# Apples, Oranges, and Software Engineering: Study Selection Challenges for Secondary Research on Latent Variables

Marvin Wyrich
Saarland University
Saarbrücken, Germany
wyrich@cs.uni-saarland.de

Marvin Muñoz Barón
University of Stuttgart
Stuttgart, Germany
marvin.munoz-baron@iste.uni-stuttgart.de

Justus Bogner
Vrije Universiteit Amsterdam
Amsterdam, The Netherlands
j.bogner@vu.nl

## ABSTRACT

Software engineering (SE) is full of abstract concepts that are crucial for both researchers and practitioners, such as programming experience, team productivity, code comprehension, and system security. Secondary studies aimed at summarizing research on the influences and consequences of such concepts would therefore be of great value.

However, the inability to measure abstract concepts directly poses a challenge for secondary studies: primary studies in SE can operationalize such concepts in many ways. Standardized measurement instruments are rarely available, and even if they are, many researchers do not use them or do not even provide a definition for the studied concept. SE researchers conducting secondary studies therefore have to decide a) which primary studies intended to measure the same construct, and b) how to compare and aggregate vastly different measurements for the same construct.

In this experience report, we discuss the challenge of study selection in SE secondary research on latent variables. We report on two instances where we found it particularly challenging to decide which primary studies should be included for comparison and synthesis, so as not to end up comparing apples with oranges. Our report aims to spark a conversation about developing strategies to address this issue systematically and pave the way for more efficient and rigorous secondary studies in software engineering.

## CCS CONCEPTS

• **General and reference → Surveys and overviews**.

## KEYWORDS

Secondary research, concepts, constructs, unobserved variables, experience report

**ACM Reference Format:**
Marvin Wyrich, Marvin Muñoz Barón, and Justus Bogner. 2024. Apples, Oranges, and Software Engineering: Study Selection Challenges for Secondary Research on Latent Variables. In *International Workshop on Methodological Issues with Empirical Studies in Software Engineering (WSESE '24), April 16, 2024, Lisbon, Portugal.* ACM, New York, NY, USA, 6 pages. https://doi.org/10.1145/3643664.3648213


*WSESE '24, April 16, 2024, Lisbon, Portugal*

ACM ISBN 979-8-4007-0567-0/24/04
https://doi.org/10.1145/3643664.3648213

## 1 INTRODUCTION

*The only valid measurement of code quality: WTFs per minute.* While one can easily smile at the comic by Thom Holwerda [19], in which the number of swear words used during a code review decides whether code is good or bad, the comic highlights a real challenge that is as old as the software engineering field itself. Different people have different ideas about what code quality is and how best to measure it. This is because code quality is a *construct*, i.e. "a concept that is not directly measurable but is represented by indicators at the operational level to make it measurable" [43].

Software engineering (SE) is full of such constructs. They can enable everyone involved to communicate efficiently on an abstract level, e.g., about a team being more *productive* than in the previous sprint, a software system being sufficiently *secure*, or certain parts of the code not yet being *understandable* enough. Since all of these constructs can be relevant in practice, SE research is concerned with them: primary studies investigate how software teams can become more productive, which security vulnerabilities make a system no longer secure, and what has a positive and negative effect on the understandability of code. Secondary studies then summarize the results of related primary studies to inform research and practice about available evidence—or at least that is how it is theoretically meant to be.

Secondary research often faces a challenge: abstract concepts can have very different meanings in the minds of different researchers. And, to date, primary research in software engineering still rarely defines its investigated concepts [45, 48]. An implicit definition by task and measure prevails, but the task and measure may be different in each primary study. This makes it difficult for someone conducting a literature review to decide which primary study to include to still ensure a meaningful comparison on the conceptual level. We will illustrate the issue with two examples from our experience with secondary studies in the field of code comprehension.

### 1.1 Case 1: An Attempt at Meta-Analysis

Many attempts were made to find a metric that automatically measures code understandability. A few years ago, we empirically validated such a metric, i.e., we investigated whether a metric proposed by the industry really measures code understandability as it claims [32]. However, we did not conduct our own experiment. Instead, we used data sets from ten existing code comprehensibility studies, in which the comprehensibility of certain code snippets had already been measured with human participants.

We obtained around 24,000 understandability evaluations of 427 code snippets, far more data than we could have collected in a single study. We were then able to correlate the understandability

evaluations with the automated metric values. This way, we knew how well the proposed metric correlated with the human-collected understandability evaluations within each study.

In a final step, we planned to statistically summarize the correlation coefficients obtained from each study in a meta-analysis. And here we were faced with a difficult decision. Only at this point did we realize how differently primary studies actually measure code comprehensibility: starting from the time needed to calculate a return value, over the correctness to comprehension questions, all the way to the strength of brain deactivation measured as a proxy for concentration through the usage of an fMRI scanner. Statistically, it would have been possible to combine all this data, but would it have made sense?

At the time, we decided against a holistic analysis and instead divided our research question into five sub-questions, each of which was to investigate the correlation of the metric with a specific way of measuring code comprehensibility. To answer each of the sub-questions, we then conducted a separate meta-analysis and created forest plots. Then, as now, we believe that "while it is a tempting thought to combine the types of variables rather than dividing each of them into separate research questions, in our case the lack of information on the relationships and validity of these measures prevented us from doing so" [32].

Now, we still needed an answer to whether the metric correlated with code understandability. The somewhat sobering answer is: it depends. The metric correlated with comprehension time and subjective ratings of understandability. It correlated less well with the correctness of comprehension tasks and physiological measures. Although these were interesting results, we were a little dissatisfied with the situation, as splitting the research into five sub-questions is almost guaranteed to produce an "it depends" result. Would we have been comparing apples with oranges if we had thrown all the studies into one bowl? Our results suggest so, but we would only have certainty if we specifically investigated whether all these metrics measured the same construct.

### 1.2 Case 2: A Systematic Mapping Study

The diversity of study designs that seemed to be present in code comprehension studies made us curious. We therefore conducted a systematic mapping study to take a closer look at the design landscape of 95 source code comprehension experiments published between 1979 and 2019 [48]. However, identifying these 95 studies in the first place was challenging, which can be explained in part by the results of the mapping study: each study in our dataset was unique in its design, they all used different tasks and metrics to measure a study participant's code comprehension, and there was no homogeneity in the naming of the construct of interest. For example, we also examined studies that did not refer to comprehension but to *readability*, *task difficulty*, or *mental effort*. Notably, hardly any of the 95 studies provided a definition of these constructs.

Thus, we were faced with the challenge of speculating about the intention of the authors of the primary studies based on the methodology of the studies: do they all intend to measure the same construct, although they sometimes call it differently, and they all operationalize it somewhat differently? We read all the papers carefully and decided whether to include a primary study or not to the best of our knowledge and ability. This was more time-consuming and involved more uncertainty than should have been necessary.

In both of these cases, we eventually found a way to address the challenge of comparing and synthesizing primary studies on code comprehension. For example, regarding case 2, we decided not to trust the used concept names by the authors, but instead carefully analyzed the experiment designs to find out if the authors really wanted to measure our studied construct, bottom-up source code comprehension from the perspective of human participants. This required considerably more time for extraction and discussion, with many instances of us angrily throwing our hands up in frustration, but ultimately, it led to much more consistent results. Nevertheless, it appeared as if we were fighting symptoms rather than a cause, and we did not see an easy way to define generalizable guidance for other researchers based on these experiences. Variability in study designs is desirable, as it contributes to a comprehensive understanding of a research question. It is also possible, in principle, to make a statistical comparison despite a diversity of study characteristics. However, a critical concern arises regarding the feasibility of meaningful comparisons across primary studies at a conceptual level. This challenge is particularly pronounced when studies operationalize variables in disparate ways without establishing a clear definition of the investigated constructs. In the remainder of this paper, we first review related literature on this topic (Section 2), which then forms the basis for our discussion of ways forward in the context of empirical software engineering research (Section 3).

## 2 RELATED WORK

We begin with an overview of the state of secondary studies in SE, followed by a glimpse beyond the borders of SE, incorporating insights from the fields of psychology and medicine.

### 2.1 Secondary Research in SE

Secondary studies in SE research have seen significant advancement since the first push towards evidence-based software engineering (EBSE) in the early 2000s [12]. Especially when it comes to systematic reviews and mapping studies, there has been a healthy number of publications [10, 11]. Systematic mapping studies have allowed SE researchers to gain quick overviews through the categorization and high-level aggregation of various topics. To complement this, in-depth systematic reviews then provide researchers with more profound insights into the available evidence about specific topics, often through qualitative synthesis [35]. Moreover, efforts have been made to establish concrete standards for conducting secondary studies [25, 27]. In addition to extensive methodological knowledge about reviews and mapping studies, these guidelines often also include chapters on quantitative synthesis. But, when it comes to quantitative secondary studies such as meta-analyses, only very few have been published in SE [11, 26].

So, why are there so few meta-analyses in our field? One possible explanation could lie in the pool of primary research suited for quantitative synthesis. In fact, difficulties in conducting meta-analysis in SE have been observed even before the advent of EBSE. At the turn of the millennium, Miller attempted to conduct a meta-analysis on defect detection experiments. He found that "the discipline must

embark upon a period of improvement to reduce the variability between replicated experiments" [30]. The issues brought up by Miller, such as the stark variability in study design, measurement, and reporting of SE experiments, have since been repeatedly found by others [2, 20, 22, 23]. Miller's investigation ended on a positive note, remarking that the field is still very young and things may improve with maturity. More than 20 years later, we still struggle with the same problems in our attempts at meta-analysis.

Variability in studies, especially in replications, is a desired trait. Study subjects, settings, materials should differ to provide robustness of the observed effect [24]. However, variability is undesired when it comes to fundamental understanding and definition of the measured concepts. If concepts or properties are only the same in name, but are understood differently by researchers, this will often be reflected in a variability in the method of measurement, experiment tasks, and measurement instruments. Often, researchers do not attempt to evaluate construct validity and do not question the measures they take, which might lead them to incorrect conclusions [36]. Further, variability in reporting may be addressed with more rigor in the adherence to reporting guidelines.

## 2.2 A Look at Other Fields of Research

Compared to the broader scientific community, software engineering is still a relatively young research area. This youth presents an opportunity to draw lessons from the rich history of other disciplines, which have faced similar challenges in the past. Specifically, the field of psychometrics [17], with its deep roots in measuring abstract concepts and latent variables, may offer valuable insights to SE researchers. In psychology, the relationship between human behavior and understanding its expression through quantifiable measures in tests has been discussed since 1886 [1].

As the years have passed since then, experimental psychologists have faced many of the same challenges that SE researchers still face today. The problem of multiple, different measures being used for the same abstract concept is a core challenge described in the literature on experimental psychology [1]. There, it is argued that the problem is a necessary consequence of the abstract nature of latent variables, as multiple instruments may measure the same psychological phenomenon. The conclusion to this problem, however, is not the indiscriminate usage of instruments, but rather that the evaluation of "the attributes of psychological testing is one of the greatest concerns of psychometrics" [1].

As a consequence, experimental psychologists have long applied systematic methods to evaluate the validity of psychometric instruments. Approaches such as classical test theory (CTT) [15], item response theory (IRT) [16], and confirmatory factor analysis (CFA) [4] could not only be applied when designing instruments to measure latent variables in SE experiments, but also be taught to aspiring SE researchers, as has been the case in experimental psychology for years [8]. In fact, when measuring latent variables such as comprehension, SE researchers assume the role of behavioral scientists, as "they identify some type of observable behavior that they think represents the particular unobservable psychological attribute, state, or process" [17]. One might even go so far as to say that measuring the human aspects of SE or conducting experiments about software development constitutes a psychological experiment, and thus, measuring the latent variables in SE is psychometrics. Controlled experiments in SE very closely mirror those in experimental psychology, but the maturity and rigor of measurement methodology in the latter field could greatly benefit the former. There have been attempts to bridge this gap and introduce these methods to SE researchers [18, 38], but, as of today, these discussions remain decidedly niche.

In psychometrics, measurement almost exclusively relates to latent human variables, which are measured through meticulously designed instruments. Software engineering, on the other hand, additionally involves the measurement of software characteristics that are directly measured, such as the number of lines of code. This difference, however, should not be seen as an excuse to omit discussions about the validity of software metrics and how they represent quality attributes. For latent human characteristics, we may directly use or adapt measurement instruments used in psychology, but for software metrics, the driving force behind their development is the SE community itself. Contrary to the lack of discussions about the validity of latent variables, discussions about the validity of software metrics have a much richer history in SE. As early as 1996, researchers have argued against the blind acceptance of traditional measurement theory in favor of pragmatism for software engineering [3]. The effect of this rejection is still felt today, with the topic of construct validity being remarkably absent in many software engineering papers, potentially leading to wrong conclusions [37]. We therefore argue that, rather than rejecting the approaches from other fields, the community should embrace them. For software metrics, methods and theories applied in psychology regarding construct validity could be particularly important in establishing empirical standards.

Notably, since the first advocacy for the evaluation of construct validity by Cronbach and Meehl in 1955 [9], psychology has noticed significant improvements in clinical assessment through critical evaluation of all aspects of the construct validity process [44]. Some even call it "one of the most important concepts in all of psychology" [46]. And while the object of measurement differs and the measured concepts are often more concrete, many of the methodological guidelines should be applied for both human- and software-focused constructs. Of particular importance is an explicit definition of the measured concept, and applying standardized tests and metrics to measure that concept, regardless of what is measured. Recent works defining guidelines on evaluating construct validity in software engineering can be seen as valuable steps toward supporting this effort [37, 42].

Furthermore, the method of meta-analysis and the challenges of synthesizing data from heterogeneous studies have been an important topic in modern medicine research [14]. The introduction of the random-effects model for the meta-analysis of clinical trials was intended to account for and explain the heterogeneity of studies due to inter-study differences in the employed methods or patient characteristics [13]. Since then, the use of meta-analysis has been standard procedure in medicine and public health research, leading to modern approaches such as meta-regression and extensions of the random-effects model for multivariate meta-analysis [14]. In addition to sophisticated statistical methods of synthesis, medical research exemplifies the adherence to strict methodological guidelines such as those outlined in the CONSORT statement [31],

and standardized terminology and measures that are published in various reporting guidelines [41]. Moreover, medicine has a larger focus on replication studies, further expanding the pool of data for established methodologies. In software engineering, the opposite is the case: replication studies and meta-analyses are exceedingly scarce [7, 21], and while a few standards and guidelines for measurement and experiment design exist, they are not universally followed.

## 3 WAYS FORWARD

It is not a novel finding that constructs exist in SE research and that they are measured in different ways. Indeed, our community seems even so aware of them that there are guidelines written for SE research to ensure construct validity [37, 43], which should lead to meaningful and valid operationalization. Furthermore, we were not the first to carry out a secondary study and, in fact, there are now guidelines for selecting an appropriate type of secondary research in software engineering as well [36]. Practically speaking, however, the devil is in the detail when conducting a secondary study on certain constructs. For example, we experienced the aforementioned challenge of deciding which primary studies are even comparable with each other on a conceptual level. Based on our experience and what we can learn from other research fields, we see two concrete action items for the way forward that can make SE research more effective and efficient in the future.

### 3.1 Define, Model, and Discuss Constructs

We need clarity about the construct under investigation already during the design and execution of a primary study. Treating this as an afterthought during the study reporting is *not* an acceptable practice. When using established constructs from other fields, such as personality and intelligence, definitions can usually be found and cited. In some cases, these constructs are based not only on dictionary-like definitions, but also on entire models that semantically describe the construct. In the case of personality, e.g., one could refer to the Five-Factor Model [29], and for intelligence, one could build on Carroll's Three-Stratum theory [6], to explore a research question. If two studies cited and adhered to the same definition or model, it should be straightforward to argue that they are conceptually comparable.

Now, in the fast-paced field of software engineering, we do not always have the luxury of relying on established and well-defined constructs. Sometimes, a construct in a research field may be well-defined, but it has a slightly different meaning in the SE context. For example, text comprehension is a slightly older construct than code comprehension, and the code comprehension research drew on some ideas from its ancestor during its inception [33]. However, conceptually, according to the current state of research, we are well-advised to define code and text comprehension as distinct constructs [5, 28, 34]. So, what do we do when such a definition is lacking?

Our suggestion is to propose missing construct definitions ourselves. This can result from a community effort, such as a workshop where experts on a construct participate. It can also arise from a public debate, in which individual researchers or research groups engage in discourse and articulate their proposals in position papers. We have observed that there is insufficient exchange among SE researchers regarding the semantics of abstract concepts, leading to uncertainty in the design of studies. Especially when there are no standardized tests, designing a primary study becomes a challenge in justifying one's operationalization of a construct if the construct has not been defined beforehand. Motivated by this, Wyrich [47] recently proposed a definition and a conceptual model for the concept of *source code comprehension*. Secondary studies could make it a specific inclusion criterion that a primary study must be anchored in such a conceptual model to ensure comparability of included primary studies. We hope for more initiatives like these and advocate for a space for construct discussion within the respective research communities.

### 3.2 Standardize and Validate Instruments

Definitions help articulate the intentions of researchers, and we would appreciate it if each primary study defines its constructs. However, from the perspective of primary researchers, it remains laborious to develop a test for the construct under investigation for each study and potentially have to validate it as well. Therefore, the presence of standardized and validated tests, as known in psychology, would be a relief for primary research in SE. Secondary studies, in turn, could make use of a specific standardized test in primary studies as a criterion for inclusion to ensure comparability between primary studies. In some cases, there may still be the issue of an implicit definition of the construct through the tasks of such a test, but at least all included primary studies intended to measure *something* in the exact same way.

An example from SE is the work of Siegmund et al. [40] on measuring *programming experience*. They developed a model of programming experience, designed a questionnaire, and followed a psychometrics approach to validate the questionnaire with computer science students. The work represents, according to the authors themselves, just the beginning of a larger effort to create a reliable and reusable test instrument for measuring programming experience. When the original conference article won the most influential paper award (MIP) at ICPC 2022, in the MIP Talk [39], Siegmund drew a somewhat sobering conclusion, stating that the test has not been further developed by either the authors or the community. One reason mentioned by Siegmund was that the validation of a measurement instrument often takes several years, and this is rarely appreciated by appointment or PhD committees. Another reason for the slow progress is that useful datasets exist but are not accessible. While many studies already build on the work of Siegmund et al. [40], these studies do not publish their own research data that could be used for further validation of the test instrument [39].

To make standardized measurement instruments for frequently studied constructs in the SE context a reality, a paradigm shift is needed within our research community. And there are some reasons to be optimistic about our methodological future. For example, SIGSOFT promotes its open science policies, e.g., at the International Conference on Software Engineering (ICSE), and several SE conferences now have dedicated replication tracks. It is our hope that this increasing recognition of open data and replications will

over time increase the availability of data for validating measurement instruments in SE. At the same time, SE researchers need to be educated about the importance of construct validity and the significant value of validated measurement instruments. While it is completely normal in other disciplines to spend one or even several PhD projects on the development of a reliable and valid measurement instrument, a similar project would still raise the eyebrows of many SE researchers. In peer review, there is currently often not even a minimum requirement for construct validation on the part of the study authors. Additionally, in the use of validated tests, we often observe researchers modifying them without being aware of the consequences for construct validity. This is an indicator that guidelines and introductory articles on the subject are needed [18, 37].

## 4 CONCLUSION

When comparing primary studies on abstract concepts, there is a risk of literally comparing apples to oranges. In our opinion, there is nothing wrong with a fresh fruit salad. However, in SE research, it is currently challenging to discern the variety of fruits mixed in the bowl. Often, definitions for constructs are lacking, different terms are used interchangeably, and for hardly any construct, there exists a validated, reusable measurement instrument.

We have reported on two instances where the current situation posed challenges for us. Deciding which primary studies to select for a secondary study currently is, on a conceptual level, akin to a matter of intuition and laborious interpretative work. Related work from other fields has shown us that this situation can be improved, and that SE research can evolve methodologically.

Our resulting two-step guide to success can be summarized as follows: 1. Discuss, define, and model constructs within the respective research community, 2. Standardize and validate the measurement of these constructs. The outcome should benefit everyone involved in SE research: authors of primary studies, researchers conducting secondary studies, reviewers of both types of papers, and of course readers of SE papers.

## REFERENCES


[1] Luis Anunciacao. 2018. An Overview of the History and Methodological Aspects of Psychometrics: History and Methodological Aspects of Psychometrics. *Journal for ReAttach Therapy and Developmental Diversities* 1, 1 (2018), 44–58.

[2] Pearl Brereton, Barbara A. Kitchenham, David Budgen, Mark Turner, and Mohamed Khalil. 2007. Lessons from Applying the Systematic Literature Review Process within the Software Engineering Domain. *Journal of systems and software* 80, 4 (2007), 571–583.

[3] Lionel Briand, Khaled El Emam, and Sandro Morasca. 1996. On the Application of Measurement Theory in Software Engineering. *Empirical Software Engineering* 1, 1 (1996), 61–88. https://doi.org/10.1007/BF00125812

[4] Timothy A. Brown. 2015. *Confirmatory Factor Analysis for Applied Research.* Guilford publications.

[5] Teresa Busjahn, Roman Bednarik, Andrew Begel, Martha Crosby, James H Paterson, Carsten Schulte, Bonita Sharif, and Sascha Tamm. 2015. Eye movements in code reading: Relaxing the linear order. In *2015 IEEE 23rd International Conference on Program Comprehension.* IEEE, 255–265.

[6] John B Carroll. 2005. The three-stratum theory of cognitive abilities. *Contemporary intellectual assessment: theories, tests, and issues* (2005).

[7] Jeffrey C. Carver, Natalia Juristo, Maria Teresa Baldassarre, and Sira Vegas. 2014. Replications of Software Engineering Experiments. *Empirical Software Engineering* 19, 2 (April 2014), 267–276. https://doi.org/10.1007/s10664-013-9290-8

[8] Denis Cousineau. 2005. The Rise of Quantitative Methods in Psychology. *Tutorials in Quantitative Methods for Psychology* 1, 1 (2005), 1–3.

[9] Lee J. Cronbach and Paul E. Meehl. 1955. Construct Validity in Psychological Tests. *Psychological bulletin* 52, 4 (1955), 281.

[10] Daniela S. Cruzes and Tore Dybå. 2011. Research Synthesis in Software Engineering: A Tertiary Study. *Information and Software Technology* 53, 5 (2011), 440–455.

[11] Fabio QB Da Silva, André LM Santos, Sérgio Soares, A. César C. França, Cleviton VF Monteiro, and Felipe Farias Maciel. 2011. Six Years of Systematic Literature Reviews in Software Engineering: An Updated Tertiary Study. *Information and Software Technology* 53, 9 (2011), 899–913.

[12] Ray Dawson, Phil Bones, Briony J. Oates, Pearl Brereton, Motoei Azuma, and Mary Lou Jackson. 2003. Empirical Methodologies in Software Engineering. In *Eleventh Annual International Workshop on Software Technology and Engineering Practice.* IEEE, 52–58.

[13] Rebecca DerSimonian and Nan Laird. 1986. Meta-Analysis in Clinical Trials. *Controlled clinical trials* 7, 3 (1986), 177–188.

[14] Rebecca DerSimonian and Nan Laird. 2015. Meta-Analysis in Clinical Trials Revisited. *Contemporary clinical trials* 45 (2015), 139–145.

[15] Robert F. DeVellis. 2006. Classical Test Theory. *Medical care* (2006), S50–S59. jstor:41219505

[16] Susan E. Embretson and Steven P. Reise. 2013. *Item Response Theory.* Psychology Press.

[17] R. Michael Furr. 2021. *Psychometrics: An Introduction.* SAGE publications.

[18] Daniel Graziotin, Per Lenberg, Robert Feldt, and Stefan Wagner. 2021. Psychometrics in Behavioral Software Engineering: A Methodological Introduction with Guidelines. *ACM Trans. Softw. Eng. Methodol.* 31, 1, Article 7 (sep 2021), 36 pages. https://doi.org/10.1145/3469888

[19] Thom Holwerda. 2008. *WTFs/m.* https://web.archive.org/web/20231122170548/https://www.osnews.com/story/19266/wtfsm/

[20] Seyedrebvar Hosseini, Burak Turhan, and Dimuthu Gunarathna. 2017. A Systematic Literature Review and Meta-Analysis on Cross Project Defect Prediction. *IEEE Transactions on Software Engineering* 45, 2 (2017), 111–147.

[21] Letizia Jaccheri, Zamira Kholmatova, and Giancarlo Succi. 2021. Systematizing the Meta-Analytical Process in Software Engineering. In *2021 2nd European Symposium on Software Engineering.* ACM, Larissa Greece, 1–5. https://doi.org/10.1145/3501774.3501775

[22] Andreas Jedlitschka and Marcus Ciolkowski. 2004. Towards Evidence in Software Engineering. In *Proceedings. 2004 International Symposium on Empirical Software Engineering, 2004. ISESE'04.* IEEE, 261–270.

[23] Vigdis By Kampenes, Tore Dybå, Jo E. Hannay, and Dag IK Sjøberg. 2007. A Systematic Review of Effect Size in Software Engineering Experiments. *Information and Software Technology* 49, 11-12 (2007), 1073–1086.

[24] Barbara Kitchenham. 2008. The Role of Replications in Empirical Software Engineering—a Word of Warning. *Empirical Software Engineering* 13, 2 (April 2008), 219–221. https://doi.org/10.1007/s10664-008-9061-0

[25] Barbara Kitchenham, Lech Madeyski, and David Budgen. 2022. SEGRESS: Software Engineering Guidelines for Reporting Secondary Studies. *IEEE Transactions on Software Engineering* 49, 3 (2022), 1273–1298.

[26] Barbara Kitchenham, Rialette Pretorius, David Budgen, O. Pearl Brereton, Mark Turner, Mahmood Niazi, and Stephen Linkman. 2010. Systematic Literature Reviews in Software Engineering–a Tertiary Study. *Information and software technology* 52, 8 (2010), 792–805.

[27] Barbara Ann Kitchenham. 2015. *Evidence-Based Software Engineering and Systematic Reviews.* Chapman & Hall / CRC Innovations in Software Engineering and Software Development Series, Vol. v.4. CRC Press, Boca Raton.

[28] Ian McChesney and Raymond Bond. 2019. Eye tracking analysis of computer program comprehension in programmers with dyslexia. *Empirical Software Engineering* 24, 3 (2019), 1109–1154.

[29] Robert R. McCrae. 2020. The Five-Factor Model of personality traits: consensus and controversy. In *The Cambridge Handbook of Personality Psychology* (2nd ed.). Cambridge University Press.

[30] James Miller. 2000. Applying Meta-Analytical Procedures to Software Engineering Experiments. *Journal of Systems and Software* 54, 1 (Jan. 2000), 29–39. https://doi.org/10.1016/S0164-1212(00)00024-8

[31] David Moher, Alison Jones, Leah Lepage, and for the CONSORT Group. 2001. Use of the CONSORT Statement and Quality of Reports of Randomized TrialsA Comparative Before-and-After Evaluation. *JAMA* 285, 15 (April 2001), 1992–1995. https://doi.org/10.1001/jama.285.15.1992

[32] Marvin Muñoz Barón, Marvin Wyrich, and Stefan Wagner. 2020. An empirical validation of cognitive complexity as a measure of source code understandability. In *Proceedings of the 14th ACM/IEEE International Symposium on Empirical Software Engineering and Measurement (ESEM)* (Bari, Italy). Association for Computing Machinery, New York, NY, USA, Article 5, 12 pages. https://doi.org/10.1145/3382494.3410636

[33] P David Pearson and Gina N Cervetti. 2015. Fifty years of reading comprehension theory and practice. *Research-based practices for teaching Common Core literacy* (2015), 1–24.

[34] Norman Peitek, Janet Siegmund, and Sven Apel. 2020. What Drives the Reading Order of Programmers? An Eye Tracking Study. In *Proceedings of the 28th International Conference on Program Comprehension* (Seoul, Republic of Korea) *(ICPC '20).* Association for Computing Machinery, New York, NY, USA, 342–353.

https://doi.org/10.1145/3387904.3389279

[35] Kai Petersen, Robert Feldt, Shahid Mujtaba, and Michael Mattsson. 2008. Systematic Mapping Studies in Software Engineering. In *12th International Conference on Evaluation and Assessment in Software Engineering (EASE) 12*. 1–10.

[36] Paul Ralph and Sebastian Baltes. 2022. Paving the Way for Mature Secondary Research: The Seven Types of Literature Review. In *Proceedings of the 30th ACM Joint European Software Engineering Conference and Symposium on the Foundations of Software Engineering* (Singapore) *(ESEC/FSE 2022)*. Association for Computing Machinery, New York, NY, USA, 1632–1636. https://doi.org/10.1145/3540250.3560877

[37] Paul Ralph and Ewan Tempero. 2018. Construct Validity in Software Engineering Research and Software Metrics. In *Proceedings of the 22nd International Conference on Evaluation and Assessment in Software Engineering 2018* (Christchurch, New Zealand) *(EASE '18)*. Association for Computing Machinery, New York, NY, USA, 13–23. https://doi.org/10.1145/3210459.3210461

[38] Martin Schmettow and Wolfgang Vietze. 2008. Introducing Item Response Theory for Measuring Usability Inspection Processes. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. ACM, Florence Italy, 893–902. https://doi.org/10.1145/1357054.1357196

[39] Janet Siegmund. 2022. *MIP Talk: Measuring Programming Experience: 2012 vs. 2022*. https://www.youtube.com/watch?v=_OwQiUyVp-M

[40] Janet Siegmund, Christian Kästner, Jörg Liebig, Sven Apel, and Stefan Hanenberg. 2014. Measuring and modeling programming experience. *Empirical Software Engineering* 19 (2014), 1299–1334.

[41] I. Simera, D. Moher, J. Hoey, K. F. Schulz, and D. G. Altman. 2010. A Catalogue of Reporting Guidelines for Health Research. *European Journal of Clinical Investigation* 40, 1 (Jan. 2010), 35–53. https://doi.org/10.1111/j.1365-2362.2009.02234.x

[42] Dag IK Sjøberg and Gunnar R. Bergersen. 2023. Improving the Reporting of Threats to Construct Validity. *arXiv preprint arXiv:2306.05336* (2023). arXiv:2306.05336

[43] Dag I.K. Sjøberg and Gunnar R. Bergersen. 2022. Construct Validity in Software Engineering. *IEEE Transactions on Software Engineering* 49, 3 (2022), 1374–1396. https://doi.org/10.1109/TSE.2022.3176725

[44] Gregory T. Smith. 2005. On Construct Validity: Issues of Method and Measurement. *Psychological assessment* 17, 4 (2005), 396.

[45] Ewan Tempero and Paul Ralph. 2016. A Model for Defining Coupling Metrics. In *2016 23rd Asia-Pacific Software Engineering Conference (APSEC)*. IEEE, 145–152. https://doi.org/10.1109/APSEC.2016.030

[46] Drew Westen and Robert Rosenthal. 2003. Quantifying Construct Validity: Two Simple Measures. *Journal of personality and social psychology* 84, 3 (2003), 608.

[47] Marvin Wyrich. 2023. Source Code Comprehension: A Contemporary Definition and Conceptual Model for Empirical Investigation. arXiv:2310.11301 [cs.SE]

[48] Marvin Wyrich, Justus Bogner, and Stefan Wagner. 2023. 40 Years of Designing Code Comprehension Experiments: A Systematic Mapping Study. *ACM Comput. Surv.* (2023), 42 pages. https://doi.org/10.1145/3626522