Construct Validity in Software Engineering

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

Abstract—Empirical research aims to establish generalizable claims from data. Such claims may involve concepts that 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 misunderstanding of experiment material. The diversity of topics included as threats to construct validity calls for a more minimalist approach. Based on the review, we propose seven guidelines to improve how construct validity is handled and reported in software engineering.

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

1 Introduction

In many sciences, researchers attempt to measure things that are not directly measurable. A set of indicators may then 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 to express how a company performs.

In software engineering, studies typically investigate the effect of using various processes, methods, technologies, or practices for software development [85]. If a software development method is evaluated according to the maintainability of the developed software, a set of indicators must be identified to measure maintainability.

A concept that is not directly measurable but is represented by indicators at the operational level to make it measurable is called a *construct*. The validity of a construct, which is called *construct validity* (CV), is defined by how adequate a concept definition is and how well the indicators represent the concept [19], [21]. The development and use of well-defined concepts are at the core of all sciences, as explained by Einstein [29, p. 674]: "thinking without the positing of categories and of concepts in general would be as impossible as is breathing in a vacuum". To make valid inferences at the conceptual level, the indicators must be independent of each other and span the concept without systematic bias [54].

To support knowledge building, results from single studies should be confirmed in replications and aggregated in secondary studies, such as systematic literature reviews. An unsatisfactory CV may lead to inconclusive and contrary results in a series of replications and in systematic reviews. Kitchenham *et al.* [46, p. 29] state that part of the validity of a systematic review is "the consistency and comparability of the operationalisation of the outcome measures used in the primary studies". Suppose a set of replications or studies in a systematic review evaluate and compare different software development methods regarding the maintainability of the developed software. The results of the primary studies may be a consequence of different definitions and/or operationalizations of maintainability rather than of differences in the effects of the methods being investigated [73].

Our motivation for studying CV is that it not only appears intrinsically difficult to evaluate [62] but is also more difficult to evaluate than other kinds of validity [50, p. 43]. Consequently, there is a need for guidelines for CV.

CV in software engineering is typically addressed in textbooks [68], [85] that cover both quantitative and qualitative studies and in papers on the validity of measurements [23], [41], [62]. However, the state of awareness and practice of CV has not been systematically investigated before. In this article, we first report a quantitative analysis of how the awareness of CV in has changed over time.

We then report a qualitative review of how CV is defined and how its threats are discussed in 83 human-centric experiments published in five high-quality journals. We expected to find a higher frequency of CV discussion in human-centric studies, which tend to involve concepts such as usability, maintainability, understanding, difficulty, skill, competence, and motivation, and in experiments in particular because they require such concepts to be measured to be used in descriptive statistics or statistical hypothesis testing. Note that our focus is on constructs in a study, independently of whether humans are involved in the study. Based on the review, we propose a set of guidelines to improve the understanding and handling 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.

• *The authors are with the Department of Informatics, University of Oslo, 0315 Oslo, Norway. E-mail: {dagsj, gunnab}@ifi.uio.no.*

Manuscript received 4 Mar. 2021; revised 7 May 2022; accepted 16 May 2022. Date of publication 23 May 2022; date of current version 15 Mar. 2023. (Corresponding author: D.I.K. Sjøberg.)
Recommended for acceptance by A. Sarma.
Digital Object Identifier no. 10.1109/TSE.2022.3176725

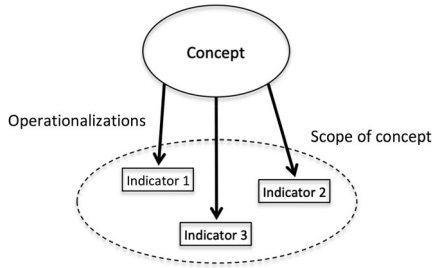

Fig. 1. Elements of a construct.

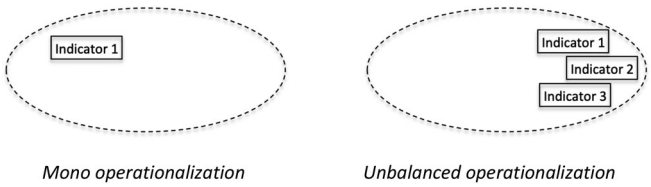

Fig. 2. Examples of construct underrepresentation.

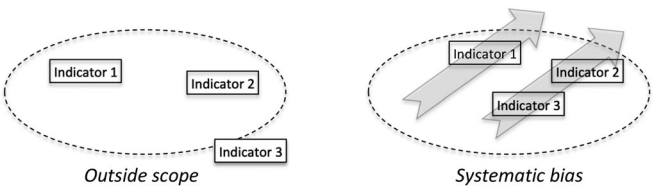

Fig. 3. Examples of construct-representation bias.

Section 3 reports the analysis of CV awareness. Section 4 reports the review. Section 5 describes the guidelines and demonstrates their use. Section 6 discusses limitations and Section 7 related work. Section 8 concludes.

2 Construct Validity

2.1 Definition of Construct

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 literature 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 [24] is identical to the notion of concept described above.[2]

In an empirical science, one may need to measure concepts. 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* [17] to represent the concept at an observational level (Fig. 1). The process of determining such indicators is called *operationalization*. A *construct* is a concept that has an associated operationalization into indicators.

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 [18].

2.2 Validity and Validation

Validity is whether an inference, proposition, or claim is (approximately) true or correct [70], preferably supported by evidence or data. Validation is the *process* of investigating validity [9], of which convergent and divergent validation are two well-known types. In this article, a "validated construct" means that the construct has undergone validation.

Convergent validation [14] investigates the extent to which the results yielded by two instruments that measure similar (or the same) concepts converge. Convergence of indicators may also be investigated, for example, whether a new indicator of system size has similar values to established indicators, such as lines of code (LOC) and function points. As part of convergent validation, one may calculate internal consistency reliability (e.g., Cronbach's alpha), or conduct a principal components analysis (PCA) or exploratory factor analysis [21]. Somewhat more advanced is to use confirmatory factor analysis [38], which requires an explicit, *a priori* model of the relationship between a concept (factor) and its indicators (observable variables). One can then test, for example, whether the variables load on the same factor according to the model.

Divergent validation[3] [14], [55] investigates the extent to which a set of indicators represents one specific concept only and no other. This type of validation might be used to determine whether two concepts that might be expected to differ (as specified in the intension of the concept) do in fact diverge according to available data. For example, if one has high skill in programming, one would learn little that is new when solving a trivial programming task. Conversely, if one has low skill, one would learn much by solving the same task. Therefore, one would expect learning and programming skill to diverge and be negatively correlated [6]. A technique used in both convergent and divergent validation is the Multitrait-Multimethod Matrix (MTMM), where each concept (trait) is measured by multiple data collection methods, and each method is used to measure multiple concepts [14].

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

2.3 Threats to Construct Validity

One major threat to CV is a missing or inadequate definition of the concept. Two other major threats concern operationalization: The indicators may underrepresent the concept, and there may be bias in the representation of the concept.

2.3.1 Inadequate Definition of Concept

CV requires an unambiguous definition or an established understanding of the concept involved in the construct. Without such a basis, it is difficult to evaluate how well a set of indicators represents the concept. The threat to CV that Cook and Campbell [19] refer to as the "inadequate preoperational explication of constructs" is informally explained by Trochim *et al.* [81]: *"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, lead time may have different definitions in a software engineering context [77]. It may denote the amount of time between the proposal for a new feature or another request and its deployment in the customer's environment. Alternatively, it may denote 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.

Inspired by Mark [49], we categorize ways researchers might err when defining or identifying concepts:

1) the concept may be at too general a level;
2) the concept may be at too specific a level;
3) the concept is not the right one, even though it is at the appropriate level of abstraction.

Suppose the real concept of interest is "programming skill". The identification, then, of "software engineering skill" is an example of the first type of error, "Java programming skill" is an example of the second, and "usability skill" is an example of the third.

Ideally, already well-defined concepts should be used if available. In particular, a theoretically defined concept, which is based on an underlying theory that describes the concept, provides guidance on how to operationalize that concept. 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 [33]. Unfortunately, most of the concepts involved in software engineering research are not theoretically defined, which is on a par with the fact that, in general, theories in the field are relatively under-used [36].

2.3.2 Construct Underrepresentation

Construct underrepresentation [54] 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.[4] The identification of these important aspects 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 consult 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 operationalization also requires measurement expertise.

Fig. 2 (left) shows the result of a single operationalization (also referred to as *mono-operation* [19]); 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, for which the number of bugs is only one of several possible indicators.

Detecting single operationalizations is trivial (counting to one). However, determining the quality of a single operationalization is not trivial. Measuring the concept of program length in terms of LOC would not be a serious case of underrepresentation, but using LOC to measure the concept of software complexity would be serious because important aspects of software complexity are not captured by LOC alone. For example, poor naming of identifiers [26] would be another indicator that the software is complex, that is, "not easy to analyze or understand" [37].

Fig. 2 (right) shows another example of construct underrepresentation where there are multiple indicators that nevertheless do not span the concept sufficiently because they are clustered and unbalanced. An example is if software complexity is measured in terms of both LOC and function points. Even though they 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 $> .85$ to $> .95$) may indicate that the questions do not span the concept being measured and thus cause construct underrepresentation [59].

2.3.3 Construct-Representation Bias

The third major threat to CV is construct-representation bias.[5] 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 form

4. Underrepresentation has also been referred to as "incomplete representation" [78].

5. Messick [54] 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, and at the same conceptual level as *construct underrepresentation*.

the concept. Evaluating CV would then be meaningless because there is nothing more to the concept than what the variables define.

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 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* [19]. 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. An example of avoiding mono-method bias is when the quality of code is evaluated using objective code metrics provided by tools in addition to the subjective opinions of experts.

A construct with only one indicator has a mono-operation bias [19] and thus faces two threats. The first is underrepresentation, because there is only a single indicator. The second is construct-representation bias, because any (random or systematic) measurement error in the indicator will be included in the representation of the concept.

2.4 Internal, External, and Statistical Conclusion Validity

According to Rogers [62, p. 246], Courtis [20] provided the first "institutional definition of validity" in 1921, in the context of psychological testing.[6]

Internal validity refers to the extent to which an observed co-variation between treatment and outcome variables reflects a causal relationship between treatment and outcome concepts [70]. Threats to internal validity are factors that negatively affect, or threaten, the inferences about such a causal relationship.

6. "Two of the most important types of problems in measurement are those connected with the determination of what a test measures, and of how consistently it measures. The first should be called the problem of validity, the second, the problem of reliability." [20], as cited in [64, p. 80]. Since then, several distinct aspects of validity have been described and named in the literature. In software engineering, empirical studies typically focus on internal validity, external validity, and statistical conclusion validity, in addition to CV [85].

In the context of causal or explanatory case studies, Yin describes internal validity as "the strength of the causal or other 'how' and 'why' inferences made in a case study, in part bolstered by showing the absence of spurious relationships and the rejection of rival hypotheses" [87, p. 351]. Even though cause and effect may be studied in case studies [52] and surveys [12], experiments are considered the primary research method for causal inference [69], [70].

Research questions may include causal relationships at the conceptual level (i.e., *A* affects *B*), while hypotheses and data analyses used to investigate such relationships occur at the operational level using variables that represent *A* or *B*. Fig. 4 shows the conceptual and operational levels for such cause-effect studies.

The figure also shows a *moderator* construct. A moderator changes the *strength* and/or *direction* of the relationship from cause to effect [4]. A change of direction means that a positive effect becomes negative, or vice versa. In regression and variance analyses, a moderator variable is often modelled as an interaction effect.

Unlike a moderator, a *mediator* is on the causal chain (not shown in the figure); the relationship from cause to effect is not direct but indirect via the mediator [4]. For example, the effect of working memory capacity (WMC) on code review performance was studied as a direct relationship by Baum *et al.* [A8]. However, a previous study reported that the effect of WMC on programming performance was mediated by programming knowledge [8]. Thus, an alternative causal chain in the study of Baum *et al.* could be that the effect of WMC on code review performance was mediated by code review knowledge.

Related to internal validity is also the notion of *confounding* [70]. Spurious relationships between the cause and effect concepts may occur due to a confounder (also called confounding factor or confounding variable). A confounder is associated, or correlated, with both the cause and effect concepts but is not on the cause-effect chain. For example, the use of sunglasses will typically correlate with the sale of ice cream. However, it is not because people wear sunglasses that they buy ice cream; instead, temperature, a confounder, causes an increase in both the use of sunglasses and ice cream sales. In software engineering, suppose one observes that software systems with a large share of design patterns are more maintainable than systems with a small share of design patterns. A potential confounding factor is that developer experience affects both the use of design patterns and the general maintainability of the systems (beyond a possible improvement of maintainability due to the use of design patterns). The standard method to reduce confounding effects in experiments in general is to randomly allocate the units (which in human-centric experiments are participants) to treatment and control groups.

External validity refers to whether a study's findings are generalizable beyond the immediate study [86]. In case studies, one often generalizes *analytically* using theory [86]. In experiments and surveys, however, one typically generalizes *statistically* from the elements that are sampled to the population of interest. In addition, one may wish to generalize the findings of a study to elements that the study did not

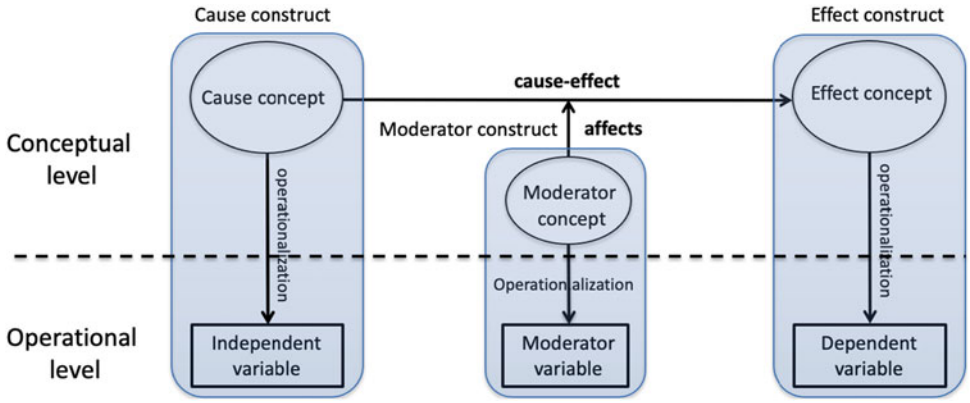

Fig. 4. Reference model for constructs in cause-effect studies.

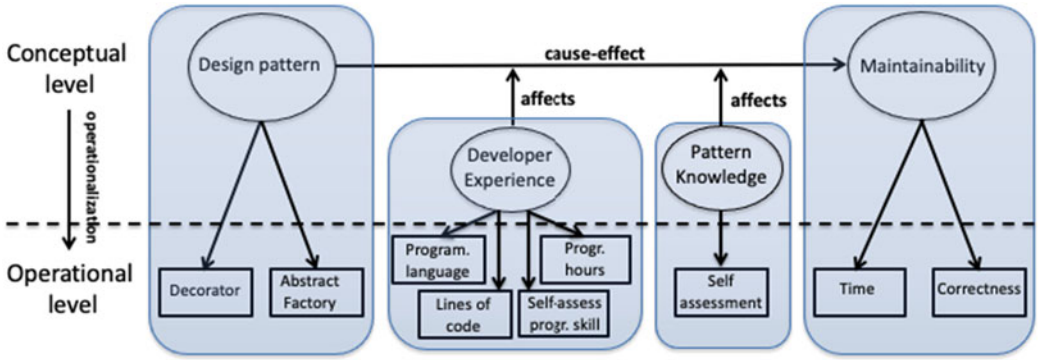

Fig. 5. Model of constructs in an experiment replicated by Krein *et al.* [A39].

include [70]. For example, even if only students participated in a study, one may argue that the results also hold for professionals [31]. In software engineering, the elements may belong to different types, including *actors* (individuals, teams, projects, or organizations), *technologies* (process models, methods, techniques, tools, or languages), *activities* (plan, create, modify or analyze (a software system), and *software systems* (characterized by size, complexity, application domain, project (business, scientific, student), type (administrative, embedded, real-time), etc.) [72].

Statistical conclusion validity refers to the appropriate use of statistics to infer whether the independent and dependent variables correlate, or co-vary [19]. Typical aspects addressed are choice of statistical tests, sample size, measurement reliability, effect size [40], and statistical power [28].

To summarize, internal validity concerns inferences between the cause and effect concepts, representing a horizontal dimension at the conceptual level; see Fig. 4. Statistical conclusion validity concerns statistical analysis on the cause and effect variables, representing a horizontal dimension at the operational level. In contrast, CV represents a vertical dimension in Fig. 4. External validity concerns generalizations from the elements used in a study to other elements of the same type that were not studied.

2.5 Construct Validity in an Example Experiment

The reference model in Fig. 4 is instantiated in Fig. 5, using the experiment reported by Krein *et al.* Their experiment replicates another replication [82], which in turn replicated the original experiment that investigated the effect of using design patterns on maintainability [61]. The replication presented in Fig. 5, whose constructs this section discusses, focuses on the moderating effects of developer experience and pattern knowledge.

The terms independent and dependent variables are sometimes used at the conceptual level. However, we will use the term variable only at the operational level. Moreover, indicators are variables but not vice versa: an indicator is a particular variable that is associated with a concept that is not directly measurable. 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" [32].[7]

2.5.1 Independent Concept: Design Pattern

The concept of design pattern is adequately defined (Section 2.3.1) by Krein *et al.* [A39] through a reference to a well-known book on design patterns [34], which clearly describes it.

Concrete patterns must represent the concept at the operational level. The experiment by Krein *et al.* used the Decorator and Abstract Factory patterns. Use of concrete

7. 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.

patterns in experimental groups may be modelled by an indicator variable ("1" if the pattern is present, otherwise "0", i.e., a dummy variable). To what extent do these two instances underrepresent or give a biased representation of the concept of design pattern? Gamma *et al.* list 23 patterns in their book published in 1995 [34]. Since then, many more patterns have been proposed. Even though the operationalization does not suffer from mono-operation, using only two patterns may still under-represent the concept. In both the original experiment [61] and the previous replication [82], the patterns Observer and Visitor were used in addition. However, Krein *et al.* excluded Observer, because it was only used in a programming task that appeared to be too easy in the previous replication and thus posed a threat to external validity, not CV. Krein *et al.* also excluded Visitor because "the prior two studies both found the Visitor pattern to be overly difficult" [A39]. Including Visitor might then have made the sample of design patterns unbalanced towards difficult patterns. Ideally, underrepresentation might have been reduced if a set of (say five or more) design patterns had been selected randomly.

We observe no construct-representation bias regarding scope (Fig. 3); Decorator and Abstract Factory are clearly instances of design patterns. However, it is difficult to judge whether the mono-method of data collection is present because there is no description of how the four design patterns were selected in the original experiment, which, in turn, determined the patterns used in the replications.

The original experiment and the replications found different effects of the different patterns. Krein *et al.* [A39] quote Vokac *et al.* [82]: "each design pattern ...has its own nature, so that it is not valid to characterize patterns as useful or harmful in general". Similarly, in another study, four functionally equivalent Java systems were developed independently [2]. Each of the systems could be considered most maintainable depending on the indicators used to measure maintainability [73]. Thus, using different indicators may lead to different conclusions of a study.

2.5.2 Moderating Concept: Developer Experience

Krein *et al.* introduce "developer experience" and "pattern knowledge" as moderator concepts of the investigated causal relation. Concepts commonly understood and agreed upon in a given context, such as "developer" among TSE readers, do not require an explicit definition to ensure CV. However, "experience" is ambiguous, as discussed by Dieste *et al.* [A20]. They refer to two dictionary definitions of experience: "(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)". Siegmund *et al.* [71] adhere to the first interpretation when they define programming experience as "the amount of acquired knowledge regarding the development of programs". It is apparent that Krein *et al.* adhere to the second interpretation, as discussed by Ericsson [30], both from the choice of the indicators and that "knowledge" is used instead of "experience" in the other moderator ("pattern knowledge"). Nevertheless, it is prudent to be explicit in unclear cases; judging how well a set of indicators represents a concept relies on its definition.

Four indicators operationalize "developer experience": number of programming languages, lines of code written, number of hours programming, and self-assessed programming skill. Consequently, the construct does not face mono-operation. However, it does face the threat of mono-method bias because all the indicators are based on self-reporting.

2.5.3 Moderating Concept: Pattern Knowledge

Krein *et al.* do not explicitly define the concept of pattern knowledge. However, like "developer", "knowledge" is a term that is commonly understood. Thus, "pattern knowledge" would also be easily understood given that the concept of (design) pattern is already defined and explained.

The concept of pattern knowledge is operationalized by only one indicator and thus the construct suffers from mono-operation. The construct also has mono-method bias because the only method used is self-reporting. Objective tests of pattern knowledge might also have been used.

A major finding in the experiment is that both developer experience and pattern knowledge change the *direction* of the effect of using design patterns; the effect is positive if experience and knowledge are high and negative if they are low. Such a moderating effect of expertise is also referred to as the *expertise reversal effect* [39], [74].

2.5.4 Dependent Concept: Maintainability

Maintainability [48] denotes how easy it is to maintain a software system and is a commonly understood concept in the software engineering community. Maintainability may be evaluated regarding the effort needed for software developers to perform maintenance tasks and the quality of the outcome,[8] as is the case in the experiments conducted by Krein *et al.* However, quality is underrepresented as it only focuses on one aspect, correctness. Another quality aspect might have been whether the maintainability had been improved or deteriorated after the tasks had been performed. However, resources are limited. Correctness as the single indicator of quality is a pragmatic choice in the original and replicated experiments. Nevertheless, they are examples of comprehensive and resource-demanding experiments in software engineering.

3 Awareness of Construct Validity in Software Engineering

This section reports on the awareness of CV as reflected in the general software engineering research literature and in specific software engineering journals and experiment articles, respectively.

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.

8. A general challenge is how to weigh different indicators of performance in an analysis, for example, when combining time and quality [5].

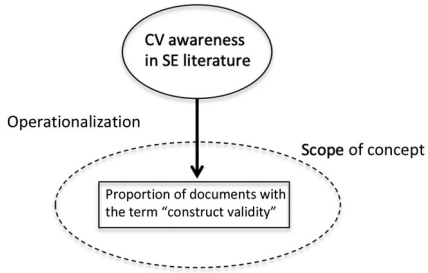

Fig. 6. The construct "CV awareness in SE literature".

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 (see Fig. 6); 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.

Producing Figs. 7 and 8 required 180 Google Scholar searches. Each of the 20 years in the period from 2000 to 2019 entailed nine searches: three denominators common for both figures ("information systems", "software engineering", and "hardware"); three numerators for Fig. 7 ("construct validity" in each of the three categories); and three new numerators for Fig. 8 ("external validity", "internal validity", and "conclusion validity").

However, there is a probabilistic component in Google Scholar. Identical searches within a few hours usually give a unique estimate, but the same searches a day later differ, possibly by more than 15 percent, particularly for broad searches. Averaging over time will improve the stability and reproducibility of the numbers but may need to be repeated over several weeks. An alternative method that can be executed in a few minutes is to exploit the fact that even though Google Scholar is case-insensitive, the same search string but with different use of uppercase and lowercase letters, e.g., "software engineering" and "Software engineering", will give different counts. It appears that a query for "Software engineering" does not use the cached value of "software engineering" but makes its own estimate. One can then average over different letter cases, e.g., "soft..", "Soft..", "sOft..", and "SOft..". This method is used by the Python script in Appendix A, which can be found on the Computer Society Digital Library at http://doi.ieeecomputersociety.org/10.1109/TSE.2022.3176725, available in the online supplemental material.

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: *IEEE Transactions on Software Engineering (TSE)*, *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 calculated the proportion of research articles that included the term construct validity for each year in the period 2000 to 2019. We used the 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. The search procedure and a supporting R script

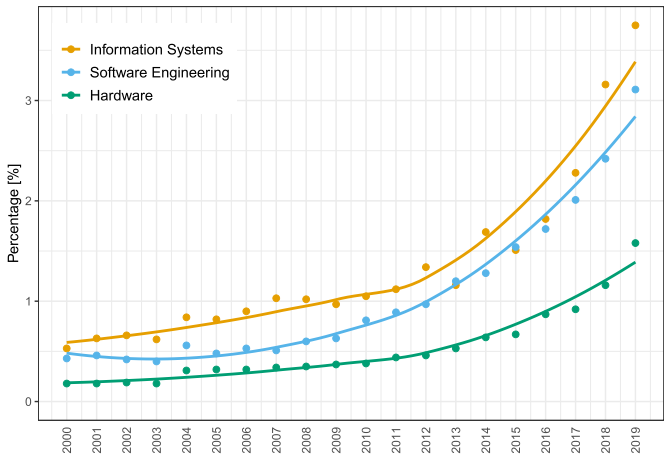

Fig. 7. Percentage of documents in Google Scholar that contain "construct validity" in three subdisciplines.

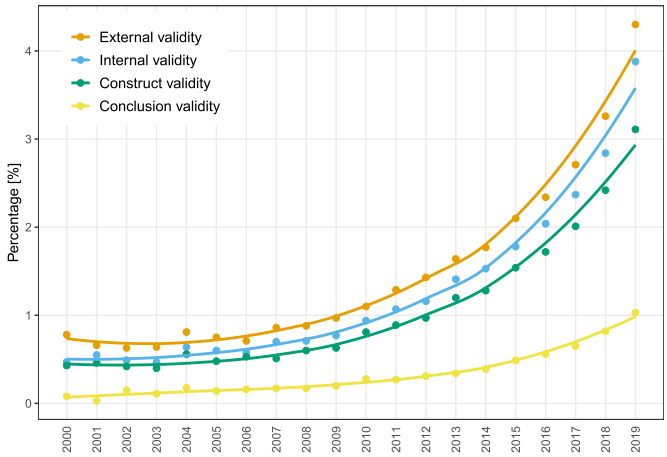

Fig. 8. Percentage of software engineering documents that contain "external validity", "internal validity", "construct validity", and "conclusion validity", respectively.

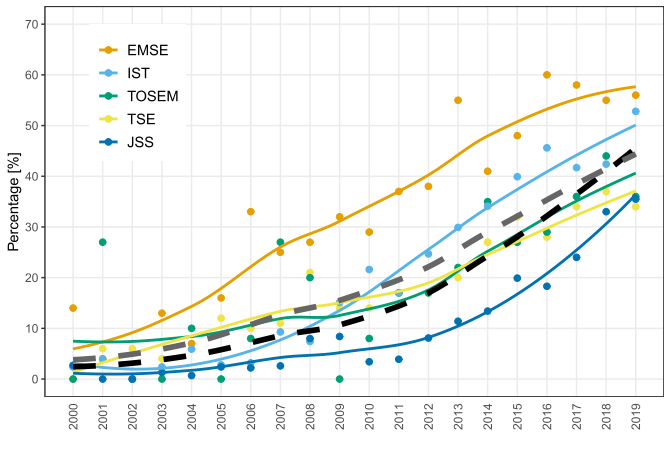

Fig. 9. Percentage of articles that contain "construct validity" in five journals. Weighted/unweighted means in dashed black/grey.

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)

are shown in Appendices B and C, respectively, available in the online supplemental material.

3.1.3 CV in Software Engineering Experiments

We identified the human-centric experiments published in the five journals listed in Section 3.1.2 for the period 2015 to 2019[9] using the procedure and definition of human-centric experiment described in Appendix D, available in the online supplemental material. 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. Fig. 7 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 that of other computing disciplines, we conducted searches on "information systems" and "hardware" that were similar to those for "software engineering"; see Fig. 7. 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.

Regarding the types of validity commonly discussed in the software engineering research literature, Fig. 8 shows that "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.

9. 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.

3.2.2 CV in Specific Software Engineering Journals

For the five specific software engineering journals, there is a similar pattern of growth (Fig. 9) as in the general software engineering literature. The 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 [76]. 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 increased from 2.8 percent in the period 1993 to 2002 to 4.5 percent in the period 2015 to 2019. Neto and Conte [56] reported a study of experiments published in nine journals (including the five just mentioned) and four conferences in the period 2003 to 2011. In their sample of 46 papers, 10 discussed CV (22 percent).

4 Construct Validity Discussions in Software Engineering Experiments

This section reports how the 83 experiment articles identified in Section 3.1.3 define CV and discuss threats to CV.

4.1 Data Collection

We analyzed a set of articles reporting human-centric experiments in depth because we expected to find a higher frequency of CV discussions in such articles than in those reporting other kinds of empirical studies.[10] Operationalizing the concepts of the research questions of an experiment that are not directly measurable is required before descriptive statistics or results of statistical hypothesis testing can be reported. Particularly, in human-centric experiments, which focus on "software engineering methods, techniques and procedures that depend on human expertise" [45], the concepts studied (e.g., usability, maintainability, competence, understanding, difficulty) tend to be more abstracted than those studied in experiments that do not involve humans (e.g., time, defect).

4.2 Analysis Method

Both authors read the whole 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.

10. Data from TSE in the period 2015 to 2019 supports our expectations: 9 of 11 (82 percent) of the human-centric experiments discussed CV (see Table 1), whereas the proportion was 20 of 48 (42 percent) for the qualitative studies to be reported in Section 5.3.

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

To help ensure that we did not miss relevant CV information through our reading, we searched through plain text versions of the articles, using Linux *grep -i* (case-insensitive), for the terms "bias", "confound", "construct", "effect", "indicator", "mono-operation", "mono-method", "operationalization", "monomethod", "threat", "underrepresentation", "validation", and "validity", including spelling variations ("monomethod", "monooperation", "operationalisation", and "under-representation"). We also searched for seven named types of validity that we argue are peripheral to CV; see Section 4.4.5 and Table 4.

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; see Table 3. The sentences 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 4.4.4. A large group of issues that we consider outside CV are described in Section 4.4.6.

In most cases, we quickly agreed on the encoding. In three 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.

4.3 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.* [84], [85],[11] which provides the statement, "Construct validity is concerned with the relation between theory and observation" [85, 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 [79] 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) [25], 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 [36], [72] but only as something that resides at the conceptual (abstract) level and in contrast to the operational (concrete) level, as shown in Fig. 4. A similar figure, which distinguishes between the levels of "theory" and "observation", is shown in Trochim [80] and adapted in [85]. 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 is "the extent to which the experiment and its various measures test and measure what they claim to test and measure". This definition adheres to an early definition of validity (before CV was conceptualized) given by Buckingham [13] in the context of intelligence testing: "Validity [is] the extent to which [intelligence tests] measure what they purport to measure".

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 [15], [42]. 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" [9, 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 on the intention (for example, measured by inter-rater agreement), the subjective nature of the intention makes it challenging to agree on the operationalization of the concept. If a concept is vaguely and

11. Appendix E shows a Sankey diagram of the references that the authors use to support their definition of CV.

TABLE 3
Threats to CV that are Present, Mitigated, or not Present in 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 [All], [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]	Monomethod [A18] Subjectivity [A33], [A49], [A53], [A56], [A65], [A79] Indicators outside scope [A30]	Mono-method [A32], [A74] Subjectivity [Al], [A4], [A7], [A15], [A18], [A43], [A75], [A76], [A78]	**21**
Total		**25**	**26**	**18**	**69**

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]	[Al], [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]	[Al], [A9], [A16], [A17], [A23], [A24], [A25] ,[A28], [A32], [A72]	**17**
Experimenter expectancies	[A32], [A51], [A55]	[Al], [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**

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 [53] and when they equate software maintainability with the maintainability index [58]. Software complexity and maintainability are certainly more comprehensive concepts than can be fully represented by these metrics [73].

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 and illustrated in Fig. 4.

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).

4.4 Results: Threats to CV

This section reports on discussions in the 83 analyzed articles of the three core threats to CV introduced in Section 2.3. Table 3 summarizes the findings. The table includes three columns that indicate whether the authors of the articles consider a given threat to be present, mitigated, or not present. The frequency of specific threats shown in the table is as follows (the percent of the total of 69 is given in parentheses): mono-operation 28 (41%), subjectivity 19 (28%), multiple indicators 7 (10%), imported definition 3 (4%), mono-method 3 (4%), definition by examples 2 (3%), different definitions 2 (3%), disagreement about extension 2 (3%), indicators outside scope 1 (1%), no operation 1 (1%), and redefinition 1 (1%).

The first three subsections discuss the core threats to CV. The remaining three subsections report on, respectively, import of constructs, peripheral threats to CV, and other kinds of validity than CV.

4.4.1 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 mitigate [A73] or avoid [A18], [A28] the threat that their concepts are inadequately defined.

Definition by Examples. Two articles discuss how a concept is formed from concrete examples, cf. *denotative definition* [17]. 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. Bernandez *et al.* [A9] investigate the effect of mindfulness in the performance of conceptual modeling. They state that to avoid "inadequate pre-operational explication of constructs", 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.

4.4.2 Construct Underrepresentation

Twenty-eight of the articles address threats to construct underrepresentation, which we categorize as follows.[12]

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 operationalization" 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 [81], 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 [19], [70].

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".

4.4.3 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 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

12. Note that one article may appear several times in Table 3.

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 Fig. 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".

4.4.4 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) [25] to evaluate technologies. Two articles [A23], [A31] use a framework for evaluating comprehensibility of conceptual models described in [3]. This framework is, in turn, based on several theories.

Psychometric Tests. The fields of psychology and education have developed many 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 [11] and the Net Promoter Score questionnaire [35], 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.

4.4.5 Peripheral Threats to CV

Sections 4.4.1– 4.4.3 discussed the three core types of threats to CV, which are also the first three of ten types defined by Cook and Campbell [19]. The remaining seven types are listed in Table 4. Almost half of the articles discuss at least one of those seven types. All the articles with citations regarding these types cite Wohlin *et al.* [84], [85], who brought forward the CV types introduced by Cook and Campbell [19].

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 [70, p. 73]. The fifth type, "evaluation apprehension", was subsumed in a broader CV type called "reactivity to the experimental situation" in [70]. Furthermore, four types that were categorized as internal validity in [19], were later categorized as CV in [70]. 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 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 "threat" (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. The 16 articles that mention this kind of threat consider it mitigated

or not present, indicating that it is of little concern 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" [19, 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" [19]. 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 [70].

Restricted Generalizability Across Constructs. Cook and Campbell [19] 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 their potentially poor performance may become visible to others, which is termed evaluation apprehension [65]. 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 [19], also referred to as the "observer-expectancy" effect [66], 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. Ideally, a double-blind study may be designed to deal with this effect where 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 [81] 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 and duration of mindfulness training sessions) [A9]. None of the analyzed articles that discuss this threat [A9], [A61], [A52], [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 confound, which are all mostly used in descriptions of threats to internal validity in the general literature; see Section 4.4.6. Accordingly, three of the articles investigated [A61], [A72], [A73] describe problems of internal validity when they discuss "confounding constructs and levels of constructs". One article [A52] describes problems of external validity. The remaining article [A9] just stated that this confounding problem 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 2.3.

4.4.6 Study 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 our definition. We recommend that these aspects are discussed under threats to internal, external, or statistical conclusion validity or as general limitations of the reported study.

Internal Validity. The articles discuss inequality between treatment and control groups with respect to a range of factors as part of threats to CV:

- 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].

However, because unequal treatment groups may result in wrong inferences about the causal relationships, we consider such group differences to be a threat to internal validity.

Furthermore, 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. As part of threats to CV, several of the articles discuss generalizations regarding participants, tasks, and material used in the experiments. However, according to the description in Section 2.4, such generalizations should be discussed as part of external validity. The

validity of using students is discussed in [A64], [A83], which is a subject of recurring debate in software engineering [31]. Whether the 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]. A concern is also whether the experimental material is representative for material used in industry, which includes system requirements [A65], code [A56], [A62], code comments [A11], and seeded faults [A2], [A27].

Measurement Scales. Under their discussions on threats to CV, four articles [A8], [A21], [A22], [A51] reflect on 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 accurate 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 [A33], [A6] simply state that they use a Likert scale. Core literature and textbooks describe measurement scales as separate from CV [54], [57], [70]. We share the view of Kaner and Bond [41]: "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 heterogenity of units may be better discussed under statistical conclusion validity, as recommended by Shadish *et al.* [70].

Fundamental Study Requirements. Empirical research assumes willing participants who follow the procedure of a study. The issues below are discussed under threats to CV in the articles, but we recommend they be discussed as limitations of the reported study:

- 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 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.

We illustrate the use of the guidelines by applying them specifically to the construct "CV awareness in software engineering literature'' investigated in Section 3 and generally to a set of qualitative studies published in TSE.

5.1 Proposed Guidelines

The guidelines, numbered G1 to G5, are shown in Table 5.

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 or table, that includes the concepts, their indicators, and the relationships among the concepts. While creating the model, for each construct, do G2 if possible, else do G3.
G2	Import constructs if possible	Import already validated constructs from established theories, frameworks, standards, tests, instruments, or questionnaires, or refer to studies that have formerly validated the constructs. If the imported construct needs modification, consider Steps G3.1-G3.3.
G3	Define new constructs	
	G3.1 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.
	G3.2 Avoid construct underrepresentation	Select indicators that span the concepts. In general, the more indicators that are non-overlapping, the less underrepresentation.
	G.3.3 Avoid construct-representation bias	Select indicators that avoid systematic bias in the representations of the concepts; for example, avoid using only one data collection method or collecting only subjective data.
G4	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.
G5	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.

5.1.1 Create Model of Constructs

The constructs of a study typically correspond to the concepts used in the research question of the study. However, for some of the analyzed articles, it is difficult to obtain an overview of the constructs under study and how they are used. We recommend creating a model of the constructs 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 if Possible

One-fifth of the articles imported 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. Note that there should be evidence that the imported construct has been subject to a validation process.

5.1.3 Define New Constructs

If a suitable construct is not found under G2, define a construct according to guidelines G3.1–G3.3. These guidelines could also be followed if the concept definition or set of indicators of the imported construct need modification.

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* [42], 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 the concepts have different names (*jangle*). An example of a jingle fallacy is the two definitions of experience described in Section 2.5.2: "(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 fallacy is if *experience* and *expertise* are both defined by (2). If a jingle fallacy is present, it will become clear which definition the authors adhere to if they refer to a specific alternative in a dictionary definition (as for *experience* above) or to a source that defines the concept in the meaning intended by the authors. Alternatively, the authors themselves must carefully define the concept as intended. If a jangle fallacy is present, it may be useful to mention that another concept (with another name) also has, or may have, the same definition.

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 [75]: 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.

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 4.4.3). 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.* [27] identified differences between subjective and objective measures of software methodology usage. Generally, to avoid mono-method bias, one should use several methods, that is, method triangulation [14], [60].

5.1.4 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 our guidelines. 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 4.4.6. In particular, we recommend that the set of threats to CV proposed by Cook and Campbell [19] more than 40 years ago (Table 4) should not be regarded as threats to CV. Moreover, even though

measurement scales are central in measurements, we argue that such threats should be included under statistical conclusion validity.

5.1.5 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 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 discussed.

5.2 Guidelines Applied to a Specific Construct

Below, we apply the guidelines numbered in Table 5 to the construct "CV awareness in software engineering literature".

G1 Fig. 6 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.

G3.1 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.

G3.2 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, i.e., 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. Certainly, there are documents where it is not relevant to mention CV. Such cases would not indicate a lack of awareness. Thus, when we report increased awareness, there is an assumption that the proportion of relevant documents has increased at a lower pace than the proportion of documents with the term construct validity. When claiming that a sevenfold increase over two decades in the appearance of the term construct validity shows an increase in awareness, it is assumed that the increase in the relevance of the documents is (much) less than sevenfold. Still, a challenge is that our single indicator is biased (Section 2.3.3) in some way because the extent to which a variation in relevance affects our indicator of awareness is unknown.

G3.3 Our approach faces mono-method bias because we use only text analysis to collect data. If we had had the extra indicator (item (3) in G3.2 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.

G4 By following the guidelines, we have only discussed the three core aspect of CV. Other quality aspects are discussed under limitations; see Section 6.

G5 We explicitly stated the construct in question and what the threats to CV were.

5.3 Guidelines Applied to Qualitative Studies

To assess the applicability of the guidelines beyond experiments, we identified 20 TSE articles published in the period 2015 to 2019 that included the term "construct validity" in the body text and the terms "case study" (16 articles), "field study" (2 articles), or "qualitative study" (2), in the title, abstract, title, or set of keywords. (One method article that included "case study" as an index term was excluded.) Twenty-eight other articles met the same search criteria apart from "construct validity" in the body text.

Using the categorization in Table 2, the articles define CV as, respectively, *represents the concepts of study* (3 articles), *the relationship between theory and observation* (2), *the measure of intention* (1), *miscellaneous definitions* (1), and *no definition* (13). Regarding threats to validity, three articles cite Wohlin *et al*. [84], [85], who define CV as *the relationship between theory and observation*. Another three articles cite the work by Runeson *et al*., on case studies in software engineering [67], [68], which states that CV "reflects to what extent the operational measures that are studied really represent what the researcher have in mind", which complies with our category of *the measure of intention*. Yet another three articles cite Yin [86], who defines CV as the "accuracy with which a case study's measures reflect the concepts being studied", which complies with our category of *representing the concepts of study*. The only definition found in the experiment articles not found in the qualitative articles is *the relationship between experiment goals and results*.

Our informal assessment of the 20 articles regarding the guidelines found the following:

G1 One-fourth of the articles include figures, tables, or structured text that show clearly how the core concepts of the investigation are operationalized. Only some of the constructs are sufficiently illustrated or described in another two-fourths of the articles. We spent considerable time discerning how the concepts were operationalized for the remaining articles.

G2 Two-thirds of the articles import at least one construct from methods, tools, frameworks, or predefined sets of measures. No article imports all the constructs used, which is unsurprising given that none of the reported studies are replications.

G3.1 The articles vary from assuming that a core concept is generally well understood to discussing the definition of the concept in depth. In one study, a tool was developed to identify the extension (Section 2.1) of the concept of configuration constraint in a set of software systems, which required a particularly precise and unambiguous definition.

G3.2 Five articles involve constructs with mono-operationalization, of which three did not address this limitation as a threat. The remaining articles have made efforts to avoid construct underrepresentation.

G3.3 The five articles with mono-operationalization also involve mono-method (as the former implies the latter). One of those articles states in its CV discussion that there might be bias in measuring time for fixing bugs. Another article explicitly follows Yin's recommendation of using multiple sources of evidence (data sources) to improve CV [86]. Multiple sources help reduce both construct underrepresentation and construct-representation bias.

G4 Two-thirds of the articles discuss threats outside the way CV is defined and interpreted in this article. Examples are *confounding factors* (Section 2.4), *maturation* [19], and *evaluation apprehension* (Section 4.4.5), which belong to internal validity. We consider *skewness distribution* and *log transformations* to belong to statistical conclusion validity and *data correctness* to fundamental study requirements (Section 4.4.6).

G5 The articles vary from clearly stating which constructs face which threats to vague statements like "to reduce CV threats, we [did something]". In our view, the readability and understandability of CV could be improved in most cases if the constructs and threats were described explicitly.

As illustrated above, the qualitative articles comply with the proposed guidelines to varying extents. If all articles had fully complied with these guidelines already, then they might have been redundant. In summary, our assessment indicates that the guidelines apply to qualitative studies even though they are based on experiments; see further discussion in Section 6.3.

6 Limitations

This section discusses limitations of our quantitative and qualitative studies and the general applicability of the guidelines.

6.1 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, since we investigated thousands of documents, it follows from the law of large numbers that the overall effect of random errors (noise) is likely to be small.

Furthermore, it is much more likely that people who use a specific term are more aware of the phenomenon denoted by the term than those who do not. Nevertheless, documents might include the term construct validity accidentally or deliberately without the author being "aware of CV". However, such noise may be ignored when the number of documents is large. Therefore, a sevenfold increase in the number of occurrences of the term construct validity would reflect a substantial increase in the awareness among the document authors.

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 between the general software engineering literature, selected journals, and selected experiment articles. However, the increase is evident for all of them, strengthening the overall result.

6.2 Qualitative Study of CV Discussions

Some of the CV analyses reported in Section 4 are objective, for example, the frequency of articles with specific terms (e.g., CV types reported in [19]). 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 [76] and specifically on CV when we constructed and validated an instrument for measuring programming skill [6], [7].

In the investigated literature, CV may have been discussed under other terms (cf. the jingle-jangle fallacy) and thus may not be detected in our study. For example, the term theoretical validity has been used synonymously with CV in the quality research literature [43]. As such, Fig. 9 may underreport the number of articles that discuss CV. However, only 29 of all the 7538 research articles published in TSE, EMSE, IST, TOSEM, and JSS in the period 2000 to 2019 (i.e., 0.4 percent) use the term theoretical validity. None of the 29 articles reported experiments. (In particular, none of the 83 experiments that we investigate uses the term.) Nine of those articles explicitly use the term construct validity to distinguish CV from theoretical validity and other types of validity. The 29 articles describe theoretical validity in five ways.

1) One group refers to the understanding described by Maxwell [51]: "theoretical validity closely matches what is generally known as construct validity, and is primarily what Kirk and Miller [43] mean by theoretical validity. The second aspect includes, but is not limited to, what is commonly called internal or causal validity [19]".
2) Another group refers to the interpretation of theoretical validity given by Briand *et al.* [10], which resembles CV in that "theoretical validation is concerned with demonstrating that a measure is measuring the concept it is purporting to measure", albeit within the formalism offered by representational measurement theory [47], [63].
3) Yet another group refers to the framework proposed by Kitchenham *et al.* [44], which is also based on representational measurement theory but widens the understanding of theoretical validation to also include both direct and indirect measures, measurement unit definitions, measurement instruments, and measurement protocols (data collection procedures).
4) A few articles discuss confounders and learning effects under theoretical validity, which we consider

part of internal validity as it refers to the *relations* between constructs and not the constructs per se; see Section 4.4.6.

5) The final group of articles uses the term theoretical validity without explicating what it means in the article.

We are not aware of any synonyms for CV other than "theoretical validity", but articles may discuss CV issues without using any specific term. For example, among the 19 experiments in Table 1 that do not use the term CV, one article discusses validity related to measuring the usability of security APIs [83].

6.3 General Applicability of Guidelines

We do not claim any generalizability of the results of the review of the 83 human-centric experiments beyond what is reflected in the guidelines.

A threat to the external validity of the guidelines may be that they are not based on other types of study than human-centric experiments. Three specific challenges are

1) whether the guidelines are unsuitable or useless when using other research methods,
2) whether the guidelines are redundant because they are followed anyway by researchers who use other methods, and
3) whether other guidelines should be prioritized or added when researchers use other methods.

The guidelines contain no references to particular aspects of research methods. The research method used will not affect whether a diagram gives a good overview of constructs (G1) or whether a construct is imported (G2). It will also not affect whether a concept is defined adequately, is underrepresented by its indicators, or has a biased representation (G3). The guideline that constructs and threats should be explicitly mentioned when discussing threats to CV (G5) is clearly independent of the research method. Finally, while an excessive interpretation of CV (G4) is more subjective than the other guidelines, we cannot envisage this guideline would depend on the research method.

The set of qualitative studies analyzed in Section 5.3 varied in the degree to which they followed the guidelines; they were, therefore, not redundant for those studies. Nevertheless, we cannot exclude the possibility that sub-communities that conduct surveys, action research, experiments without human subjects, etc., already comply with the guidelines.

When conducting other types of studies, other guidelines may be added to our proposed set or prioritized over some of those we have included. For example, a guideline may help design questionnaires to better support CV when conducting surveys [1].

Another limitation is that we reviewed only articles published in top-tier journals. However, we believe the guidelines would be helpful and not redundant also for papers published in less prestigious journals, conference or workshop proceedings, or book chapters. Nevertheless, for papers published at those venues, new guidelines may need to be added or prioritized over some of those we have proposed.

Future work may include analyzing papers that claim to follow the guidelines and obtaining feedback from the authors of such papers.

7 Related Work

Already in 1980, Curtis [23] imported discussions of CV from the general psychological literature to the software engineering literature. In a framework for evaluating software metrics, Kaner and Bond [41] 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.* [84], [85] is primarily based on Cook and Campbell [19], whose work then became better known within software engineering.

Ralph and Tempero [62] proposed a set of guidelines for assessing CV. We share the philosophical view of scientific realism as the underpinning of CV.[13] Their guideline on generating a nomological network[14] somewhat 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 encompasses all aspects of a construct") corresponds to our guideline on avoiding construct underrepresentation (G3.2). We also agree on their convergent/divergent approach for CV, which is reflected in our guideline G3.

Ralph and Tempero also 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 4.4.3. Moreover, Messick [54] 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*[15] 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 [22, p. 126], "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 [54], which included topics such as the social consequences of using measurements in particular ways.

13. 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 [16], who discusses CV in relation to *deconstruction* (Derrida), *critical theory* (Habermas), and *interpretive analytics* (Foucault).

14. The term nomological network was coined by Cronbach and Meehl [21]. 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 an unfamiliar term in software engineering.

15. Predictive validity is also referred to as empirical, statistical, and criterion-related validity [57].

More recently, Borsboom *et al*. [9] 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.

8 Conclusion

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 five top-tier journals that we investigated.

We reviewed CV in 83 articles that reported human-centric experiments between 2015 and 2019. 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 concepts. We also advise researchers to be explicit about which constructs face which threats when discussing threats to CV.

One in five of the reviewed articles reuse formerly defined and validated constructs. To save effort and improve quality, we recommend that researchers 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 principles brought forward from the classical literature: state how the concept is defined, how indicators span the concept, and how bias is handled.

CV is crucial to the overall quality of empirical research. We hope that our recommendations, with illustrative examples from the analyzed articles and summarized in a set of guidelines, will help improve how CV is handled and reported in the software engineering community.

Acknowledgments

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

References

[1] N. K. Agarwal, "Verifying survey items for construct validity: A two-stage sorting procedure for questionnaire design in information behavior research," *Proc. Amer. Soc. Inf. Sci. Technol.*, vol. 48, no. 1, pp. 1–8, 2011.

[2] B. C. Anda, D. I. Sjøberg, and A. Mockus, "Variability and reproducibility in software engineering: A study of four companies that developed the same system," *IEEE Trans. Softw. Eng.*, vol. 35, no. 3, pp. 407–429, May/Jun. 2009.

[3] J. Aranda, N. Ernst, J. Horkoff, and S. Easterbrook, "A framework for empirical evaluation of model comprehensibility," in *Proc. IEEE Int. Workshop Model. Softw. Eng.*, 2007, pp. 7–13.

[4] R. M. Baron and D. A. Kenny, "The moderator–mediator variable distinction in social psychological research: Conceptual, strategic, and statistical considerations," *J. Pers. Soc. Psychol.*, vol. 51, no. 6, pp. 1173–1182, 1986.

[5] G. R. Bergersen, J. E. Hannay, D. I. Sjoberg, T. Dyba, and A. Karahasanovic, "Inferring skill from tests of programming performance: Combining time and quality," in *Proc. IEEE Int. Symp. Empirical Softw. Eng. Meas.*, 2011, pp. 305–314.

[6] G. R. Bergersen, D. I. K. Sjøberg, and T. Dybå, "Construction and validation of an instrument for measuring programming skill," *IEEE Trans. Softw. Eng.*, vol. 40, no. 12, pp. 1163–1184, Dec. 2014.

[7] G. R. Bergersen, "Measuring programming skill–construction and validation of an instrument for evaluating java developers," Ph.D. dissertation, Dept. Inform., Univ. Oslo, Oslo, Norway, 2015.

[8] G. R. Bergersen and J.-E. Gustafsson, "Programming skill, knowledge, and working memory among professional software developers from an investment theory perspective," *J. Individual Differences*, vol. 32, no. 4, pp. 201–209, 2011.

[9] D. Borsboom, G. J. Mellenbergh, and J. Van Heerden, "The concept of validity," *Psychol. Rev.*, vol. 111, no. 4, pp. 1061–1071, 2004.

[10] L. Briand, K. El Emam, and S. Morasca, "Theoretical and empirical validation of software product measures," Int. Softw. Eng. Res. Netw. Tech. Rep., 1995, pp. 1–23.

[11] 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., New York, NY, USA: Taylor, 1996, pp. 189–194.

[12] A. Bryman and D. Cramer, *Quantitative Data Analysis With SPSS Release 10 for Windows: A Guide for Social Scientists*. Evanston, IL, USA: Routledge, 2002.

[13] B. Buckingham, "Intelligence and its measurement: A symposium. XIV," *J. Educ. Psychol.*, vol. 12, no. 5, pp. 271–275, 1921.

[14] D. T. Campbell and D. W. Fiske, "Convergent and discriminant validation by the multitrait-multimethod matrix," *Psychol. Bull.*, vol. 56, no. 2, pp. 81–105, 1959.

[15] R. B. Cattell, "Personality structure and measurement," *Brit. J. Psychol. Gen. Sect.*, vol. 36, no. 3, pp. 159–174, 1946.

[16] C. H. Cherryholmes, "Construct validity and the discourses of research," *Amer. J. Educ.*, vol. 96, no. 3, pp. 421–457, 1988.

[17] B. P. Cohen, *Developing Sociological Knowledge: Theory and Method*, vol. 1123. Belmont, CA, USA: Wadsworth, 1989.

[18] T. Coltman, T. M. Devinney, D. F. Midgley, and S. Venaik, "Formative versus reflective measurement models: Two applications of formative measurement," *J. Bus. Res.*, vol. 61, no. 12, pp. 1250–1262, 2008.

[19] 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*. Akron, Ohio: Goodyear Publishing Company, 1979.

[20] S. Courtis, "Report of the standardization committee," J. Educ. Res., vol. 4, no. 1, pp. 78–90, 1921.

[21] L. J. Cronbach and P. E. Meehl, "Construct validity in psychological tests," *Psychol. Bull.*, vol. 52, no. 4, pp. 281–302, 1955.

[22] L. Cronbach, *Essentials of Psychological Testing*. New York, NY, USA: Harper& Row, 1984.

[23] B. Curtis, "Measurement and experimentation in software engineering," *Proc. IEEE*, vol. 68, no. 9, pp. 1144–1157, 1980.

[24] O.-J. Dahl and K. Nygaard, "SIMULA: An algol-based simulation language," *Commun. ACM*, vol. 9, no. 9, pp. 671–678, 1966.

[25] F. D. Davis, "Perceived usefulness, perceived ease of use, and user acceptance of information technology," *MIS Quart.*, vol. 13, no. 3, pp. 319–340, 1989.

[26] F. Deissenboeck and M. Pizka, "Concise and consistent naming," *Softw. Qual. J.*, vol. 14, no. 3, pp. 261–282, 2006.

[27] T. Dybå, N. B. Moe, and E. Arisholm, "Measuring software methodology usage: Challenges of conceptualization and operationalization," in *Proc. IEEE Int. Symp. Empirical Softw. Eng.*, 2005, pp. 447–457.

[28] T. Dybå, V. B. Kampenes, and D. I. Sjøberg, "A systematic review of statistical power in software engineering experiments," *Inf. Softw. Technol.*, vol. 48, no. 8, pp. 745–755, 2006.

[29] A. Einstein, "Remarks concerning the essays brought together in this co-operative volume," *Albert Einstein: Philosopher-Scientist 2*, vol. 2, pp. 665–688, 1949.

[30] K. A. Ericsson, "The influence of experience and deliberate practice on the development of superior expert performance," in *The Cambridge Handbook of Expertise and Expert Performance*, K. A. Ericsson, N. Charness, P. J. Feltovich, and R. R. Hoffman, Eds., ch. 38. New York, NY, USA: Cambridge Univ. Press, 2006, pp. 685–705.

[31] R. Feldt *et al.*, "Four commentaries on the use of students and professionals in empirical software engineering experiments," *Empirical Softw. Eng.*, vol. 23, no. 6, pp. 3801–3820, 2018.

[32] N. Fenton, "Software measurement: A necessary scientific basis," *IEEE Trans. Softw. Eng.*, vol. 20, no. 3, pp. 199–206, Mar. 1994.

[33] P. M. Fitts and M. I. Posner, "Human performance," *Cole Belmont, CA*, vol. 5, pp. 7–16, 1967.

[34] E. Gamma *et al.*, *Design Patterns: Elements of Reusable Object-Oriented Software*. Reading, MA, USA: Addison-Wesley, 1995.

[35] D. B. Grisaffe, "Questions about the ultimate question: Conceptual considerations in evaluating reichheld's net promoter score (NPS)," *J. Consum. Satisfaction Dissatisfaction Complaining Behav.*, vol. 20, pp. 36–53, 2007.

[36] J. E. Hannay, D. I. K. Sjøberg, and T. Dybå, "A systematic review of theory use in software engineering experiments," *IEEE Trans. Softw. Eng.*, vol. 33, no. 2, pp. 87–107, Feb. 2007.

[37] A. Hobson, *The Oxford Dictionary of Difficult Words*. New York, NY, USA: Oxford Univ. Press, 2004.

[38] K. G. Jöreskog, "A general approach to confirmatory maximum likelihood factor analysis," *Psychometrika*, vol. 34, no. 2, pp. 183–202, 1969.

[39] S. Kalyuga, "The expertise reversal effect," in *Managing Cognitive Load in Adaptive Multimedia Learning*. Hershey, PA, USA: IGI Global, 2009, pp. 58–80.

[40] V. B. Kampenes, T. Dybå, J. E. Hannay, and D. I. Sjøberg, "A systematic review of effect size in software engineering experiments," *Inf. Softw. Technol.*, vol. 49, no. 11–12, pp. 1073–1086, 2007.

[41] C. Kaner and W. P. Bond, "Software engineering metrics: What do they measure and how do we know?," in *Proc. IEEE 10th Int. Softw. Metrics Symp.*, 2004, pp. 1–12.

[42] T. Kelly, *Interpretation of Educational Measurements Yonkers*, New York, NY, USA: World Book, pp. 62–65, 1927.

[43] J. Kirk and M. L. Miller, *Reliability and Validity in Qualitative Research*, vol. 1. Newbury Park, CA, USA: Sage, 1986.

[44] B. Kitchenham, S. L. Pfleeger, and N. Fenton, "Towards a framework for software measurement validation," *IEEE Trans. Softw. Eng.*, vol. 21, no. 12, pp. 929–944, Dec. 1995.

[45] B. Kitchenham *et al.*, "Trends in the quality of human-centric software engineering experiments—a quasi-experiment," *IEEE Trans. Softw. Eng.*, vol. 39, no. 7, pp. 1002–1017, Jul. 2013.

[46] B. A. Kitchenham, D. Budgen, and P. Brereton, *Evidence-Based Software Engineering and Systematic Reviews*, vol. 4. Boca Raton, FL, USA: CRC Press, 2015.

[47] D. H. Krantz, R. D. Luce, P. Suppes, and A. Tversky, *Foundations of Measurement, Vol. I: Additive and Polynomial Representations*, New York, NY, USA: Academic Press, 1971.

[48] B. P. Lientz, E. B. Swanson, and G. E. Tompkins, "Characteristics of application software maintenance," *Commun. ACM*, vol. 21, no. 6, pp. 466–471, 1978.

[49] M. M. Mark, *Realism, Validity, and the Experimenting Society*, vol. 1. Newbury Park, CA, USA: Sage, 2000, pp. 141–166.

[50] G. Maruyama and C. S. Ryan, *Research Methods in Social Relations*. Hoboken, NJ, USA: Wiley, 2014.

[51] J. Maxwell, "Understanding and validity in qualitative research," *Harvard Educ. Rev.*, vol. 62, no. 3, pp. 279–301, 1992.

[52] J. A. Maxwell, "Using qualitative methods for causal explanation," *Field Methods*, vol. 16, no. 3, pp. 243–264, 2004.

[53] T. J. McCabe, "A complexity measure," *IEEE Trans. Softw. Eng.*, no. 4, pp. 308–320, Apr. 1976.

[54] S. Messick, "Validity," *Educ. Meas.*, vol. 1989, pp. 13–103, 1989.

[55] N. Miller, W. C. Pedersen, and V. E. Pollock, *Discriminative Validity*, vol. 1, Newbury Park, CA, USA: Sage, 2000, pp. 65–99.

[56] A. A. Neto and T. Conte, "A conceptual model to address threats to validity in controlled experiments," in *Proc. 17th Int. Conf. Eval. Assessment Softw. Eng.*, 2013, pp. 82–85.

[57] J. C. Nunnally and I. H. Bernstein, *Psychometric Theory*, 3rd ed. New York, NY, USA: New York: McGraw-Hill, 1994.

[58] P. Oman and J. Hagemeister, "Metrics for assessing a software system's maintainability," in *Proc. Conf. Softw. Maintenance*, 1992, pp. 337–338.

[59] P. Panayides, "Coefficient alpha: Interpret with caution," *Eur. J. Psychol.*, vol. 9, no. 4, pp. 687–696, 2013.

[60] M. Q. Patton, *How to Use Qualitative Methods in Evaluation*. Newbury Park, CA, USA: Sage, 1987.

[61] L. Prechelt, B. Unger, W. F. Tichy, P. Brossler, and L. G. Votta, "A controlled experiment in maintenance: Comparing design patterns to simpler solutions," *IEEE Trans. Softw. Eng.*, vol. 27, no. 12, pp. 1134–1144, Dec. 2001.

[62] P. Ralph and E. Tempero, "Construct validity in software engineering research and software metrics," in *Proc. 22nd Int. Conf. Eval. Assessment Softw. Eng.*, 2018, pp. 13–23.

[63] F. S. Roberts, *Measurement Theory*. New York, NY, USA: Cambridge Univ. Press, 1985.

[64] T. B. Rogers, *The Psychological Testing Enterprise: An Introduction*. California, USA: Thomson Brooks/Cole, 1995.

[65] M. J. Rosenberg, "The conditions and consequences of evaluation apprehension," in *Artifact in Behavioral Research*, R. Rosenthal and R. L. Rosnow, Eds., San Francisco, CA, USA: Academic Press, 1969, pp. 279–349.

[66] R. Rosenthal, *Experimenter Effects in Behavioral Research*. Irvington, New Jersey: Irvington, 1976.

[67] P. Runeson and M. Höst, "Guidelines for conducting and reporting case study research in software engineering," *Empirical Softw. Eng.*, vol. 14, no. 2, pp. 131–164, 2009.

[68] P. Runeson, M. Host, A. Rainer, and B. Regnell, *Case Study Research in Software Engineering: Guidelines and Examples*. New York, NY, USA: Wiley, 2012.

[69] P. J. Runkel and J. E. McGrath, *Research on Human Behavior: A Systematic Guide to Method*. New York, NY, USA: Holt, Rinehart & Winston, 1972.

[70] W. R. Shadish, T. D. Cook, and D. T. Campbell, *Experimental and Quasi-Experimental Designs for Generalized Causal Inference*. Boston, MA, USA: Houghton Mifflin, 2002.

[71] J. Siegmund, C. Kästner, J. Liebig, S. Apel, and S. Hanenberg, "Measuring and modeling programming experience," *Empirical Softw. Eng.*, vol. 19, no. 5, pp. 1299–1334, 2014.

[72] 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., Berlin, Germany: Springer, 2008, pp. 312–336.

[73] D. I. K. Sjøberg, B. Anda, and A. Mockus, "Questioning software maintenance metrics: A comparative case study," in *Proc. ACM-IEEE Int. Symp. Empirical Softw. Eng. Meas.*, 2012, pp. 107–110.

[74] D. I. K. Sjøberg, G. R. Bergersen, and T. Dybå, "Why theory matters," in *Perspectives on Data Science for Software Engineering*. New York, NY, USA: Elsevier, 2016, pp. 29–33.

[75] D. I. K. Sjøberg, T. Dybå, and M. Jørgensen, "The future of empirical methods in software engineering research," in *Proc. IEEE Future Softw. Eng.*, 2007, pp. 358–378.

[76] D. I. K. Sjøberg *et al.*, "A survey of controlled experiments in software engineering," *IEEE Trans. Softw. Eng.*, vol. 31, no. 9, pp. 733–753, Sep. 2005.

[77] D. I. K. Sjøberg, A. Johnsen, and J. Solberg, "Quantifying the effect of using kanban versus scrum: A case study," *IEEE Softw.*, vol. 29, no. 5, pp. 47–53, Sep./Oct. 2012.

[78] M. J. Strube, L. C. Newman, A. N. Lord, and P. L. Nguyen, *Psychometrics*, 4th ed., ser. New York, NY, USA: Cambridge Univ. Press, 2016, pp. 612–627.

[79] F. Suppe, *The Structure of Scientific Theories*. Urbana, IL, USA: Univ. of Illinois Press, 1974.

[80] W. M. Trochim and J. P. Donnelly, *Research Methods Knowledge Base*, vol. 2. Ohio, USA: Atomic Dog Publ., 2001.

[81] W. Trochim, J. Donnelly, and K. Arora, *Research Methods: The Essential Knowledge Base, 2nd ed.*. Boston, MA, USA: Cengage Learning, 2016.

[82] M. Vokáč, W. Tichy, D. I. Sjøberg, E. Arisholm, and M. Aldrin, "A controlled experiment comparing the maintainability of programs designed with and without design patterns—a replication in a real programming environment," *Empirical Softw. Eng.*, vol. 9, no. 3, pp. 149–195, 2004.

[83] C. Wijayarathna and N. A. G. Arachchilage, "Using cognitive dimensions to evaluate the usability of security APIs: An empirical investigation," *Inf. Softw. Technol.*, vol. 115, pp. 5–19, 2019.
[84] C. Wohlin, P. Runeson, M. Höst, M. C. Ohlsson, B. Regnell, and A. Wesslén, "Experimentation in software engineering: An introduction," *The Kluwer Int. Ser. Softw. Eng.*, pp. 1–204, 2000.
[85] C. Wohlin, P. Runeson, M. Höst, M. C. Ohlsson, B. Regnell and A. Wesslén, *Experimentation in Software Engineering*. Berlin, Germany: Springer, 2012.
[86] R. K. Yin, *Case Study Research: Design and Methods.*, 3rd ed. Newbury Park, CA, USA: Sage, 2003.
[87] R. K. Yin, *Case Study Research and Applications: Design and Methods.*, 6th ed. Newbury Park, CA, USA: Sage, 2018.

ARTICLES STUDIED

[A1] S. Abrah ao, 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," *Inf. Softw. Technol.*, vol. 116, 2019, Art. no. 106171.
[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 Softw. Eng.*, 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," *Inf. Softw. Technol.*, 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 Trans. Softw. Eng.*, vol. 44, no. 4, pp. 365–383, Apr. 2018.
[A5] A. M. Aranda, O. Dieste, and N. Juristo, "Effect of domain knowledge on elicitation effectiveness: An internally replicated controlled experiment," *IEEE Trans. Softw. Eng.*, vol. 42, no. 5, pp. 427–451, May 2016.
[A6] V. Arnaoudova, M. Di Penta, and G. Antoniol, "Linguistic antipatterns: What they are and how developers perceive them," *Empirical Softw. Eng.*, 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 Softw. Eng.*, 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 Softw. Eng.*, vol. 24, no. 4, pp. 1762–1798, 2019.
[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," *J. Syst. Softw.*, vol. 136, pp. 153–172, 2018.
[A10] G. Borrego, A. L. Morán, R. R. Palacio, A. Vizcaíno, and F. O. García, "Towards a reduction in architectural knowledge vaporization during agile global software development," *Inf. Softw. Technol.*, 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 Trans. Softw. Eng.*, vol. 42, no. 9, pp. 886–898, Sep. 2016.
[A12] C. Cachero, S. Meliá, and J. M. Hermida, "Impact of model notations on the productivity of domain modelling: An empirical study," *Inf. Softw. Technol.*, 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," *J. Syst. Softw.*, vol. 144, 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," *Inf. Softw. Technol.*, 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 Trans. Softw. Eng. Methodol.*, 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 Trans. Softw. Eng.*, vol. 41, no. 3, pp. 241–263, Mar. 2015.
[A17] J. Cunha, J. Paulo Fernandes, P. Martins, J. Mendes, R. Pereira, and J. Saraiva, "Evaluating refactorings for spreadsheet models," *J. Syst. Softw.*, vol. 118, pp. 234–250, 2016.
[A18] C. Czepa and U. Zdun, "How understandable are pattern-based behavioral constraints for novice software designers?," *ACM Trans. Softw. Eng. Methodol.*, 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," *Inf. Softw. Technol.*, vol. 110, pp. 3–16, 2019.
[A20] O. Dieste *et al.*, "Empirical evaluation of the effects of experience on code quality and programmer productivity: An exploratory study," *Empirical Softw. Eng.*, vol. 22, no. 5, pp. 2457–2542, 2017.
[A21] M. El-Attar, "Evaluating and empirically improving the visual syntax of use case diagrams," *J. Syst. Softw.*, 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 Trans. Softw. Eng.*, vol. 41, no. 7, pp. 661–690, Jul. 2015.
[A23] M. Felderer and A. Herrmann, "Manual test case derivation from UML activity diagrams and state machines: A controlled experiment," *Inf. Softw. Technol.*, 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," *Inf. Softw. Technol.*, 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 Softw. Eng.*, vol. 21, no. 1, pp. 212–259, 2016.
[A26] A. Fernandez and A. Bergel, "A domain-specific language to visualize software evolution," *Inf. Softw. Technol.*, 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 Trans. Softw. Eng. Methodol.*, 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 development skills: An industrial empirical study," *Inf. Softw. Technol.*, vol. 68, pp. 82–97, 2015.
[A29] H. Ghanbari, J. Similä, and J. Markkula, "Utilizing online serious games to facilitate distributed requirements elicitation," *J. Syst. Softw.*, vol. 109, pp. 32–49, 2015.
[A30] C. Hannebauer, M. Hesenius, and V. Gruhn, "Does syntax highlighting help programming novices?," *Empirical Softw. Eng.*, 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," *Inf. Softw. Technol.*, vol. 79, pp. 52–62, 2016.
[A32] I. U. Hassan, N. Ahmad, and B. Zuhaira, "Calculating completeness of software project scope definition," *Inf. Softw. Technol.*, vol. 94, pp. 208–233, 2018.
[A33] N. Itzik, I. Reinhartz-Berger, and Y. Wand, "Variability analysis of requirements: Considering behavioral differences and reflecting stakeholders' perspectives," *IEEE Trans. Softw. Eng.*, vol. 42, no. 7, pp. 687–706, Jul. 2015.
[A34] K. Jaber, B. Sharif, and C. Liu, "An empirical study on the effect of 3D visualization for project tasks and resources," *J. Syst. Softw.*, 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," *J. Syst. Softw.*, 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 Softw. Eng.*, 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," *Inf. Softw. Technol.*, 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 Softw. Eng.*, vol. 23, no. 5, pp. 2734–2763, 2018.

[A39] J. L. Krein *et al.*, "A multi-site joint replication of a design patterns experiment using moderator variables to generalize across contexts," *IEEE Trans. Softw. Eng.*, vol. 42, no. 4, pp. 302–321, Apr. 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 versus graphical representations," *Empirical Softw. Eng.*, 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 Softw. Eng.*, 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," *J. Syst. Softw.*, vol. 137, pp. 497–511, 2018.
[A43] I. Lytra, P. Gaubatz, and U. Zdun, "Two controlled experiments on model-based architectural decision making," *Inf. Softw. Technol.*, 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 Softw. Eng.*, vol. 20, no. 2, pp. 413–441, 2015.
[A45] S. Makady and R. J. Walker, "Debugging and maintaining pragmatically reused test suites," *Inf. Softw. Technol.*, 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," *Inf. Softw. Technol.*, 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," *J. Syst. Softw.*, vol. 100, pp. 1–14, 2015.
[A48] J. M. Morales, E. Navarro, P. Sánchez, and D. Alonso, "A family of experiments to evaluate the understandability of tristar and I for modeling teleo-reactive systems," *J. Syst. Softw.*, 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 Trans. Softw. Eng.*, vol. 43, no. 2, pp. 106–127, Feb. 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 Softw. Eng.*, 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 Softw. Eng.*, 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," *J. Syst. Softw.*, 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 Trans. Softw. Eng.*, vol. 44, no. 10, pp. 977–1000, Oct. 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," *Inf. Softw. Technol.*, 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," *Inf. Softw. Technol.*, 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 Softw. Eng.*, vol. 21, no. 5, pp. 2190–2231, 2016.
[A57] A. Przybyłek, "An empirical study on the impact of aspectj on software evolvability," *Empirical Softw. Eng. Int. J.*, vol. 23, no. 4, pp. 2018–2050, 2017.
[A58] I. Reinhartz-Berger, K. Figl, and Ø. Haugen, "Investigating styles in variability modeling: Hierarchical versus constrained styles," *Inf. Softw. Technol.*, vol. 87, pp. 81–102, 2017.
[A59] M. Riaz *et al.*, "Identifying the implied: Findings from three differentiated replications on the use of security requirements templates," *Empirical Softw. Eng.*, 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 Softw. Eng.*, 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 Trans. Softw. Eng. Methodol.*, vol. 24, no. 1, pp. 1–38, 2014.
[A62] E. Macedo Rodrigues *et al.*, "An empirical comparison of model-based and capture and replay approaches for performance testing," *Empirical Softw. Eng.*, 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," *Inf. Softw. Technol.*, 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," *J. Syst. Softw.*, 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," *J. Syst. Softw.*, vol. 109, pp. 124–136, 2015.
[A66] L. Sabatucci, M. Cossentino, and A. Susi, "A goal-oriented approach for representing and using design patterns," *J. Syst. Softw.*, 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 Softw. Eng.*, 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," *Inf. Softw. Technol.*, 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 Softw. Eng.*, 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 Softw. Eng.*, 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," *J. Syst. Softw.*, 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 Softw. Eng.*, 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 Trans. Softw. Eng. Methodol.*, 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 Softw. Eng.*, 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," *J. Syst. Softw.*, 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," *Inf. Softw. Technol.*, vol. 104, pp. 179–193, 2018.
[A77] S. Tiwari and A. Gupta, "Investigating comprehension and learnability aspects of use cases for software specification problems," *Inf. Softw. Technol.*, vol. 91, pp. 22–43, 2017.
[A78] A. Tosun *et al.*, "An industry experiment on the effects of test-driven development on external quality and productivity," *Empirical Softw. Eng.*, 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," *Inf. Softw. Technol.*, 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," *Inf. Softw. Technol.*, vol. 116, 2019, Art. no. 106169.

[A81] Y.-C. Tu, E. Tempero, and C. Thomborson, "An experiment on the impact of transparency on the effectiveness of requirements documents," *Empirical Softw. Eng.*, 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 Softw. Eng.*, 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," *Inf. Softw. Technol.*, vol. 106, pp. 68–84, 2019.

Dag I. K. Sjøberg (Member, IEEE) received the MSc degree in computer science from the University of Oslo, in 1987, and the PhD degree in computing science from the University of Glasgow, in 1993. He has five years of industry experience as a developer and group leader. In 2001, he formed the Software Engineering Department, Simula Research Laboratory and was its leader until 2008. Since 1999, he has been a full professor of software engineering with the University of Oslo. His main research interests include the software life cycle, including agile and lean development processes, and theories and empirical research methods in software engineering.

Gunnar Rye Bergersen received the Cand. Scient. and PhD degrees in computer science from the University of Oslo, in 2001 and 2015, respectively. His PhD work, which was partially completed with the Simula Research Laboratory, has later been used commercially in a start-up. He has more than 20 years of industry experience as a software developer, project manager, product manager, and CEO. Since 2022, he has been an associate professor with the University of Oslo. His main research interests include quantitative research methods with a particular focus on individual and team capabilities and their measurement.