTRIC EXPERIMENTS

To identify the human-centric experiments published in the five journals listed in Section 3.1.2 for the period 2015–2019, we applied the string as follows to Scopus in January 2020:

```
TITLE-ABS-KEY (
  (experiment OR study OR "empirical evaluation" OR
   "empirical analysis" OR "empirical investigation")
  AND
  ({professionals} OR {humans} OR {subjects} OR
   {students} OR {users} OR {software engineers} OR
   {developers} OR {practitioners} OR {participants} OR
   {groups} OR team)
  AND NOT
  ("qualitative study" OR "case study" OR "field study") )
AND SRCTITLE (
  {acm transactions on software engineering and methodology}
  OR {ieee transactions on software engineering}
  OR {empirical software engineering}
  OR {information and software technology}
  OR {journal of systems and software} )
AND DOCTYPE ( ar )
```

Regarding the meaning of “experiment”, we adhere to the definition given in [20]: “An experiment is a study in which an intervention is deliberately introduced to observe its effects.” However, in software engineering, an experiment is sometimes denoted a “study”, “empirical evaluation”, “empirical analysis”, or “empirical investigation”, which were, therefore, included as search terms in addition to “experiment”. On the other hand, “experiment” is frequently used to denote any empirical study. Thus, we excluded articles with titles, abstracts, or keywords that contained “qualitative studies”, “case studies” and “field studies”.

We specified document type “ar” in the Scopus search string, which indicates an article and thus excluded editorials, commentaries, and letters. We also excluded review articles (indicated by “re” in Scopus), because their CV discussions would typically concern the reviews themselves, not the particular experiments being reviewed.

The search string yielded a set of 571 articles, which were 25 percent of the total number of articles published; see Table 1. In the next step, we manually inspected these articles to identify whether a human-centric experiment was reported. The most common reason for false positives found in the search was that the terms “professionals”, “humans”, “subjects”, “students”, “users”, “software engineers”, “developers”, “practitioners”, “participants”, “groups”, or “team” were referred to in a title or abstract

16. Predictive validity is also referred to as empirical, statistical, and criterion-related validity [45].

even though these people or group of people did not participate as subjects in the reported experiment. This filtering process resulted in 102 articles with experiments, of which 83 discuss CV.

As a check for false negatives (experiments not found by the search string), we manually inspected all the 296 articles published in ESEM, TOSEM, and TSE for the years 2015 and 2016. We found two articles that the automated search missed due to no indication in the abstract that humans were subjects in the experiments even though that was the case. Assuming the same rate of missed experiment articles in the unchecked volumes, we have missed 13 articles in total. (The two experiments we found in the manual search are included in the 83 experiments shown in Table 1.) Nevertheless, it is unlikely that the missing articles would be different from those that we collected regarding discussions on CV.

ACKNOWLEDGMENTS

The authors would like to thank Børge Kile Gjelsten for help with the analysis reported in Section 3, Lenore Hietkamp for editing the article, and Marthe Berntzen, Raluca Florea, Børge Kile Gjelsten, Antonio Martini, and Viktoria Stray for useful comments and discussions.

REFERENCES

- [1] L. J. Cronbach and P. E. Meehl, "Construct validity in psychological tests," *Psychological bulletin*, vol. 52, no. 4, p. 281, 1955.
- [2] B. P. Cohen, *Developing sociological knowledge: Theory and method*. Wadsworth Publishing Company, 1989, vol. 1123.
- [3] A. Einstein, "Remarks concerning the essays brought together in this co-operative volume," P. A. Schilpp, Ed. Open Court, La Salle, 1949, vol. 2, pp. 665–688.
- [4] B. Curtis, "Measurement and experimentation in software engineering," *Proceedings of the IEEE*, vol. 68, no. 9, pp. 1144–1157, 1980.
- [5] C. Kaner and W. P. Bond, "Software engineering metrics: What do they measure and how do we know?" in *Proc. 10th Int'l Software Metrics Symp.* IEEE Computer Society, 2004.
- [6] C. Wohlin, P. Runeson, M. Höst, M. C. Ohlsson, B. Regnell, and A. Wesslén, *Experimentation in software engineering*. Springer Science & Business Media, 2012.
- [7] P. Ralph and E. Tempero, "Construct validity in software engineering research and software metrics," in *Proc. 22nd Int'l Conference on Evaluation and Assessment in Software Engineering*, 2018, pp. 13–23.
- [8] O.-J. Dahl and K. Nygaard, "Simula: an algol-based simulation language," *Communications of the ACM*, vol. 9, no. 9, pp. 671–678, 1966.
- [9] D. T. Campbell and D. W. Fiske, "Convergent and discriminant validation by the multitrait-multimethod matrix," *Psychological bulletin*, vol. 56, no. 2, pp. 81–105, 1959.
- [10] K. G. Jöreskog, "A general approach to confirmatory maximum likelihood factor analysis," *Psychometrika*, vol. 34, no. 2, pp. 183–202, 1969.
- [11] T. D. Cook and D. T. Campbell, "The design and conduct of true experiments and quasi-experiments in field settings," in *Reproduced in part in Research in Organizations: Issues and Controversies*. Goodyear Publishing Company, 1979.
- [12] W. Trochim, J. Donnelly, and K. Arora, *Research methods: The essential knowledge base, 2nd edition*. Cengage Learning, 2016.
- [13] D. I. K. Sjøberg, A. Johnsen, and J. Solberg, "Quantifying the effect of using kanban versus scrum: A case study," *IEEE software*, vol. 29, no. 5, pp. 47–53, 2012.
- [14] M. M. Mark, "Realism, validity, and the experimenting society," *Validity and social experimentation: Donald Campbell's legacy*, vol. 1, pp. 141–166, 2000.
- [15] P. M. Fitts and M. I. Posner, "Human performance," *Cole, Belmont, CA*, vol. 5, pp. 7–16, 1967.
- [16] J. E. Hannay, D. I. K. Sjøberg, and T. Dybå, "A systematic review of theory use in software engineering experiments," *IEEE Transactions on Software Engineering*, vol. 33, no. 2, pp. 87–107, 2007.
- [17] S. Messick, "Validity," *Educational measurement (3rd edn.)*, vol. 1989, pp. 13–103, 1989.
- [18] M. J. Strube and L. C. Newman, "Psychometrics," in *Handbook of psychophysiology*, J. T. Cacioppo, L. G. Tassinari, and G. Berntson, Eds. Cambridge University Press New York, NY, 2007, vol. 3, pp. 789–811.
- [19] P. Panayides, "Coefficient alpha: interpret with caution," *Europe's Journal of Psychology*, vol. 9, no. 4, pp. 687–696, 2013.
- [20] W. R. Shadish, T. D. Cook, and D. T. Campbell, *Experimental and quasi-experimental designs for generalized causal inference*. Boston: Houghton Mifflin, 2002.
- [21] N. Fenton, "Software measurement: A necessary scientific basis," *IEEE Transactions on software engineering*, vol. 20, no. 3, pp. 199–206, 1994.
- [22] R. M. Baron and D. A. Kenny, "The moderator–mediator variable distinction in social psychological research: Conceptual, strategic, and statistical considerations," *Journal of personality and social psychology*, vol. 51, no. 6, p. 1173, 1986.
- [23] G. R. Bergersen and J.-E. Gustafsson, "Programming skill, knowledge, and working memory among professional software developers from an investment theory perspective," *Journal of individual Differences*, vol. 32, no. 4, pp. 201–209, 2011.
- [24] D. Djamel, "Latest impact factors journal list, thomson reuters based on 2019 journal citation reports," 05 2019.
- [25] B. Kitchenham, D. I. K. Sjøberg, T. Dybå, O. P. Brereton, D. Budgen, M. Höst, and P. Runeson, "Trends in the quality of human-centric software engineering experiments—a quasi-experiment," *IEEE Transactions on Software Engineering*, vol. 39, no. 7, pp. 1002–1017, 2012.
- [26] D. I. K. Sjøberg, J. E. Hannay, O. Hansen, V. B. Kampenes, A. Karahasanovic, N.-K. Liborg, and A. C. Rekdal, "A survey of controlled experiments in software engineering," *IEEE Transactions on Software Engineering*, vol. 31, no. 9, pp. 733–753, 2005.
- [27] A. A. Neto and T. Conte, "A conceptual model to address threats to validity in controlled experiments," in *Proceedings of the 17th International Conference on Evaluation and Assessment in Software Engineering*, 2013, pp. 82–85.
- [28] C. Wohlin, P. Runeson, M. Höst, M. C. Ohlsson, B. Regnell, and A. Wesslén, "Experimentation in software engineering: An introduction," *The Kluwer International Series In Software Engineering*, 2000.
- [29] F. Suppe, *The Structure of Scientific Theories*. Urbana, University of Illinois Press, 1974.
- [30] F. D. Davis, "Perceived usefulness, perceived ease of use, and user acceptance of information technology," *MIS quarterly*, pp. 319–340, 1989.
- [31] D. I. K. Sjøberg, T. Dybå, B. C. D. Anda, and J. E. Hannay, "Building theories in software engineering," in *Guide to advanced empirical software engineering*, F. Shull, J. Singer, and D. I. K. Sjøberg, Eds. London: Springer-Verlag, 2008, pp. 312–336.
- [32] W. Trochim and J. P. Donnelly, *The research methods knowledge base., 2nd ed.* Cornell University, 1999.
- [33] R. B. Cattell, "Personality structure and measurement," *British Journal of Psychology. General Section*, vol. 36, no. 3, pp. 159–174, 1946.
- [34] T. Kelly, "Interpretation of educational measurements," *Yonkers, New York: World Book Company*, pp. 62–65, 1927.
- [35] D. Borsboom, G. J. Mellenbergh, and J. Van Heerden, "The concept of validity," *Psychological review*, vol. 111, no. 4, pp. 1061–1071, 2004.
- [36] T. J. McCabe, "A complexity measure," *IEEE Transactions on Software Engineering*, no. 4, pp. 308–320, 1976.
- [37] P. Oman and J. Hagemeister, "Metrics for assessing a software system's maintainability," in *Proceedings Conference on Software Maintenance 1992*. IEEE Computer Society, 1992, pp. 337–338.
- [38] D. I. K. Sjøberg, B. Anda, and A. Mockus, "Questioning software maintenance metrics: a comparative case study," in *Proceedings of the 2012 ACM-IEEE International Symposium on Empirical Software Engineering and Measurement*. IEEE, 2012, pp. 107–110.
- [39] J. Aranda, N. Ernst, J. Horkoff, and S. Easterbrook, "A framework for empirical evaluation of model comprehensibility," in *International Workshop on Modeling in Software Engineering (MISE'07: ICSE Workshop 2007)*. IEEE, 2007, pp. 7–13.
- [40] J. Brooke, "Sus: a 'quick and dirty' usability," in *Usability evaluation in industry*, P. W. Jordan, B. Thomas, I. L. McClelland, and

- B. Weerdmeester, Eds. London: Taylor & Francis Ltd, 1996, pp. 189–194.
- [41] D. B. Grisaffe, “Questions about the ultimate question: conceptual considerations in evaluating reichheld’s net promoter score (nps),” *Journal of Consumer Satisfaction, Dissatisfaction and Complaining Behavior*, vol. 20, pp. 36–53, 2007.
- [42] M. J. Rosenberg, “The conditions and consequences of evaluation apprehension,” in *Artifact in behavioral research*, R. Rosenthal and R. L. Rosnow, Eds. New York : Academic Press, 1969, pp. 279–349.
- [43] R. Rosenthal, *Experimenter effects in behavioral research*. Irvington, 1976.
- [44] R. Feldt, T. Zimmermann, G. R. Bergersen, D. Falesi, A. Jedlitschka, N. Juristo, J. Münch, M. Oivo, P. Runeson, M. Shepperd, D. I. K. Sjøberg, and B. Turhan, “Four commentaries on the use of students and professionals in empirical software engineering experiments,” *Empirical Software Engineering*, vol. 23, no. 6, pp. 3801–3820, 2018.
- [45] J. C. Nunnally and I. H. Bernstein, *Psychometric theory*, 3rd ed. New York: McGraw-Hill, 1994.
- [46] D. I. K. Sjøberg, T. Dybå, and M. Jørgensen, “The future of empirical methods in software engineering research,” in *Future of Software Engineering (FOSE’07)*. IEEE, 2007, pp. 358–378.
- [47] T. Dybå, N. B. Moe, and E. Arisholm, “Measuring software methodology usage: challenges of conceptualization and operationalization,” in *2005 International Symposium on Empirical Software Engineering*, 2005. IEEE, 2005, pp. 447–457.
- [48] M. Q. Patton, *How to use qualitative methods in evaluation*. Sage Publications, 1987.
- [49] G. R. Bergersen, D. I. K. Sjøberg, and T. Dybå, “Construction and validation of an instrument for measuring programming skill,” *IEEE Transactions on Software Engineering*, vol. 40, no. 12, pp. 1163–1184, 2014.
- [50] G. R. Bergersen, “Measuring programming skill – construction and validation of an instrument for evaluating java developers,” Ph.D. dissertation, University of Oslo, 2015.
- [51] C. H. Cherryholmes, “Construct validity and the discourses of research,” *American Journal of Education*, vol. 96, no. 3, pp. 421–457, 1988.
- [52] L. Cronbach, *Essentials of Psychological Testing*. Harper & Row, 1984.
- [A9] B. Bernárdez, A. Durán, J. A. Parejo, and A. Ruiz-Cortés, “An experimental replication on the effect of the practice of mindfulness in conceptual modeling performance,” *The Journal of Systems and Software*, vol. 136, pp. 153–172, 2018.
- [A10] G. Borrego, A. L. Morán, R. R. Palacio, A. Vizcaino, and F. O. García, “Towards a reduction in architectural knowledge vaporization during agile global software development,” *Information and Software Technology*, vol. 112, pp. 68–82, 2019.
- [A11] J. Börstler and B. Paech, “The role of method chains and comments in software readability and comprehension—an experiment,” *IEEE Transactions on Software Engineering*, vol. 42, no. 9, pp. 886–898, 2016.
- [A12] C. Cachero, S. Meliá, and J. M. Hermida, “Impact of model notations on the productivity of domain modelling: An empirical study,” *Information and Software Technology*, vol. 108, pp. 78–87, 2019.
- [A13] D. Caivano, D. Fogli, R. Lanzilotti, A. Piccinno, and F. Cassano, “Supporting end users to control their smart home: design implications from a literature review and an empirical investigation,” *Journal of Systems and Software*, pp. 295–313, 2018.
- [A14] M. Campusano, J. Fabry, and A. Bergel, “Live programming in practice: A controlled experiment on state machines for robotic behaviors,” *Information and Software Technology*, vol. 108, pp. 99–114, 2019.
- [A15] M. Ceccato, A. Marchetto, L. Mariani, C. D. Nguyen, and P. Tonella, “Do automatically generated test cases make debugging easier? an experimental assessment of debugging effectiveness and efficiency,” *ACM Transactions on Software Engineering and Methodology (TOSEM)*, vol. 25, no. 1, pp. 1–38, 2015.
- [A16] J. Cunha, J. P. Fernandes, J. Mendes, and J. Saraiva, “Embedding, evolution, and validation of model-driven spreadsheets,” *IEEE Transactions on software Engineering*, vol. 41, no. 3, pp. 241–263, 2015.
- [A17] J. Cunha, J. Paulo Fernandes, P. Martins, J. Mendes, R. Pereira, and J. Saraiva, “Evaluating refactorings for spreadsheet models,” *The Journal of Systems and Software*, vol. 118, pp. 234–250, 2016.
- [A18] C. Czepa and U. Zdun, “How understandable are pattern-based behavioral constraints for novice software designers?” *ACM Transactions on Software Engineering and Methodology*, vol. 28, no. 2, pp. 1–38, 2019.
- [A19] F. Dalpiaz, I. Van Der Schalk, S. Brinkkemper, F. B. Aydemir, and G. Lucassen, “Detecting terminological ambiguity in user stories: tool and experimentation,” *Information and Software Technology*, vol. 110, pp. 3–16, 2019.
- [A20] O. Dieste, A. M. Aranda, F. Uyaguari, B. Turhan, A. Tosun, D. Fucci, M. Oivo, and N. Juristo, “Empirical evaluation of the effects of experience on code quality and programmer productivity: an exploratory study,” *Empirical Software Engineering*, vol. 22, no. 5, pp. 2457–2542, 2017.
- [A21] M. El-Attar, “Evaluating and empirically improving the visual syntax of use case diagrams,” *The Journal of Systems and Software*, vol. 156, pp. 136–163, 2019.
- [A22] M. El-Attar, H. Luqman, P. Karpati, G. Sindre, and A. L. Opdahl, “Extending the uml statecharts notation to model security aspects,” *IEEE Transactions on Software Engineering*, vol. 41, no. 7, pp. 661–690, 2015.
- [A23] M. Felderer and A. Herrmann, “Manual test case derivation from uml activity diagrams and state machines: A controlled experiment,” *Information and Software Technology*, vol. 61, pp. 1–15, 2015.
- [A24] A. M. Fernández-Sáez, M. Genero, and Chaudron, “Are forward designed or reverse-engineered uml diagrams more helpful for code maintenance?: A family of experiments,” *Information and Software Technology*, vol. 57, pp. 644–663, 2015.
- [A25] A. M. Fernández-Sáez, M. Genero, D. Caivano, and M. R. Chaudron, “Does the level of detail of uml diagrams affect the maintainability of source code?: a family of experiments,” *Empirical Software Engineering*, vol. 21, no. 1, pp. 212–259, 2016.
- [A26] A. Fernandez and A. Bergel, “A domain-specific language to visualize software evolution,” *Information and Software Technology*, vol. 98, pp. 118–130, 2018.
- [A27] G. Fraser, M. Staats, P. McMinn, A. Arcuri, and F. Padberg, “Does automated unit test generation really help software testers? a controlled empirical study,” *ACM Transactions on Software Engineering and Methodology (TOSEM)*, vol. 24, no. 4, pp. 1–49, 2015.
- [A28] D. Fucci, B. Turhan, N. Juristo, O. Dieste, A. Tosun-Misirli, and M. Oivo, “Towards an operationalization of test-driven

ARTICLES STUDIED

- [A1] S. Abrahão, E. Insfran, F. González-Ladrón-de Guevara, M. Fernández-Diego, C. Cano-Genoves, and R. P. de Oliveira, “Assessing the effectiveness of goal-oriented modeling languages: A family of experiments,” *Information and Software Technology*, vol. 116, p. 106171, 2019.
- [A2] W. Afzal, A. N. Ghazi, J. Itkonen, R. Torkar, A. Andrews, and K. Bhatti, “An experiment on the effectiveness and efficiency of exploratory testing,” *Empirical Software Engineering*, vol. 20, no. 3, pp. 844–878, 2015.
- [A3] N. Ahmad, A. Rextin, and U. E. Kulsoom, “Perspectives on usability guidelines for smartphone applications: An empirical investigation and systematic literature review,” *Information and Software Technology*, vol. 94, pp. 130–149, 2018.
- [A4] E. L. Alves, M. Song, T. Massoni, P. D. Machado, and M. Kim, “Refactoring inspection support for manual refactoring edits,” *IEEE Transactions on Software Engineering*, vol. 44, no. 4, pp. 365–383, 2018.
- [A5] A. M. Aranda, O. Dieste, and N. Juristo, “Effect of domain knowledge on elicitation effectiveness: An internally replicated controlled experiment,” *IEEE Transactions on Software Engineering*, vol. 42, no. 5, pp. 427–451, 2016.
- [A6] V. Arnaudova, M. Di Penta, and G. Antoniol, “Linguistic antipatterns: What they are and how developers perceive them,” *Empirical Software Engineering*, vol. 21, no. 1, pp. 104–158, 2016.
- [A7] M. Asadi, S. Soltani, D. Gašević, and M. Hatala, “The effects of visualization and interaction techniques on feature model configuration,” *Empirical Software Engineering*, vol. 21, no. 4, pp. 1706–1743, 2016.
- [A8] T. Baum, K. Schneider, and A. Bacchelli, “Associating working memory capacity and code change ordering with code review performance,” *Empirical Software Engineering*, vol. 24, no. 4, pp. 1762–1798, 2019.

- development skills: An industrial empirical study," *Information and Software Technology*, vol. 68, pp. 82–97, 2015.
- [A29] H. Ghanbari, J. Similä, and J. Markkula, "Utilizing online serious games to facilitate distributed requirements elicitation," *The Journal of Systems and Software*, vol. 109, pp. 32–49, 2015.
- [A30] C. Hannebauer, M. Hesenius, and V. Gruhn, "Does syntax highlighting help programming novices?" *Empirical Software Engineering*, vol. 23, no. 5, pp. 2795–2828, 2018.
- [A31] F. Häser, M. Felderer, and R. Breu, "Is business domain language support beneficial for creating test case specifications: A controlled experiment," *Information and Software Technology*, vol. 79, pp. 52–62, 2016.
- [A32] I. u. Hassan, N. Ahmad, and B. Zuhaira, "Calculating completeness of software project scope definition," *Information and Software Technology*, vol. 94, pp. 208–233, 2018.
- [A33] N. Itzik, I. Reinhardt-Berger, and Y. Wand, "Variability analysis of requirements: Considering behavioral differences and reflecting stakeholders' perspectives," *IEEE Transactions on Software Engineering*, vol. 42, no. 7, pp. 687–706, 2015.
- [A34] K. Jaber, B. Sharif, and C. Liu, "An empirical study on the effect of 3d visualization for project tasks and resources," *The Journal of Systems and Software*, vol. 115, pp. 1–17, 2016.
- [A35] P. Karpati, A. L. Opdahl, and G. Sindre, "Investigating security threats in architectural context: Experimental evaluations of misuse case maps," *The Journal of Systems and Software*, vol. 104, pp. 90–111, 2015.
- [A36] J. King, J. Stallings, M. Riaz, and L. Williams, "To log, or not to log: using heuristics to identify mandatory log events—a controlled experiment," *Empirical Software Engineering*, vol. 22, no. 5, pp. 2684–2717, 2017.
- [A37] S. Kopczyńska, J. Nawrocki, and M. Ochodek, "An empirical study on catalog of non-functional requirement templates: Usefulness and maintenance issues," *Information and Software Technology*, vol. 103, pp. 75–91, 2018.
- [A38] T. Kosar, S. Gaberc, J. C. Carver, and M. Mernik, "Program comprehension of domain-specific and general-purpose languages: replication of a family of experiments using integrated development environments," *Empirical Software Engineering*, vol. 23, no. 5, pp. 2734–2763, 2018.
- [A39] J. L. Krein, L. Prechelt, N. Juristo, A. Nanthaamornphong, J. C. Carver, S. Vegas, C. D. Knutson, K. D. Seppi, and D. L. Eggett, "A multi-site joint replication of a design patterns experiment using moderator variables to generalize across contexts," *IEEE Transactions on Software Engineering*, vol. 42, no. 4, pp. 302–321, 2016.
- [A40] K. Labunets, F. Massacci, F. Paci, S. Marczak, and F. M. de Oliveira, "Model comprehension for security risk assessment: an empirical comparison of tabular vs. graphical representations," *Empirical Software Engineering*, vol. 22, no. 6, pp. 3017–3056, 2017.
- [A41] P. Lenberg, L. G. W. Tengberg, and R. Feldt, "An initial analysis of software engineers' attitudes towards organizational change," *Empirical Software Engineering*, vol. 22, no. 4, pp. 2179–2205, 2017.
- [A42] O. A. Lazzarini Lemos, F. Fagundes Silveira, F. Cutigi Ferrari, and A. Garcia, "The impact of software testing education on code reliability: An empirical assessment," *The Journal of Systems and Software*, vol. 137, pp. 497–511, 2018.
- [A43] I. Lytra, P. Gaubatz, and U. Zdun, "Two controlled experiments on model-based architectural decision making," *Information and Software Technology*, vol. 63, pp. 58–75, 2015.
- [A44] P. Mäder and A. Egyed, "Do developers benefit from requirements traceability when evolving and maintaining a software system?" *Empirical Software Engineering*, vol. 20, no. 2, pp. 413–441, 2015.
- [A45] S. Makady and R. J. Walker, "Debugging and maintaining pragmatically reused test suites," *Information and Software Technology*, vol. 102, pp. 6–29, 2018.
- [A46] B. Marculescu, S. Poulding, R. Feldt, K. Petersen, and R. Torkar, "Tester interactivity makes a difference in search-based software testing: A controlled experiment," *Information and Software Technology*, vol. 78, pp. 66–82, 2016.
- [A47] J. M. Morales, E. Navarro, P. Sánchez, and D. Alonso, "A controlled experiment to evaluate the understandability of kaos and i for modeling teleo-reactive systems," *The Journal of Systems and Software*, vol. 100, pp. 1–14, 2015.
- [A48] —, "A family of experiments to evaluate the understandability of tristar and i for modeling teleo-reactive systems," *The Journal of Systems and Software*, vol. 114, pp. 82–100, 2016.
- [A49] L. Moreno, G. Bavota, M. Di Penta, R. Oliveto, A. Marcus, and G. Canfora, "Arena: An approach for the automated generation of release notes," *IEEE Transactions on Software Engineering*, vol. 43, no. 2, pp. 106–127, 2017.
- [A50] S. Nielebock, D. Krolikowski, J. Krüger, T. Leich, and F. Ortmeier, "Commenting source code: is it worth it for small programming tasks?" *Empirical Software Engineering*, vol. 24, no. 3, pp. 1418–1457, 2019.
- [A51] A. Niknafs and D. Berry, "The impact of domain knowledge on the effectiveness of requirements engineering activities," *Empirical Software Engineering*, vol. 22, no. 1, pp. 80–133, 2017.
- [A52] R. Novais, J. A. Santos, and M. Mendonça, "Experimentally assessing the combination of multiple visualization strategies for software evolution analysis," *The Journal of Systems and Software*, vol. 128, pp. 56–71, 2017.
- [A53] F. Palomba, A. Panichella, A. Zaidman, R. Oliveto, and A. De Lucia, "The scent of a smell: An extensive comparison between textual and structural smells," *IEEE Transactions on Software Engineering*, vol. 44, no. 10, pp. 977–1000, 2018.
- [A54] J. I. Panach, S. Espana, O. Dieste, O. Pastor, and N. Juristo, "In search of evidence for model-driven development claims: An experiment on quality, effort, productivity and satisfaction," *Information and Software Technology*, vol. 62, pp. 164–186, 2015.
- [A55] J. I. Panach, N. Juristo, F. Valverde, and O. Pastor, "A framework to identify primitives that represent usability within model-driven development methods," *Information and Software Technology*, vol. 58, pp. 338–354, 2015.
- [A56] L. Ponzanelli, G. Bavota, M. Di Penta, R. Oliveto, and M. Lanza, "Turning the ide into a self-confident programming assistant," *Empirical Software Engineering*, vol. 21, no. 5, pp. 2190–2231, 2016.
- [A57] A. Przybyłek, "An empirical study on the impact of aspectj on software evolvability," *Empirical software engineering : an international journal*, vol. 23, no. 4, pp. 2018–2050, 2017.
- [A58] I. Reinhardt-Berger, K. Figl, and Ø. Haugen, "Investigating styles in variability modeling: Hierarchical vs. constrained styles," *Information and Software Technology*, vol. 87, pp. 81–102, 2017.
- [A59] M. Riaz, J. King, J. Slankas, L. Williams, F. Massacci, C. Quesada-López, and M. Jenkins, "Identifying the implied: Findings from three differentiated replications on the use of security requirements templates," *Empirical Software Engineering*, vol. 22, no. 4, pp. 2127–2178, 2017.
- [A60] F. Ricca, M. Torchiano, M. Leotta, A. Tiso, G. Guerrini, and G. Reggio, "On the impact of state-based model-driven development on maintainability: a family of experiments using unimod," *Empirical Software Engineering*, vol. 23, no. 3, pp. 1743–1790, 2018.
- [A61] F. Ricca, G. Scanniello, M. Torchiano, G. Reggio, and E. Astesiano, "Assessing the effect of screen mockups on the comprehension of functional requirements," *ACM Transactions on Software Engineering and Methodology (TOSEM)*, vol. 24, no. 1, pp. 1–38, 2014.
- [A62] E. Macedo Rodrigues, F. Moreira de Oliveira, L. Teodoro Costa, M. Bernardino, A. F. Zorzo, S. do Rocio Senger Souza, and R. Saad, "An empirical comparison of model-based and capture and replay approaches for performance testing," *Empirical Software Engineering*, vol. 20, no. 6, pp. 1831–1860, 2015.
- [A63] S. Romano, N. Capece, U. Erra, G. Scanniello, and M. Lanza, "On the use of virtual reality in software visualization: The case of the city metaphor," *Information and Software Technology*, vol. 114, pp. 92–106, 2019.
- [A64] G. Rong, H. Zhang, B. Liu, Q. Shan, and D. Shao, "A replicated experiment for evaluating the effectiveness of pairing practice in psp education," *The Journal of Systems and Software*, vol. 136, pp. 139–152, 2018.
- [A65] L. Sabatucci, M. Ceccato, A. Marchetto, and A. Susi, "Ahab's legs in scenario-based requirements validation: An experiment to study communication mistakes," *The Journal of Systems and Software*, vol. 109, pp. 124–136, 2015.
- [A66] L. Sabatucci, M. Cossentino, and A. Susi, "A goal-oriented approach for representing and using design patterns," *The Journal of Systems and Software*, vol. 110, pp. 136–154, 2015.
- [A67] V. Sakhnini, L. Mich, and D. M. Berry, "Group versus individual use of power-only epmcreate as a creativity enhancement technique for requirements elicitation," *Empirical Software Engineering*, vol. 22, no. 4, pp. 2001–2049, 2017.

- [A68] F. Saleh and M. El-Attar, "A scientific evaluation of the misuse case diagrams visual syntax," *Information and Software Technology*, vol. 66, pp. 73–96, 2015.
- [A69] I. Salman, B. Turhan, and S. Vegas, "A controlled experiment on time pressure and confirmation bias in functional software testing," *Empirical Software Engineering*, vol. 24, no. 4, pp. 1727–1761, 2018.
- [A70] A. Rodrigues Santos, I. do Carmo Machado, E. Santana de Almeida, J. Siegmund, and S. Apel, "Comparing the influence of using feature-oriented programming and conditional compilation on comprehending feature-oriented software," *Empirical Software Engineering*, vol. 24, no. 3, pp. 1226–1258, 2018.
- [A71] B. M. Santos, A. d. S. Landi, D. S. Santibáñez, R. S. Durelli, and V. V. de Camargo, "Evaluating the extension mechanisms of the knowledge discovery metamodel for aspect-oriented modernizations," *The Journal of Systems and Software*, vol. 149, pp. 285–304, 2019.
- [A72] G. Scanniello, A. Marcus, and D. Pascale, "Link analysis algorithms for static concept location: an empirical assessment," *Empirical Software Engineering*, vol. 20, no. 6, pp. 1666–1720, 2015.
- [A73] G. Scanniello, C. Gravino, M. Risi, G. Tortora, and G. Dodero, "Documenting design-pattern instances: A family of experiments on source-code comprehensibility," *ACM Transactions on Software Engineering and Methodology (TOSEM)*, vol. 24, no. 3, pp. 1–35, 2015.
- [A74] B. Sharif, J. Meinken, T. Shaffer, and H. Kagdi, "Eye movements in software traceability link recovery," *Empirical Software Engineering*, vol. 22, no. 3, pp. 1063–1102, 2017.
- [A75] A. P. Sinha and H. Jain, "Reusing business components and objects for modeling business systems: The influence of decomposition characteristics and analyst experience," *The Journal of Systems and Software*, vol. 131, pp. 550–569, 2017.
- [A76] E. Souza, A. Moreira, J. Araújo, S. Abrahão, E. Insfran, and D. S. d. Silveira, "Comparing business value modeling methods: A family of experiments," *Information and Software Technology*, vol. 104, pp. 179–193, 2018.
- [A77] S. Tiwari and A. Gupta, "Investigating comprehension and learnability aspects of use cases for software specification problems," *Information and Software Technology*, vol. 91, pp. 22–43, 2017.
- [A78] A. Tosun, O. Dieste, D. Fucci, S. Vegas, B. Turhan, H. Erdogmus, A. Santos, M. Oivo, K. Toro, J. Jarvinen *et al.*, "An industry experiment on the effects of test-driven development on external quality and productivity," *Empirical Software Engineering*, vol. 22, no. 6, pp. 2763–2805, 2017.
- [A79] S. Tragatschnig, S. Stevanetic, and U. Zdun, "Supporting the evolution of event-driven service-oriented architectures using change patterns," *Information and Software Technology*, vol. 100, pp. 133–146, 2018.
- [A80] M. Trkman, J. Mendling, P. Trkman, and M. Krisper, "Impact of the conceptual model's representation format on identifying and understanding user stories," *Information and Software Technology*, vol. 116, p. 106169, 2019.
- [A81] Y.-C. Tu, E. Tempero, and C. Thomborson, "An experiment on the impact of transparency on the effectiveness of requirements documents," *Empirical Software Engineering*, vol. 21, no. 3, pp. 1035–1066, 2016.
- [A82] W. Wu, A. Serveaux, Y.-G. Guéhéneuc, and G. Antoniol, "The impact of imperfect change rules on framework api evolution identification: an empirical study," *Empirical Software Engineering*, vol. 20, no. 4, pp. 1126–1158, 2015.
- [A83] J. Zubcoff, I. Garrigós, S. Casteleyn, J.-N. Mazón, J.-A. Aguilar, and F. Gomariz-Castillo, "Evaluating different i*-based approaches for selecting functional requirements while balancing and optimizing non-functional requirements: A controlled experiment," *Information and Software Technology*, vol. 106, pp. 68–84, 2019.