Empirical Software Engineering manuscript No.
(will be inserted by the editor)

Confounding Parameters on Program Comprehension: A Literature Survey

Janet Siegmund$^{\pi}$ · **Jana Schumann**

Received: date / Accepted: date

Abstract Program comprehension is an important human factor in software engineering. To measure and evaluate program comprehension, researchers typically conduct experiments. However, designing experiments requires considerable effort, because confounding parameters need to be controlled for. Our aim is to support researchers in identifying relevant confounding parameters and select appropriate techniques to control their influence. To this end, we conducted a literature survey of 13 journals and conferences over a time span of 10 years. As result, we created a catalog of 39 confounding parameters, including an overview of measurement and control techniques. With the catalog, we give experimenters a tool to design reliable and valid experiments.

1 Introduction

Since the development of the first programmable computers around 1945 [46], many languages, tools, and processes were developed to improve program comprehension [20]. Program comprehension, which describes the process of how developers comprehend source code, is an important human factor in software engineering: Prior studies found that maintenance developers spend the majority of their time with understanding source code [41, 64, 65]. Furthermore, maintenance costs are the main cost factor for the development of a software system [9]. Hence, if we can improve the comprehensibility of source code, we can reduce time and cost of the entire software life cycle.

$^{\pi}$This author published previous work as Janet Feigenspan.

Janet Siegmund
University of Passau, Germany

Jana Schumann
www.janaschumann.com, Leipzig, Germany

The first step in improving program comprehension is to measure it reliably. However, program comprehension is a complex internal cognitive process: There are several models that describe program comprehension, such as top-down or bottom-up models. Top-down models describe that developers build a general hypothesis of a program's purpose and refine this hypothesis by looking at source code, using *beacons* (i.e., information in source code that give hint about a program's purpose) [11,56,63]. Bottom-up models describe that developers look at source code statement by statement and group statements to semantic chunks. These chunks are combined further, until developers can state hypotheses about the purpose of a program [49,60]. Typically, developers switch between top-down and bottom-up comprehension [41,40]. They use top-down comprehension where possible, because it is faster and requires fewer cognitive resources [56]. Developers use bottom-up comprehension only when necessary (i.e., when they have no knowledge of a program's domain). Thus, program comprehension is a complex internal cognitive process, and to reliably measure it, researchers typically conduct controlled experiments [22].

The problem with controlled experiments is that *confounding parameters* may bias the outcome (in our case, observed program comprehension) [27]. For example, program comprehension is influenced by the experience participants have, such that more experienced participants understand source code differently than novice programmers. If researchers do not take into account the difference in experience, they cannot be sure what they measure. Thus, it is important to control the influence of confounding parameters. Furthermore, to interpret the results of a controlled experiment, it is important to know how researchers managed a confounding parameter. For example, if an experiment was conducted with undergraduate students, the results of this experiment may not be valid for programming experts. Without knowing these details, experiments are difficult to interpret and replicate—we might even observe contradicting results.

With this paper, we support researchers in producing valid, reliable, and interpretable results. The contributions of this paper are twofold:

- A catalog of confounding parameters for program comprehension.
- An overview how confounding parameters are measured and controlled for.

First, with an extensive catalog of confounding parameters, researchers do not have to identify confounding parameters, but can consult the catalog and decide for each parameter whether it has an important influence or not (see Table 12). Hence, this catalog serves as aide not to overlook potentially relevant parameters.

Second, with an overview of well-established measurement and control techniques based on literature, we support researchers in selecting appropriate techniques for their studies (see Tables 10 and 11). In this way, the catalog of confounding parameters goes beyond well-known books on experimentation in software engineering (e.g., [69,36]), with a more specific focus on comprehension and more hands-on information regarding measurement and control techniques, based on what other researchers did. Thus, our work complements standard books on empirical research.

With this paper, we do not address only those researchers who are experienced with empirical studies, but also software engineers who did not get in touch with controlled experiments and want to evaluate how a new tool or language construct affects the targeted developers. Thus, we include also an overview of common control techniques, as well as parameters that are not specific to comprehension experiments, but typical for all experiments with human participants (such as motivation, selection, and learning effects).

To fulfill our goals, we conducted a literature survey of papers published between 2001 and 2010 in the following journals and conferences:

- *Empirical Software Engineering (ESE)*,
- *Journal of Software: Evolution and Process (JSEP)*,
- *Transactions on Software Engineering and Methodology (TOSEM)*,
- *Transactions on Software Engineering (TSE)*,
- *International Conference on Program Comprehension (ICPC)*[1],
- *International Conference on Software Engineering (ICSE)*,
- *International Conference on Software Maintenance (ICSM)*,
- *International Symposium on Empirical Software Engineering and Measurement (ESEM)*[2],
- *Symposium on the Foundations of Software Engineering (FSE)*,
- *Symposium on Visual Languages and Human-Centric Computing (VL-HCC)*[3],
- *Conference on Human Factors in Computing Systems (CHI)*,
- *Cooperative and Human Aspects of Software Engineering (CHASE)*[4], and
- *Working Conference on Reverse Engineering (WCRE).*

We selected these journal and conferences, because they are the leading platforms to publish results regarding (empirical) software engineering and program comprehension. We included 872 (of 4935) papers in our initial selection and extracted 39 confounding parameters, such as programming experience, intelligence, and ordering effects.

We found that there is only little agreement on how to manage confounding parameters. Instead, the discussion of confounding parameters often appears to be haphazard. This makes interpreting results of experiments difficult, because it is not clear whether and how all relevant confounding parameters were considered and controlled for.

The remainder of this paper is structured as follows:

Section 2: Process of selection of papers and extraction of confounding parameters.

[1] ICPC was a workshop until 2005.

[2] ESEM originated 2007 from merging the International Symposium on Empirical Software Engineering (ISESE) and International Software Metrics Symposium (METRICS)

[3] VLHCC was called Human-Centric Computing Languages and Environments until 2003.

[4] CHASE first took place in 2008.

Section 3: Overview of how confounding parameters are currently managed in literature.
Section 4: Introduction to common control techniques for confounding parameters.
Section 5: Detailed description of all extracted confounding parameters and how they are measured and controlled for in literature.
Section 6: Threats to validity of our survey.
Section 7: Recommendations on how to manage confounding parameters in program-comprehension experiments.
Section 8: Related work.
Section 9: Conclusion and future work.

2 Methodology

In this section, we discuss the selection of journals and conferences, the selection of papers, and the extraction of confounding parameters. This way, we enable other researchers to extend our data with other journals, conferences, and issues.

To collect confounding parameters, we need a representative selection of papers. To this end, we chose different journals and conferences. We selected ESE as leading platform for empirical research in the field of software engineering. We consider JSEP, TOSEM, and TSE as leading journals in software engineering. ICPC is the leading conference for program-comprehension research. ICSE and FSE are the leading conferences on software engineering. ICSM is the leading conference regarding software maintenance. We chose ESEM as platform in the empirical-software-engineering domain. Furthermore, CHI and VLHCC are the leading conferences regarding human-computer interaction, and CHASE is a recently established workshop in the context of human factors. Finally, WCRE is one of the leading conferences regarding reverse engineering. From each journal and conference, we considered all papers published between 2001 and 2010. Hence, we have a representative set of journals and conferences.

Since not all kinds of experiments are relevant for our survey, we give a short overview of different types of experiments (see, e.g., Sjoberg et al. [62]) and outline which types are relevant. In general, a setting in which a treatment is deliberately applied to a group of participants is called *experiment*, with the following different characteristics:

- randomized experiment,
- quasi experiment,
- correlational study, and
- case study.

First, if participants are randomly assigned to treatment and control condition(s), an experiment is referred to as *randomized experiment.* Second, in a *quasi experiment*, participants are not assigned randomly to conditions, for

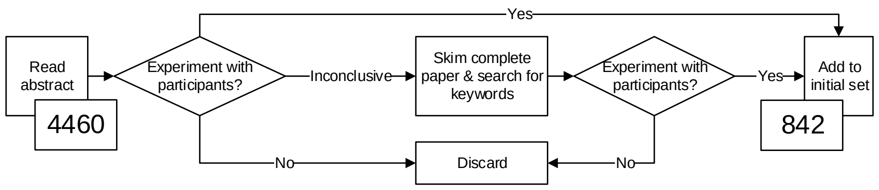

Read abstract
4460
Experiment with participants?
Inconclusive
Yes
No
Skim complete paper & search for keywords
Experiment with participants?
Yes
No
Discard
Add to initial set
842

Fig. 1: Approach to select papers that describe experiments with subjects. Numbers denote the number of papers in the according step.

example, when groups are already present (which is often the case in studies conducted in companies). Third, in a *correlational study*, size and direction of relationships among variables are observed. Fourth, in *case studies*, only one or few participants are observed and the outcome has a qualitative nature.

For our survey, we include all types of experiments except for correlational studies that observe only existing data, because no human participants were observed. For example, Bettenburg and others analyzed the commit data of an Eclipse version six month before and after its release to identify how commit comments help to predict bugs [6]. Since this experiment was not conducted with human participants, we excluded it.

We also included experiments with a qualitative focus (including case studies and quasi experiments), although confounding parameters play a minor role in these studies. For example, Ko and others conducted an exploratory study to find out how developers seek and use relevant information [38]. In this study, the goal was to generate hypotheses, so authors measured confounding parameters to get a more holistic view of developers' behavior, but did not control for all confounding parameters. Thus, in qualitative studies, relevant confounding parameters also have to be considered and reported, although controlling for them is not the primary concern.

To extract relevant papers from the selected journals and conferences, we started with reading the abstract of a paper. If the abstract described an experiment with human participants, we added the paper to our initial selection; if not, we discarded it. If the abstract was inconclusive, we skimmed through the paper for any information that indicates the conduct of an experiment. Furthermore, we searched the paper with a fixed set of keywords: (programming) experience, expert, expertise, professional, subject, and participant. Those keywords are typical for comprehension experiments with human participants. Based on skimming and the search result, we either added a paper to our initial selection or discarded it. To have a better understanding of our approach, we visualize it in Figure 1. As result of this selection process, we have an initial set of 842 papers.

As next step, we read each paper of our initial selection completely. During that process, we discarded some papers, because the described experiment was too far away from program comprehension. Before discarding a paper, we (the authors) discussed whether it is relevant until we reached inter-personal consensus. When in doubt, we included a paper to avoid omitting potentially relevant parameters. We excluded 457 papers, so we have 385 papers in the final selection. On the project's website[5], we have a catalog of all extracted papers, including the ones we discarded. In Table 1, we show how many papers we selected for the initial and final set.

As last step, we extracted confounding parameters. To this end, we included variables that authors categorized as confounding (or extraneous) variables (for example, some authors listed these variables in a table or stated *Our confounding parameters are...*). Furthermore, we included variables that followed terms like *To control for*, *To avoid bias due to*, or *A threat to validity was caused by*, because such a variable was treated as confounding variable.

We used an initial set of confounding parameters defined in the first authors master's thesis [20], also based on literature. Every time we encountered a new confounding parameter, we revisited already analyzed papers.

The selection and extraction process was done by the two authors of this paper and a research assistant. The second author and the assistant selected the papers from disjoint sets of venues; the first author checked on random samples of selected and not selected papers the correctness of the selection process. We discussed disagreements until reaching interpersonal consensus. The first author extracted confounding parameters, and the second author checked the correctness of the extraction on random samples. We discuss the validity of this approach in more detail in Section 6.

Next, we present an overview of how confounding parameters are currently managed.

3 State of the Art

In this section, we present insights of how confounding parameters are managed in literature. The main findings are:

- Only a fraction of identified confounding parameters are mentioned in each paper.
- Most confounding parameters are reported in one location.
- Researchers use different ways to control for the same confounding parameter.

We discuss each of the findings in detail.

3.1 Number of Confounding Parameters

To give a fair impression of how many confounding parameters are described, we distinguish the experiments in qualitative and quantitative experiments. Qualitative experiments typically observe few participants, but collect and

[5] `http://www.infosun.fim.uni-passau.de/spl/janet/confounding/index.php`

Source		2001	2002	2003	2004	2005	2006	2007	2008	2009	2010	Sum
ESE	All	24	18	15	15	19	21	24	26	24	16	**202**
	Extr.	2	9	5	8	10	7	7	4	7	3	**62**
	Final	1	5	1	4	3	3	3	2	5	2	**29**
JSEP	All	19	21	18	17	15	18	17	18	15	29	**187**
	Extr.	4	4	3	5	1	2	2	3	1	5	**30**
	Final	1	2	0	2	1	1	2	1	0	0	**10**
TOSEM	All	11	15	13	10	12	12	15	21	13	13	**135**
	Extr.	0	0	0	0	1	0	3	0	0	3	**7**
	Final	0	0	0	0	0	0	2	0	0	0	**2**
TSE	All	66	73	88	72	68	61	55	53	50	48	**634**
	Extr.	5	6	6	5	5	3	2	5	3	5	**45**
	Final	4	4	5	3	3	3	1	5	2	3	**33**
ICPC	All	28	24	22	21	24	23	22	21	22	16	**223**
	Extr.	2	3	3	8	8	3	4	5	4	3	**43**
	Final	2	2	3	3	7	3	4	5	4	3	**36**
ICSE	All	47	48	42	58	44	36	49	56	20	52	**482**
	Extr.	8	10	8	1	11	12	9	8	9	12	**88**
	Final	0	3	3	4	3	4	1	3	4	5	**30**
ICSM	All	67	60	41	38	55	42	46	40	34	50	**473**
	Extr.	7	10	4	5	8	11	9	4	5	5	**68**
	Final	1	1	0	2	0	3	2	2	4	2	**17**
ESEM	All	-	-	-	-	-	-	45	29	44	30	**148**
	Extr.	-	-	-	-	-	-	12	3	11	8	**34**
	Final	-	-	-	-	-	-	6	5	1	7	**19**
FSE	All	29	17	42	25	32	25	63	31	39	34	**337**
	Extr.	0	1	0	0	1	2	3	3	1	1	**12**
	Final	0	1	0	0	0	2	2	3	1	0	**9**
VLHCC	All	47	17	21	21	28	19	18	24	21	27	**240**
	Extr.	11	11	6	11	9	10	8	7	9	12	**94**
	Final	6	9	6	8	6	6	7	4	6	5	**63**
CHI	All	69	61	75	93	93	119	144	158	204	230	**1246**
	Extr.	17	16	16	19	22	24	22	35	52	47	**270**
	Final	16	8	8	9	11	12	8	14	13	21	**120**
CHASE	All	-	-	-	-	-	-	-	28	22	18	**68**
	Extr.	-	-	-	-	-	-	-	11	9	8	**28**
	Final	-	-	-	-	-	-	-	2	1	3	**6**
WCRE	All	24	33	35	28	22	24	27	32	31	29	**285**
	Extr.	4	6	6	7	3	6	9	4	5	11	**61**
	Final	0	1	1	2	1	1	1	0	1	3	**11**

All: All papers of the source in the according year. Extr.: Extracted papers in our initial selection. Final: Papers in the final selection (after discarding non-relevant papers).

Table 1: Overview of all, included, and extracted papers by year and venue.

analyze detailed information, such as think-aloud data or (screen-capture) videos. In qualitative studies, controlling for confounding parameters is not the primary concern, but rather getting a detailed insight in what participants did.

Quantitative experiments recruit a larger number of participants and are interested in quantitative information, such as response time, correctness, or efficiency. In quantitative studies, controlling for confounding parameters is more important than in qualitative, and, thus, typically more confounding parameters are taken into account. Consequently, making statements about how many confounding parameters are described independent of the kind of study would bias the presentation of results.

In Figure 2, we give an overview of how many papers mentioned how many parameters, separated by the kind of study. For example, of the qualitative studies, 17 papers did not report any confounding parameter. For both qualitative and quantitative studies, only a fraction of confounding parameters is mentioned in each paper. For qualitative experiments, the fraction of parameters is lower as for quantitative experiments. This is not surprising, because qualitative experiments are less concerned with controlling for confounding parameters.

However, most authors may have considered more parameters than they actually described, but that space restrictions prohibit mentioning each parameter and how it was controlled for. This raises the question that, if not all controlled parameters are mentioned in literature, a literature survey is the right instrument to extract confounding parameters. We discuss this in Section 6.

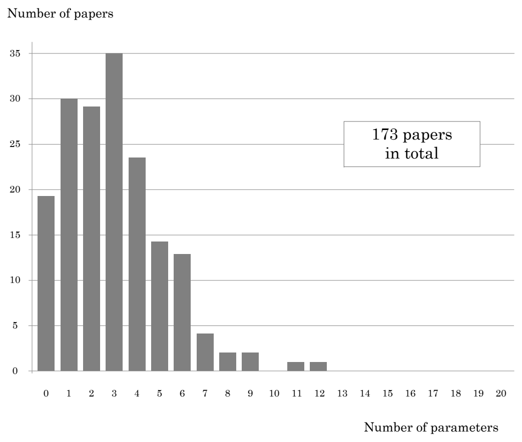
3.2 Reporting Confounding Parameters

We found that most confounding parameters are described at a distinct location in the papers. Typically, experiment descriptions consist of the following parts [34]:

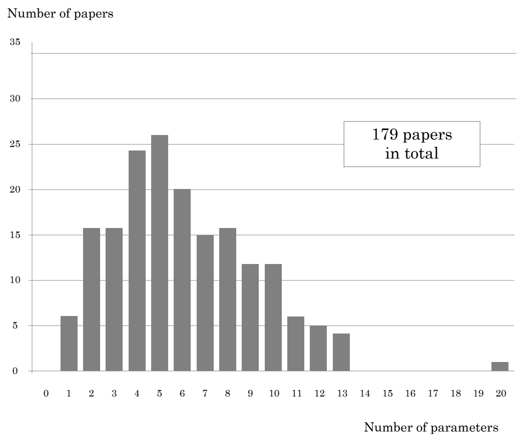
- experimental design,
- analysis,
- interpretation, and
- threats to validity.

In experimental design, authors describe the setting of an experiment, including material, participants, and means to control for confounding parameters. In the analysis, the authors present the data analysis, for example, means, standard deviations, and statistical tests. After the analysis, the results of the experiment are interpreted, such that the results are set in relation to the research questions or hypotheses. Finally, authors discuss the validity of the experiments.

In Table 2, we give an overview in which parts a parameter was mentioned first, separately for qualitative and quantitative experiments. N denotes the total amount of how often parameters were mentioned in each section; the mean



(a) Qualitative experiments



(b) Quantitative experiments

Fig. 2: Number of parameters mentioned per paper.

denotes the average relative amount of parameters of all papers mentioned in the according section. For both qualitative and quantitative experiments, most parameters were discussed during the experimental design, the stage in which means to manage confounding parameters are typically defined.

For qualitative experiments, only a small fraction of the parameters are mentioned in the other parts of the experiment descriptions. For quantitative experiments, about 17 % of the confounding parameters are described in threats to validity. In this part, authors mostly describe confounding parameters, how they could have threatened the validity of the experiments, and how they controlled a parameter so that the threat to validity is minimized.

Part	N	Mean
Experimental design	471	90.9 %
Analysis	12	2.3 %
Interpretation	15	2.9 %
Threats to validity	20	3.9 %
Total	518	100 %

(a) Qualitative experiments

Part	N	Mean
Experimental design	975	77.6 %
Analysis	42	3.3 %
Interpretation	28	2.2 %
Threats to validity	211	16.8 %
Total	1256	100 %

(b) Quantitative experiments

Table 2: Overview of how often a parameter was mentioned first in a part of the experiment description.

Thus, the major part of confounding parameters is described in the experimental design. Nevertheless, there is still room for improvement, such that all parameters are reported in the experimental design, supporting the readers of according papers in getting a quick overview of relevant confounding parameters.

Furthermore, there is no systematic way to describe confounding parameters. Although we often found terms like *Our confounding parameters are...* or *To control for*, they were not used consistently. For example, authors described that they measured programming experience or that they trained participants to use a tool, but did not describe why they did it or what control technique they applied. Experienced researchers can recognize this implicit mentioning of a confounding parameter, but researchers or students who are unfamiliar with empirical research might overlook it. Additionally, such implicit mentioning makes it difficult to get a quick overview of an experimental design.

3.3 Controlling for Confounding Parameters

There are various ways to control the influence of a confounding parameter[6]. For example, to control for programming experience, authors kept the level of programming experience constant by recruiting only students or created two groups with comparable level of programming experience. To create comparable groups, researchers had to measure programming experience, which they realized (among others) by using the years a participant has been programming, a participant's level of education (e.g., undergraduate vs. graduate level), self estimation, or supervisor estimation. In some cases, authors wrote that they controlled for a parameter, but did not specify how.

The different means of controlling for confounding parameters can make the comparison of different experiments difficult. For example, when comparing programming experience based on years a participant has been programming, and based on the level of education, it is likely that both measure different things; an undergraduate student may have been programming for 20 years,

[6] In Section 5, we discuss techniques and parameters in detail. Here, we give only an overview.

whereas a graduate student may have started programming when starting to study. This gets worse when we do not know how a parameter was managed. Thus, researchers might not be able to fully understand and replicate an experiment.

To summarize, there is effort to control for confounding parameters and to describe them consistently. However, reporting this effort is too unsystematic, so it is difficult to evaluate the soundness of an experimental design. To address the identified problems, we give recommendations in Section 7.

4 Techniques to Control for Confounding Parameters

In this section, we present common techniques to control for confounding parameters. This section is aimed at researchers who are inexperienced with conducting experiments. Readers familiar with controlling for confounding parameters may skip this section.

Experimentation in psychology has a long history [70]. Hence, all control techniques are based on psychological research and have proved useful in countless experiments. There are five typical ways to control for confounding parameters, which we present in detail in this section:

1. randomization,
2. matching,
3. keep confounding parameter constant,
4. use confounding parameter as independent variable, and
5. analyze the influence of confounding parameters on results.

For better illustration, we describe the control techniques with the confounding parameter *programming experience* as example. It describes how familiar participants are with implementing source code (we go into more detail in Section 5.1.2).

4.1 Randomization

Using randomization, a confounding parameter is randomly assigned to experimental groups, for example, by tossing a coin or rolling a dice. This way, the influence of confounding parameters is assumed to spread evenly across experimental groups, such that the influence is comparable in all groups [27]. For example, a sample of students should be split into two comparable groups regarding programming experience. To this end, researchers toss a coin to assign all participants to two groups. Since participants are randomly assigned to groups, there is no systematic bias. That is, the coin toss does not assign more experienced participants to one group and less experienced participants to another group. Hence, both groups should be comparable, or homogeneous, regarding programming experience.

Participant	Value	Group
5	67	A
7	63	B
1	62	B
10	59	A
8	57	A
6	57	B
3	53	B
2	50	A
9	45	A
4	43	B

Table 3: Fictional programming-experience values and according group assignments.

For randomization to be effective, the sample size needs to be large enough, so that statistical errors can even out [2]. Unfortunately, large cannot be defined as a fixed number. Assigning 30 participants to two experimental groups seems reasonably large for creating 2 comparable groups, but assigning 30 participants to six experimental groups may be too small to ensure six homogeneous groups. Thus, the more experimental groups there are, the more participants we need. In personal correspondence with other researchers, we found that five participants per group are too few, but ten seem to be sufficient.

Randomization is the most convenient way to control for a confounding parameter, because it does not require measuring a parameter. However, one disadvantage is that researchers cannot draw any conclusions about the effect of a confounding parameter on program comprehension. For that, it needs to measured, which is required by the remaining control techniques.

4.2 Matching

If the sample size is too small, researchers can apply matching or balancing [27]. In this case, researchers measure a confounding parameter and assign participants to experimental groups, such that both groups have about the same size and same level of a confounding parameter. To illustrate matching, we show fictional values for programming experience of participants in Table 3. The participants are ordered according to the quantified programming-experience value. Now, we assign Participant 5 to Group A, Participant 7 to Group B, Participant 1 to Group B, and Participant 10 to Group A. We repeat this process until all participants are assigned to groups.

Matching ensures homogeneous groups according to a parameter. However, as a drawback, researchers have to measure a confounding parameter. For example, for programming experience, researchers can ask participants to estimate their experience or to implement a simple task and use the performance as indicator for programming experience. But it is not clear how well this captures true programming experience. Thus, requires a valid and reliable way to measure a parameter.

4.3 Keep Confounding Parameter Constant

When keeping a confounding parameter constant, there is exactly one level of this parameter in an experimental design [20]. For example, to keep programming experience constant, researchers can measure programming experience and recruit participants only with a certain value. Alternatively, researchers can recruit participants from a population of which they know that a parameter has only one level. For instance, freshmen typically have one low, comparable programming-experience level. Students who started programming before they enrolled can be excluded. This way, researchers can minimize the effort of measuring a parameter. However, the generalizability reduces, because the results are only applicable to the selected level of programming experience. Next, we present a technique that allows researchers to maintain generalizability.

4.4 Use Confounding Parameter as Independent Variable

A confounding parameter can be included as independent variable in an experimental design [20]. This way, researchers can manipulate it and control its influence. For example, researchers can recruit participants with high and low programming experience, such that the results are applicable to people with high and low experience. However, the experimental design becomes more complex, because now there is one more independent variable; if the initial independent variable has two levels, and programming experience, also with two levels, is included, there are four different experimental groups. Additionally, there may be an interaction between both factors.

In addition to a more complex design, researchers also need to extend the research hypotheses to include the confounding parameter. Furthermore, with increasing number of experimental groups, more participants are necessary. As a benefit, internal validity can be increased without decreasing external validity at the same time.

4.5 Analyze the Influence of Confounding Parameter on Result

When participants cannot be assigned to experimental groups, researchers can analyze the influence of a confounding parameter afterwards [55]. In this case, researchers can measure a parameter and analyze its influence on the result after conducting the experiment. This is often necessary when researchers recruit participants from companies, because they cannot assign participants to different companies. This technique is similar to using a parameter as independent variable, but it allows researchers to also analyze confounds that emanated during the experiment (e.g., a system crash). To this end, there are

Technique	Sample size	Requires measurement	Effort	Generalizability
Randomization	large	no	low	depends on sample
Matching	any	yes	depends on parameter	limited
Constant	any	depends on parameter	depends on whether measurement is needed	limited
Independent	large	depends on parameter	high	good
Analyzed afterwards	any	yes	depends on parameter	depends on parameter

Table 4: Benefits and drawbacks of control techniques.

different techniques, for example, an ANOVA to evaluate whether the comprehension of participants depends on the employing company, in addition to or in interaction with the independent variable(s) [2]. However, an ANOVA assumes that the data are normally distributed—otherwise, researchers need to apply a non-parametric test, such as the Friedman test (if the experimental design is perfectly balanced and if there are repeated measures) [23] or a permutation test [1].

These five techniques are the most common control techniques. There are also other techniques that are specific for a confounding parameter. We describe these techniques when we explain a corresponding parameter.

In Table 4, we summarize the control techniques and their benefits and drawbacks. For example, randomization requires a relatively large sample size, does not require measuring a parameter, the effort is low, and the generalizability depends on the selected sample; if it consists only of students, the results are only applicable for students, but if researchers include several levels of experience, the results also apply to more experienced programmers. Note that the benefits and drawbacks also depend on how a technique is applied and circumstances of experiments, so the benefits and drawbacks are only an approximation.

5 Confounding Parameters

In this section, we present the confounding parameters we extracted. For a better overview, we divide confounding parameters into two categories: individual and experimental parameters. Individual parameters are related to the person of the participants, such as programming experience or intelligence. Experimental parameters are related to the experimental setting, such as tasks or source code. We found 16 individual and 23 experimental parameters, which we discuss in detail.

5.1 Individual Parameters

In Table 5, we summarize how often individual confounding parameters on program comprehension are considered. We found 16 individual parameters that are mentioned in literature. To have an understanding of the role of the parameters, we describe each parameter, including how it influences the result, and give an overview of how it can be measured and controlled for, which is all based on the literature survey. Some parameters are specifically important for program comprehension, which we explicitly discuss for according parameters. In the appendix (Table 10), we present a summary of the measurement of confounding parameters.

Variable	ESE	JSEP	TOSEM	TSE	ICPC	ICSE	ICSM	ESEM	FSE	VLHCC	CHI	CHASE	WCRE	Sum
Individual background (Section 5.1.1)														
Color blindness	0	0	0	0	1	0	0	0	0	0	3	0	0	**4**
Culture	0	0	0	2	1	0	0	0	0	0	4	0	0	**7**
Gender	0	0	0	3	4	1	0	0	0	5	50	1	0	**72**
Intelligence	0	0	0	2	4	1	0	0	0	0	4	0	0	**11**
Individual knowledge (Section 5.1.2)														
Ability	12	4	2	12	7	5	6	4	2	2	10	0	2	**68**
Domain knowledge	3	1	0	5	4	0	6	0	0	4	17	0	3	**43**
Education	8	0	1	6	15	8	3	3	0	14	2	0	0	**60**
Familiarity with study object	19	7	2	17	10	10	11	12	6	29	51	2	4	**180**
Familiarity with tools	5	3	2	9	8	9	13	1	3	31	62	3	2	**151**
Programming experience	24	4	2	25	23	22	14	11	5	37	35	3	4	**209**
Reading time	0	0	0	0	3	0	0	1	0	0	0	0	0	**4**
Individual circumstances (Section 5.1.3)														
Fatigue	8	2	0	5	0	2	2	5	0	0	9	0	0	**22**
Motivation	12	2	0	10	7	3	1	2	0	0	10	0	1	**48**
Treatment preference	0	0	0	0	3	1	0	2	0	0	4	0	0	**10**

Table 5: Individual confounding parameters.

Parameter	Ran.	Mat.	Con.	Ind.	Ana.	Dis.	Not sp.	Other
Individual background (Section 5.1.1)								
Color blindness	0	0	3	0	0	1	0	0
Culture	0	0	2	0	0	3	0	3
Gender	1	6	5	5	12	5	0	34
Intelligence	0	0	4	0	1	0	1	2
Individual knowledge (Section 5.1.2)								
Ability	9	11	8	6	18	4	7	7
Domain knowledge	1	1	22	3	4	8	3	2
Education	0	3	27	2	5	6	4	12
Familiarity with study object	2	6	123	2	12	6	11	20
Familiarity with tools	2	0	132	0	6	4	3	9
Programming experience	10	13	40	19	9	19	37	22
Reading time	0	0	1	0	2	1	0	0
Individual circumstances (Section 5.1.3)								
Fatigue	1	8	1	0	2	3	0	17
Motivation	3	1	23	0	5	5	4	7
Treatment preference	0	2	3	0	1	3	0	1

Ran.: Randomization; Mat.: Matching; Con.: Kept constant; Ind.: Used as independent variable; Ana.: Analyzed afterwards; Dis.: A parameter was discussed; Not sp.: A parameter was not specified; Other: Other control technique than mentioned

Table 6: Control techniques for individual confounding parameters.

For a better overview, we present a summary of how each parameter was controlled for in Table 6 and divide individual parameters into the categories *individual background*, *individual knowledge*, and *individual circumstances*.

5.1.1 Individual Background

Individual background describes parameters that have a fixed value for a participant, that is, with which participants are born and that hardly change during life time.

Color blindness describes the limited perception of certain colors, for example, red and green [25]. When colors play a role in an experiment, for example, when participants see source code with syntax highlighting or when the effectiveness of background colors is analyzed, color-blind participants might respond slower than other participants or be unable to solve a task if they cannot distinguish colors.

Color blindness was considered in four experiments. Jablonksi and Hou [32] described the color-blindness of one participant as threat to validity. In other experiments, it was kept constant by including only participants with normal color vision. None of the authors mentioned how they determined color-blind participants. To measure color blindness, the Ishihara test was developed [31]. When controlling for color blindness, researchers need to keep in mind that only a small fraction of people are color blind [25]. Thus, randomization may not be suitable, because from 20 participants, the one or two potentially color blind might easily be assigned to the same group.

Culture refers to the origin of participants. This can affect the outcome, because different cultures (especially Western compared to Asian cultures) often have different ways to solve a problem (e.g., [30]). Consequently, some participants may be slower, but more thorough when completing a task or hide their real opinion to not annoy experimenters.

In seven of the reviewed papers, culture was mentioned. Some mentioned that by recruiting participants from the same company or class, culture was kept constant. However, this assumption holds only partially, because often, students have different background. Another way was to include a representative set of different cultural backgrounds to avoid the influence of culture on results, or to measure culture of participants [43]. To avoid discriminating against participants by excluding them, researchers can also let a participant complete the experiment and then exclude the data set from the analysis.

Gender of participants might influence program comprehension, as several studies show. For example, Beckwith and others found that females are reluctant (compared to males) to accept new debugging features when working with spreadsheets [4], but that proper tutorials can help females to accept new features [28]. In another study, Sharafi and others found that female participants are more careful when selecting and ruling out wrong identifiers [57]. Thus, gender can influence how participants perform in comprehension experiments.

Gender was mentioned in numerous papers in literature. On one occasion, authors used randomization [67]. Often, authors balanced gender among groups, included it as independent variable, or analyzed it afterwards. As with culture, researchers have to be careful not to discriminate against participants.

Intelligence[7] has long tradition in psychology and many different definitions and views exist. Unfortunately, generations of researchers did not come to an agreement about one definition of intelligence. It can be defined as the ability to solve problems, memorize material (e.g., using working-memory capacity), recognize complex relationships, or combinations thereof [33, 50, 68]. Intelligence can influence program comprehension, because higher problem-solving skills and/or memory skills can enable participants to faster understand source code.

In our literature review, authors rarely considered intelligence. When authors did take it into account, they often focused on one facet of intelligence. Most often, this facet was working memory. To keep it constant, such that the working-memory capacity was not exceeded, material was either presented on paper to participants (so they can look it up any time and do not need to keep it in working memory), or the number of items (such as elements in UML diagrams) was in the range 7 ± 2, which is the average[8] working-memory ca-

[7] There are voices that say intelligence is rather something learned than something inborn. Thus, we could also classify it as individual knowledge. However, since our classification aims at a better overview, we do not step into this discussion.

[8] There are controversial discussion about the magical number seven [3].

pacity [44]. However, authors rarely applied a test to confirm that the working-memory capacity of participants was not exceeded. If working memory plays a crucial role, researchers can also apply tests to measure it [47]. In two papers, intelligence was specified not as working memory: Ko and Uttl applied a verbal intelligence test as indicator for general intelligence [39], and Corbett and Anderson used the math score of the SAT as indicator [14]. Thus, intelligence has many facets.

5.1.2 Individual Knowledge

Individual knowledge describes parameters that are influenced by learning and experience. These parameters change, but rather slowly over a period of weeks, months, or years.

Ability as a general term describes skills or competence of participants. The more and higher ability participants have (e.g., regarding implementing code or using language constructs), the better they may comprehend source code. Unfortunately, authors rarely specified what they mean with ability. Based on the descriptions in the papers, ability can be summarized as the skill level of participants regarding the study object, such as writing code or UML modeling. Since we intend to have a broad overview of confounding parameters, we keep this parameter without specifying it further.

Measuring ability often includes a further test or task in the experiment (e.g., a short programming task), which increases the experiment time. One often applied way was to use the grade of participants. Another way was to let superiors estimate participants' ability, or to let participants estimate their own ability. There are also tests to measure ability in terms of programming skills [5], but none of the papers mentioned such a test.

Domain knowledge describes how familiar participants are with the domain of the study object, for example, databases. It influences whether they use top-down or bottom-up comprehension. Usually, top-down comprehension is faster than bottom-up comprehension, because developers can compare source code with what is in their memory [56], and familiar identifier names give hints about the purpose of a method or variable [11]. With bottom-up comprehension, a developer has to analyze each statement, which inherently takes more time.

Domain knowledge was considered in 43 papers. To measure it, authors either asked participants or assumed familiarity based on the courses participants were enrolled in or already completed. In some cases, authors selected uncommon domains, such as hydrology, and assumed that participants had no knowledge about it. Domain knowledge has a strong influence on the comprehension process (fast top down vs. slow bottom-up comprehension), so assessing it can reduce bias to the results.

Education describes the topics participants learned during their studies. It does not capture the status of participants' studies (e.g., freshman, sophomore, graduate student). If students visited mostly programming courses, their skills are different from students who mostly visited database or graphical-user-interface courses, in which programming is not the primary content.[9]

Authors often considered the education of participants. In most cases, authors kept it constant by recruiting participants of the same course. In some other cases, authors asked participants the courses they completed. Based on the courses, authors assumed that participants learned specific topics. Education can directly affect domain knowledge, because participants obtained knowledge through the courses they completed. Thus, assessing relevant topics of participants' education can help to better understand the results of an experiment.

Familiarity with study object/tools refers to how experienced participants are with the evaluated concepts or tools, such as Oracle database or Eclipse. Familiarity with the study object appears to be the same as domain knowledge. However, looking closer, they slightly differ: Domain knowledge describes the domain of a study object (e.g., databases), familiarity with the study object the object itself (e.g., Oracle database as one concrete database system). If participants are familiar with the study object or tools, they do not need as much cognitive resources as unfamiliar participants, because learning something new requires an initial cognitive effort that decreases with increasing familiarity [54]. Thus, participants who are familiar with the study object or the tool might perform better. We summarize familiarity with the study object and tools, because they are closely related.

Both parameters were often considered in our review. In most cases, authors kept the influence constant. To assure a comparable level of familiarity, participants were often trained or required to be familiar with a tool. To measure familiarity, authors asked participants how familiar they are or conducted a pretest. Familiarity with the study object/tools can influence results, because familiar participants use certain features of a tool that makes a task easier (e.g., using the feature *Call Hierarchy* of an IDE to see the call graph of a variable). There are different options of controlling both parameters, for example, recruiting only unfamiliar participants, train all participants, or deactivate features that make tasks easier.

Programming experience describes the experience participants had so far with writing and understanding source code. The more source code participants have seen and implemented, the better they can adapt to comprehending source code, and the higher the chance is that they will be more efficient in comprehension experiments [53,42].

Programming experience is the major confounding parameter in program-comprehension experiments: The longer a participant has been programming, the more insignificant other influences (e.g., intelligence, education, or ability) become. Not surprisingly, it was considered most often in our review (209 times). However, researchers often used their own definition of programming experience, such as the years a participants has been programming, the education level, self estimation, the size of completed projects, supervisor estimation,

[9] The specific contents of courses depend on the country and specific university.

or a pretest. Beyond that, many researchers did not specify how they defined and measured programming experience, or did not control for it. To reliably control its influence, researchers can use a validated instrument (e.g., [21]), instead of using an ad hoc definition that differs between different experiments and researcher groups.

Reading time refers to how fast participants can read. The faster they are, the more they can read in a given time interval. Consequently, they may be faster in understanding source code.

However, reading source code is only one part in the comprehension process. Consequently, it was not often considered. In all cases where reading time was considered, researchers used an eye tracker to measure it. Another way is to let participants simply read a text and stop the time. There may be special settings where reading time is relevant, for example, when numerous comments are involved in the study, or when the readability of a new programming language should be assessed.

5.1.3 Individual Circumstances

Parameters in this category describe how participants feel at the time of the experiment. These parameters can change rapidly (i.e., within minutes).

Fatigue describes that participants get tired and lose concentration. This occurs especially in long experiments, because humans can work concentrated for about 90 minutes [35]. After that, attention decreases, which could affect performance of participants, such that the error rate increases toward the end of the experiment.

To avoid the influence of fatigue, researchers often had a short enough session. In some studies, authors asked their participants afterwards whether they felt fatigue with ongoing time, or assessed whether performance dropped toward the end of a session. With different task orders, influence of fatigue can also be reduced.

Motivation refers to how motivated participants are to take part in the experiment. If participants are not motivated, it may affect their performance negatively [45].

Most often, motivation was kept constant. To this end, most participants took part voluntarily (in contrast to making participation mandatory to successfully complete a course). Additionally, we found that authors rewarded the best-performing participant(s). In one study, authors included the performance in the experiment as part of a participant's grade for a course to ensure high motivation [58]. To measure motivation, authors asked participants to estimate their motivation.

Treatment preference refers to whether participants prefer a certain treatment, such as a new tool. This can affect performance, because participants might need more time or are not willing to work with a tool if they do not like it.

Treatment preference was not considered very often, and it does not appear very relevant for program-comprehension experiments. However, if a new tool or technique is part of the evaluation, treatment preference should at least be measured, because participant might like or dislike a tool or technique just because it is new. To measure treatment preference, researchers can ask participants afterwards about their opinion.

5.2 Experimental Parameters

Experimental parameters are related to the experiment and its setting. We found 23 parameters, which we summarize in Table 7. We describe each parameter, explain how it can influence the result, present how it was measured and controlled for in literature (summarized in Table 11 in the appendix). If a parameter is specifically important for program-comprehension experiments, we discuss this explicitly. In Table 8, we give a summary of how each parameter was controlled for. For a better overview, we divide experimental parameters into four categories: *subject-related*, *technical*, *context-related*, and *study-object-related*.

Variable	ESE	JSEP	TOSEM	TSE	ICPC	ICSE	ICSM	ESEM	FSE	VLHCC	CHI	CHASE	WCRE	Sum
Subject related (Section 5.2.1)														
Evaluation apprehension	0	1	0	1	1	0	0	0	0	0	0	0	0	**3**
Hawthorne effect	9	1	1	3	2	2	1	5	0	2	7	0	1	**36**
Process conformance	15	0	1	10	4	5	6	8	1	2	3	0	1	**56**
Study-object coverage	2	2	0	0	1	0	4	1	0	0	0	0	0	**10**
Ties to persistent memory	0	0	0	0	1	0	0	0	0	0	0	0	0	**1**
Time pressure	7	0	0	4	1	0	2	2	0	1	7	0	2	**26**
Visual effort	0	0	0	0	1	0	0	0	0	0	0	0	0	**1**
Technical (Section 5.2.2)														
Data consistency	0	1	0	1	0	0	0	0	0	0	0	0	1	**3**
Instrumentation	8	0	0	8	2	0	1	1	0	0	5	0	1	**26**
Mono-method bias	2	0	0	1	0	0	0	0	0	0	0	0	0	**3**
Mono-operation bias	2	0	0	1	1	0	0	0	0	0	0	0	0	**4**
Technical problems	0	0	0	0	2	0	0	2	0	0	4	0	0	**8**
Context related (Section 5.2.3)														
Learning effects	15	5	0	14	16	7	4	9	4	7	17	0	2	**100**
Mortality	0	0	0	1	0	0	1	0	1	0	2	0	0	**5**
Operationalization of study object	1	0	1	0	0	0	0	0	0	0	0	0	0	**2**
Ordering	5	2	0	7	8	2	1	2	3	12	48	0	1	**91**
Rosenthal	10	0	1	2	3	3	1	5	0	2	10	0	2	**39**
Selection	11	1	1	6	1	2	0	2	1	0	3	0	1	**29**
Study-object related (Section 5.2.4)														
Content of study object	5	2	1	1	9	0	2	2	1	0	2	1	0	**26**
Language	7	2	2	14	23	13	7	7	6	12	23	1	2	**119**
Layout of study object	4	1	0	2	7	0	2	3	1	4	21	0	0	**45**
Size of study object	14	3	1	19	15	9	8	6	3	6	14	0	1	**99**
Tasks	6	0	0	6	14	5	2	4	2	1	20	1	2	**63**

Table 7: Experimental confounding parameters.

5.2.1 Subject-related parameters

Subject-related parameters are caused by participants and only emerge because participants take part in an experiment. In this way, they differ from individual parameters, which are always present.

Evaluation apprehension refers to the fear of being evaluated. This may bias responses of participants toward what they perceive as better. For example, participants could judge tasks easier than they actually think to hide from the experimenter that they had difficulties. Another problem might be that participants cannot show their best performance, because they feel frightened (which decreases their performance).

Evaluation apprehension was only rarely considered. To avoid its influence, researchers assured anonymity for participants or ensured participants that their performance does not affect the grade for a course. Another way is to encourage participants to answer honestly by clarifying that only honest answers are of value.

The *Hawthorne effect* is closely related to evaluation apprehension. It describes that participants behave differently in experiments, because they are being observed [51]. Like evaluation apprehension, we may observe different behavior than we would have if we observed participants in a realistic environment.

In most cases, authors avoided the Hawthorne effect by not revealing their hypotheses to participants. Going one step farther, it is also possible not let participants know that they take part in an experiment. However, both often conflict with an informed consent that participants give before the experiment. An ethics committee helps to ensure fair treatment of all participants. In one experiment, authors measured the Hawthorne effect by comparing the performance in a context-neutral task to performance in treatment tasks [18].

Process conformance means how well participants followed their instructions. If participants deviate from their instructions, for example, searching the internet for solutions or given subsequent participants information about the experiment, the results may be biased.

We found different ways to ensure process conformance. Most often, participants were observed to assure process conformance. In one experiment with several sessions, participants were not allowed to take any material home [10], and in another experiment, data of participants who deviated from the protocol were deleted [24]. In an experiment with children, parents were allowed to watch, but not to interfere [15]. Furthermore, three experiments used different tasks for participants seated next to each other. In some experiments, it might be useful to allow participants to work at home. However, in this case, researchers cannot monitor participants' process conformance. In such settings, it can help to encourage participants to follow the instructions (e.g., by stating that data are only useful when the protocol was followed), to ask participants how well they followed the protocol, and/or to analyze the effect of deviations afterwards.

Parameter	Ran.	Mat.	Con.	Ind.	Ana.	Dis.	Not sp.	Other
Subject related (Section 5.2.1)								
Evaluation apprehension	0	0	0	0	0	0	0	3
Hawthorne effect	0	0	1	0	0	8	0	25
Process conformance	0	3	5	0	10	8	0	29
Study-object coverage	0	0	5	1	1	1	0	2
Ties to persistent memory	0	0	0	1	0	0	0	0
Time pressure	0	1	9	0	4	5	0	7
Visual effort	0	0	0	1	0	0	0	0
Technical (Section 5.2.2)								
Data consistency	0	0	0	0	0	1	0	2
Instrumentation	0	0	5	0	2	15	0	4
Mono-method bias	0	0	0	0	0	0	0	3
Mono-operation bias	0	0	0	0	0	1	0	2
Technical problems	0	0	5	0	0	0	0	3
Context related (Section 5.2.3)								
Learning effects	10	29	6	1	19	11	2	15
Mortality	0	0	0	0	0	5	0	0
Operationalization of study object	0	0	0	0	0	1	0	0
Ordering	19	56	4	0	5	5	0	3
Rosenthal	0	0	0	0	0	9	0	29
Selection	5	4	1	0	2	13	1	3
Study-object related (Section 5.2.4)								
Content of study object	2	2	8	0	3	6	0	9
Language	2	0	81	1	3	9	6	11
Layout of study object	1	7	9	8	3	8	1	7
Size of study object	1	4	6	1	4	5	2	78
Tasks	2	20	3	2	6	11	2	14

Ran.: Randomization; Mat.: Matching; Con.: Kept constant; Ind.: Used as independent variable; Ana.: Analyzed afterwards; Dis.: A parameter was discussed; Not sp.: A parameter was not specified; Other: Other control technique than mentioned

Table 8: Control techniques for experimental confounding parameters.

Study-object coverage describes how much of the study object was covered by participants. If a participant solved half as much tasks as another participant, it could bias the results, such that the slower participant was more thorough.

Often, authors controlled for study-object coverage by excluding data of participants who did not complete all tasks. In one experiment, authors compared how the difference between groups changed (based on confidence intervals) when participants who did not finish the task were excluded [48].

Ties to persistent memory refers to links of the experimental material to persistent (or long-term) memory of participants. If source code has no ties to persistent memory and working memory becomes flooded (e.g., because of long variable names or long method calls), comprehension may be impaired.

Ties to persistent memory was relevant in only one study [8]. It was measured in terms of the usage of identifiers: Identifiers often used in packages were assumed to have ties to persistent memory, whereas program or domain identifiers have no ties to persistent memory.

Time pressure means that participants feel they have to hurry to complete the experiment in a given time interval. This can bias the performance, such that participants make more errors when time is running out.

To avoid the influence of time pressure, authors often did not set a time limit for a task. However, there are often time constraints, for example, when an experiment replaces a regular lecture or exercise session or when an experiment mimics time pressure of realistic industrial settings. In these cases, authors analyzed the influence of time pressure afterwards or designed the experimental tasks such that participants can comfortably solve them within the time limit. To measure time pressure, authors often asked after the experiment whether participants experienced time pressure.

Visual effort describes the number and length of eye movements to find a correct answer. The more effort a task has, the longer it takes to find the correct answer.

Visual effort was relevant in only one experiment [59]. It was controlled for by analyzing the eye movements of participants with an eye tracker.

5.2.2 Technical Parameters

Technical parameters are related to the experimental set up, such as the tools that are used.

Data consistency refers to how consistent data of the experiment are. For example, when paper-based answers of participants are digitalized, answers can be forgotten or transferred wrongly. Inconsistent data can bias the results, because researchers might analyze something different than they measured.

In our review, three papers controlled for data consistency. For example, Biffl and others checked data digitalized from paper with two independent reviewers [7]. Especially when transcribing paper-based data to a digital form, data consistency may be compromised. In pilot studies, researchers can test whether there are any systematic threats to data consistency.

Instrumentation refers to instruments used in the experiment, such as questionnaires, tasks, or eye trackers. The use of instruments can influence the result, especially when instruments are not carefully designed or are unusual for participants.

To avoid instrumentation effects, we found several ways: Authors conducted pilot studies [29], evaluated the instruments based on design principles [17], or avoided the influence of instrumentation by using standard instruments, for example, to present speech [26]. Thus, to control for instrumentation effects, researchers can use validated instruments, or, if there are none, carefully design their own by consulting literature and/or experts.

Mono-method bias means that only one measure is used to measure a variable, for example, only response time of programming tasks to measure program comprehension. If that measure is badly chosen, the results may be biased. For example, when participants wanted to finish a task independent of correctness, response time is not a good indicator.

In three papers, we found that authors controlled for mono-method bias by using different measures for comprehension. For example, to measure program comprehension, researchers used correctness and response time of tasks, and/or an efficiency measure as combination of both.

Mono-operation bias is related to mono-method bias; it refers to an under-representation of the evaluated construct, for example, when researchers use only one task to measure comprehension. If that task is not representative, the results might be biased. For example, a task can be designed such that it confirms a hypothesis.

In our review, authors controlled for mono-operation bias by using different tasks [66] or representative tasks. To ensure representativeness, we researchers consult literature and/or domain experts.

Technical problems can occur during any experiment, for example, a computer crash or missing questionnaires for participants. This may bias the results, because participants have to repeat a task on a computer or that answers of a participants get lost.

In literature, the most common technical problem was a system crash, and authors avoided its influence by excluding data of according participants.

5.2.3 Context-Related Parameters

Context-related parameters are typical problems of experiments, such as participants who drop out or learn from experimental tasks.

Learning effects mean how participants learn during the session of an experiment. This is especially problematic in within-subject designs, in which participants experience more than one treatment level.

Authors considered learning effects very often. In most cases, authors used a counter-balanced or between-subjects design, so that learning effects are avoided or can be measured. Additionally, authors conducted a training before the experiment, so participants learned mostly during the training, not during the experiment. Furthermore, to analyze afterwards how learning affected the results, authors compared the performance of participants in subsequent tasks.

Mortality occurs when participants do not complete all tasks. This is especially a problem in multi-session experiments, where participants have to return for sessions. Mortality may influence the results, because participants may not drop out randomly, but, for example, only low-skilled participants because of frustration caused by the perceived difficulty of the experiment.

Only five papers discussed the effect of mortality on their result, but we also found only few papers with multi-session experiments. If researchers need multiple sessions, they can encourage participants to return, for example, by giving participants a reward in each session or in the last session if all other sessions have been attended.

Operationalization of study object describes how the measurement of the study object is defined. For example, to measure program comprehension, researchers can use the correctness of solutions to tasks. An example for an inappropriate measure is the number of files participants looked at. If the operationalization is inappropriate, then not the study object, but something else is measured, leading to biased results.

In our review, we found that the operationalization of study object was discussed a few times. However, authors typically carefully operationalized the study object without explicitly discussing whether their operationalization was suitable. To this end, authors often used the literature and/or experts.

Ordering describes the influence of the order in which tasks or experimental treatments are applied. If the solution of one task automatically leads to the solution of subsequent tasks, but not the other way around, a different order of these tasks leads to different results.

Most authors chose an appropriate experimental design (e.g., counter-balanced, between-subjects) to avoid or measure the effect of ordering afterwards. Another way was to randomize the order of tasks, so that, with a large enough sample, ordering effects should be ruled out.

The *Rosenthal effect* occurs when experimenters influence consciously or subconsciously the behavior of participants [52]. This can influence the result, especially when researchers assess participants' opinion about a new technique or tool, such that participants rate it more positive.

In nearly all studies in which the Rosenthal effect was considered, authors avoided its influence. To this end, authors were careful not to bias participants, were objective (i.e., they did not develop the technique under evaluation), used standardized instructions (i.e., defined the specific wording of what experimenters say to participants), left the experimenters blind regarding hypotheses or experimental group of participants, or let several reviewers evaluate the objectivity of material. Since it is difficult to measure whether and how experimenters influenced participants, researchers can use means to avoid the Rosenthal effect, for example, by using standardized sets of instructions.

Selection refers to how the participants for an experiment are selected. If the sample is not representative, the conclusions are not applicable to the intended population. For example, if researchers select students as participants, they cannot apply the results to programming experts.

To control for selection bias, researchers have to ensure selecting a representative sample, for example, by randomly recruiting participants from the intended population. However, this is not feasible in most cases (e.g., we cannot recruit all students who start to learn Java from all over the world). Typically, authors recruited participants from one university or company (i.e., convenient sampling), but took care to randomly select participants or to create a representative sample. Additionally, authors communicated the selection of participants as threat to validity.

5.2.4 Study-Object-Related Parameters

Study-object-related parameters describe properties of the study object, such as its size.

Content of study object describes what source code or models are about. If the content between two groups is different, it may bias the results, because one study object is more difficult to comprehend. For example, when comparing the comprehensibility of object-oriented with imperative programming based on two programs, researchers need to make sure that both programs differ only in the paradigm, not the language or the functionality they are implementing.

In most cases, authors used the same or comparable content of study object to avoid its influence. Furthermore, authors selected realistic task settings. Since the influence of content of study object is difficult to measure directly, authors relied on their own or expert estimation regarding comparability of content. Another way is to use standardized material if possible.

Language refers to the underlying programming language of the experiment. We could also summarize language under *familiarity with the study object* or *content of study object*, but decided to keep it separate, because for program comprehension, the underlying programming language has an important influence. If participants work with an unfamiliar programming language, their performance is different compared to when they work with a familiar language, because they need additional cognitive resources for understanding the unfamiliar language (which also counts for familiarity with study object/tools, cf. Section 5.1.2).

The influence of language is especially important for program-comprehension experiments. Consequently, many authors considered it. Most often, they kept the influence of language constant by recruiting participants with a specified skill level (e.g., at least three years of Java experience). In some cases, authors used a short pretest to determine the language skill level. If uncommon features of a language are relevant for the experiment, researchers can explicitly assess whether participants are familiar with them.

Layout of study object describes how participants see the study object, such as source code or a UML model. For example, source code can be formatted according to different guidelines or not formatted consistently, or different UML models can have different layouts. This may influence the comprehension of participants, because they have to get used to the layouts.

For layout of study object, the same counts as for content of study object: It is difficult to measure, so most authors avoided its influence by choosing comparable layouts or selecting realistic layouts (e.g., standard formatting styles). Several papers also included the layout as independent variable, so that authors could determine its influence on the result.

Size of study object refers to how large an object is, for example, the number of lines of source code or the number of elements in a UML model. The larger an object is, the more time participants need to work with it. If treatment and control object differ in their size, the results of the experiment are also influenced by different sizes, not only different treatments.

As for content and layout of study object, size should be comparable across different treatments. To measure size, authors used lines of code, number of files/classes, or number of elements in a UML model. However, many authors only measured the size of study object, but did not describe whether and how they controlled its influence. If researchers already determined the size of study object, they can also analyze afterwards whether it influenced the results.

Task describes how tasks can differ, for example, in difficulty or complexity. If the difficulty of tasks for different treatments is not the same, then the difficulty would also have an effect on the outcome, besides the independent variable.

To avoid the influence due to different tasks, authors often used matching by choosing standardized or comparable tasks. If standardized tasks are available, researchers should use them, because they have already proven useful in several experiments, and they increase comparability across different experiments. Otherwise, consulting the literature and/or experts to create tasks also helps to avoid its influence.

5.3 Concluding Remarks about Confounding Parameters

To summarize, there are numerous confounding parameters for program comprehension. There are no general measurement and control techniques for all parameters, but depending on the circumstances of the experiment, the most suitable techniques need to be chosen. To support researchers in this decision, we gave an overview of measurement and control techniques based on comprehension experiments that we encountered in our literature review.

The categorization we used here serves as an overview and should not be seen as absolute. For example, intelligence can be defined as something that is learned rather than inborn. However, since the goal of the categories is to have a better overview, we do not step into this discussion.

Furthermore, it might seem unsettling that some parameters, such as mono-operation bias or operationalization of study object, are considered in only few studies. However, authors may have controlled parameters more often than we found in our review, but space restrictions may have prohibited authors to mention all considered parameters. Thus, the actual number of how often confounding parameters are controlled may be higher than we found.

Additionally, some parameters appear very similar. For example, domain knowledge and familiarity the with study object seem to be the same at first glance. However, looking closer, they slightly differ, such that domain knowledge describes the domain of a study object (e.g., databases), and familiarity with the study object the object itself (e.g., Oracle database as one concrete database system). To have a broad overview and enable experimenters to look at parameters from different points of view, we kept the parameters separate. This way, we hope that experiments can better decide whether and how a parameter is relevant.

6 Threats to Validity

Like for every literature survey, the selection of journals, conferences, and articles as well as the data extraction may be biased. First, we selected four journals, one workshop, and eight conferences that are the leading publication platform in their field. However, we could easily select more relevant venues. To reduce this threat, we selected a broad spectrum and also included more general sources in the area of software engineering, not only venues for empirical research. Additionally, we could have considered a larger time span, but 10 years is sufficiently large to get a solid starting point for an exhaustive catalog of confounding parameters. In future work, we and others can consider papers of additional venues and years to extend our catalog.

Second, the selection of articles and extraction of parameters may be biased. In our survey, we had two reviewers selecting the papers (of disjoint sets of venues), and one reviewer extracting the confounding parameters. Due to resource constraints, we could not apply standard techniques, such as grounded theory, card sorting, or having at least two reviewers evaluate the complete selection and extraction process. To minimize bias, we checked the selection and extraction of the other reviewer on random samples. That is, the reviewers who selected the papers checked the extraction process, and the reviewer who extracted the parameters checked the selection process. When we found a different decision about the inclusion of a paper or parameter, we discussed it until reaching interpersonal consensus. In future work, we and others can increase the validity by letting independent reviewers conduct the selection and extraction process and compute agreement measures, such as Cohen's Kappa [12].

Third, the list of keywords ((programming) experience, expert, expertise, professional, subject, participant) may lead to incorrectly excluding a paper. However, based on our expertise, these keywords are typical for experiments. Additionally, we used these keywords in conjunction with skimming the paper to minimize the number of falsely discarding a paper. Furthermore, we excluded several papers of our initial selection, so we do not have irrelevant papers in our final selection. Thus, we minimized the threat caused by the selection of keywords.

Fourth, it is unlikely that we have extracted all confounding parameters that might influence the results of program-comprehension experiments. Although we had a broad selection of papers of 10 years from different journals and conferences, there might be parameters missing. For example, the size of the monitor on which the study object is presented might influence the result, or the operating system, because a participant is used to a different one than what is used in the experiment. Thus, our catalog can be extended. To minimize the number of missed parameters, we set the selection and extraction criteria for papers and confounding parameters as broad as possible. Thus, our catalog provides a good foundation for creating sound experimental designs. Nevertheless, in future work, we and others can further reduce this threat by conducting a survey with experienced empirical researchers about confounding parameters (mentioned and not mentioned in this paper) as well as their relevance.

7 Recommendations

In this section, we give recommendations on how to manage confounding parameters, which count for both, qualitative and quantitative studies:

- Decide whether a confounding parameter is relevant for an experiment and use appropriate measurement and control techniques.
- Describe all confounding parameters explicitly in the design part of a report.
- Report whether and how confounding parameters are measured and controlled for.

First, researchers have to decide whether a confounding parameter is relevant and choose appropriate measurement and control techniques. To this end, researchers can consult the catalog, including measurement techniques (cf. Tables 10 and 11 in the appendix), and decide for each parameter whether it is relevant or not and how it can be controlled for. Discussing the relevance of a parameter and according measurement and control techniques in a group of researchers can further reduce the risk of neglecting relevant parameters or choosing inappropriate measurement or control techniques.

Having decided on each relevant parameter and according measurement and control techniques, there is still a chance of missing something. For example, if researchers keep the language constant by recruiting participants with Java experience, some tasks might still require knowledge of specific Java syntax (e.g., adding a leading zero to an int treats the number as octal). In such a case, applying additional qualitative methods, such as a think-aloud protocol [19], helps experimenters to better understand what is going on with participants.

Second, we suggest to describe all confounding parameters in the design part of a report and explicitly defining it as confounding parameter. For example, Jedlitschka and others suggest reporting hypotheses and variables in one section as part of the experiment planning [34]. We recommend listing confounding parameters also in this section. This way, other researchers can easily perceive which confounding parameters were considered as relevant.

Third, to describe whether and how researchers controlled for a confounding parameter, we suggest a pattern similar to the one described in Table 9. We illustrate this pattern with the parameters programming experience, the Rosenthal effect, and ties to persistent memory.[10] We mention each parameter, provide an abbreviation to reduce the space we need to refer to it, describe the control technique(s) and why we applied it, and describe how we measured it and why we measure it that way or ensured that it does not bias

[10] For examples of all identified parameters for specific experiments, see the first author's PhD thesis [61].

Parameter	Abbr.	Control technique		Measured/Ensured	
		How?	Why?	How?	Why?
Programming experience	PE	Matching	PE major confound	Education level	undergraduates have less experience than graduates
Rosenthal effect	RE	Standardized instructions	Avoid it	Not measured	Standardized instructions to avoid it
Ties to persistent memory	Ties	None	Not relevant	-	-

Abbr.: Abbreviation for parameter.

Table 9: Pattern to describe confounding parameters

our results. This way, other researchers can see at first glance how and why a confounding parameter was measured and controlled for. This way, replicability of experiments can be improved, because all relevant information for confounding parameters is mentioned at one defined location.

We are aware that most reports on experiments have space restrictions. To avoid incomplete descriptions of confounding parameters, a short description of the most important parameters can be given in the report, and the complete catalog of parameters and according measurement and control techniques can be provided at a website or a technical report. This way, reports do not become bloated, but all relevant information is available. We hope that this way, a more standard way to manage confounding parameters will emerge, and we would be happy to learn about the experience of empirical researchers who follow these recommendations.

8 Related Work

Based on work in psychology, Wohlin and others provide a checklist of confounding parameters for software-engineering experiments, which contains general confounding parameters for experiments in software engineering [69]. This is a good starting point for experiments, and also helps researchers to not forget possibly relevant parameters. In contrast to our work, the catalog is not based on a literature survey of comprehension experiments, but on standard psychological literature [13]. Thus, this checklist applies for experiments in software engineering in general, whereas our catalog is tailored to comprehension experiments and complements the catalog of Wohlin and others.

There is a lot of work on surveys about experiments in software engineering. For example, Sjøberg and others conducted a survey about the amount of empirical research in software engineering [62]. They found that only a fraction of the analyzed papers report on controlled experiments. Furthermore, the reporting of threats to validity (which are caused by confounding parameters) is often vague and unsystematic. Dybå and others found that the statistical power in software-engineering experiments is rather low and suggested, among others, to improve validity, which in turn increases statistical power [16]. Kampenes and others analyzed the conduct of quasi experiments and found that their design, analysis as well as reporting can be improved [37]. Similar to our work, all studies showed that there is room for improvement when conducting and reporting controlled experiments in software engineering. In contrast to these studies, we focus on the aspect of confounding parameters, such that we support researchers in managing them. In the long run, design and reporting of empirical studies can be improved.

9 Conclusion

Experiments in software engineering become more and more important. However, designing experiments is tedious, because confounding parameters need to be identified, measured, and controlled for, independent of the kind of study. In this paper, we present a catalog of confounding parameters for comprehension experiments based on a literature survey, including applied measurement and control techniques. So far, we identified 39 confounding parameters that should be considered in comprehension experiments. With this catalog, we give researchers a tool that helps them to create sound experimental designs, which is necessary to obtain valid and reliable results.

In future work, there are several options to continue our work. First, our catalog can be extended by considering other years and venues, not necessarily restricted to the computer-science domain, but including other domains that use empirical research. Second, since our catalog of confounding parameters is not complete, we can conduct explorative studies to discover more relevant confounding parameters. Additionally, we can ask experts in empirical research about their opinion of relevant confounding parameters. This could also be combined with a rating of the importance of each confounding parameter, so that researchers can better decide whether a parameter may be relevant or not.

Acknowledgements Thanks to Norbert Siegmund and Christian Kästner for helpful discussions. Thanks to all reviewers for their constructive feedback. Thanks to Raimund Dachselt for his encouragement to write this article. Thanks to Andreas Meister for his support in selecting relevant papers.

A Appendix

In Tables 10 and 11, we give a summary of how each parameter was measured in literature.

The checklist in Table 12 can help researchers to control the influence of confounding parameters. Researchers can document how they measured and controlled for a parameter.

Parameter	Measurement
Individual background (Section 5.1.1)	
Color blindness	Ishihara test [31]; ask participants
Culture	Ask participants (avoid discriminating against anyone)
Gender	Ask participants (avoid discriminating against anyone)
Intelligence	Intelligence tests, e.g., BIS [33], CFT [50], WMC test [47]
Individual knowledge (Section 5.1.2)	
Ability	Grades of courses; supervisor estimation; pretest
Domain knowledge	Ask participants; assume knowledge based on courses, pretest
Education	Ask participants; assume knowledge based on courses
Familiarity with study object	Ask participants; pretest
Familiarity with tools	Ask participants; pretest
Programming experience	Questionnaire, e.g., as is currently developed by us [21]
Reading time	Eyetracker; measure time participants need to read a text
Individual circumstances (Section 5.1.3)	
Fatigue	Self estimation; performance (decreasing performance may be indicator for fatigue)
Motivation	Self estimation; performance with ongoing experiment time
Treatment preference	Self estimation

Table 10: Measurement techniques of individual confounding parameters.

References

1. Anderson, M.: Permutation Tests for Univariate or Multivariate Analysis of Variance and Regression. Canadian Journal of Fisheries and Aquatic Sciences **58**(3), 626–639 (2001)
2. Anderson, T., Finn, J.: The New Statistical Analysis of Data. Springer (1996)
3. Baddeley, A.: Is Working Memory Still Working? The American Psychologist **56**(11), 851–864 (2001)
4. Beckwith, L., Burnett, M., Wiedenbeck, S., Cook, C., Sorte, S., Hastings, M.: Effectiveness of end-user debugging software features: Are there gender issues? In: Proc. Conf. Human Factors in Computing Systems (CHI), pp. 869–878. ACM Press (2005)
5. Bergersen, G., Gustafsson, J.E.: Programming Skill, Knowledge, and Working Memory Among Professional Software Developers from an Investment Theory Perspective. Journal of Individual Differences **32**(4), 201–209 (2011)
6. Bettenburg, N., Hassan, A.: Studying the Impact of Social Structures on Software Quality. In: Proc. Int'l Conf. Program Comprehension (ICPC), pp. 124–133. IEEE CS (2010)
7. Biffl, S., Halling, M.: Investigating the Defect Detection Effectiveness and Cost Benefit of Nominal Inspection Teams. IEEE Trans. Softw. Eng. **29**(5), 385–397 (2003)
8. Binkley, D., Lawrie, D., Maex, S., Morrell, C.: Impact of Limited Memory Resources. In: Proc. Int'l Conf. Program Comprehension (ICPC), pp. 83–92. IEEE Computer Society (2008)
9. Boehm, B.: Software Engineering Economics. Prentice Hall (1981)
10. Briand, L.C., Labiche, Y., Di Penta, M., Yan-Bondoc, H.D.: An Experimental Investigation of Formality in UML-Based Development. IEEE Trans. Softw. Eng. **31**(10), 833–849 (2005)
11. Brooks, R.: Using a Behavioral Theory of Program Comprehension in Software Engineering. In: Proc. Int'l Conf. Software Engineering (ICSE), pp. 196–201. IEEE CS (1978)
12. Cohen, J.: A Coefficient of Agreement for Nominal Scales. Educational and Psychological Measurement **20**(1), 37–46 (1960)

Parameter	Measurement
Subject related (Section 5.2.1)	
Evaluation apprehension	Ask participants; better: avoid it
Hawthorne effect	Almost impossible; better: avoid it
Process conformance	Ask participants; better: observe participants to ensure it
Study-object coverage	Ask participants; look at answer and log data of participants
Ties to persistent memory	Ask participants; categorize material accordingly (e.g., based on participants' domain knowledge)
Time pressure	Ask participants
Visual effort	Eye tracker
Technical (Section 5.2.2)	
Data consistency	Check data
Instrumentation	Avoid by carefully choosing instruments (best: validated)
Mono-method bias	Avoid by using at least two measures
Mono-operation bias	Avoid by using different operations (e.g., more than one task)
Technical problems	Avoid by pilot tests; evaluate data of according participants
Context related (Section 5.2.3)	
Learning effects	Counter-balanced design allows looking for them
Mortality	Evaluate whether drop outs differ from other participants
Operationalization of study object	Consult experts on study object
Ordering	Different orders allow measurement
Rosenthal	Almost impossible to measure; better: avoid it with standardized instructions
Selection	Almost impossible to measure; better: avoid it by randomly drawing participants from population
Study-object related (Section 5.2.4)	
Content of study object	Ask independent reviewers to ensure comparable and suitable content
Language	Ask participants; pretest
Layout of study object	Ask independent reviewers to ensure comparable and suitable layout; use standardized layout
Size of study object	Ask independent reviewers to ensure comparable and suitable size
Tasks	Ask independent reviewers to ensure comparable and suitable; use standardized tasks

Table 11: Measurement techniques of experimental confounding parameters.

13. Cook, T., Campbell, D.: Quasi-Experimentation: Design & Analysis Issues for Field Settings. Houghton Mifflin (1979)
14. Corbett, A., Anderson, J.: Locus of feedback control in computer-based tutoring: Impact on learning rate, achievement and attitudes. In: Proc. Conf. Human Factors in Computing Systems (CHI), pp. 245–252. ACM Press (2001)
15. Druin, A., Foss, E., Hutchinson, H., Golub, E., Hatley, L.: Children's Roles Using Keyword Search Interfaces at Home. In: Proc. Conf. Human Factors in Computing Systems (CHI), pp. 413–422. ACM Press (2010)
16. Dybå, T., Kampenes, V.B., Sjøberg, D.: A Systematic Review of Statistical Power in software Engineering Experiments. Journal of Information and Software Technology **48**(8), 745–755 (2006)

Parameter	Abbr.	Control technique		Measured/Ensured	
		How?	Why?	How?	Why?
Individual background					
Color blindness					
Culture					
Gender					
Intelligence					
Individual knowledge					
Ability					
Domain knowledge					
Education					
Familiarity with study object					
Familiarity with tools					
Programming experience					
Reading time					
Individual circumstances					
Fatigue					
Motivation					
Treatment preference					
Subject related					
Evaluation apprehension					
Hawthorne effect					
Process conformance					
Study-object coverage					
Ties to persistent memory					
Time pressure					
Visual effort					
Technical					
Data consistency					
Instrumentation					
Mono-method bias					
Mono-operation bias					
Technical problems					
Context related					
Learning effects					
Mortality					
Operationalization of study object					
Ordering					
Rosenthal					
Selection					
Study-object related					
Content of study object					
Language					
Layout of study object					
Size of study object					
Tasks					

Table 12: Checklist of confounding parameters.

17. Dzidek, W., Arisholm, E., Briand, L.: A Realistic Empirical Evaluation of the Costs and Benefits of UML in Software Maintenance. IEEE Trans. Softw. Eng. **34**(3), 407–432 (2008)
18. Ellis, B., Stylos, J., Myers, B.: The Factory Pattern in API Design: A Usability Evaluation. In: Proc. Int'l Conf. Software Engineering (ICSE), pp. 302–312. IEEE CS (2007)
19. Ericsson, K., Simon, H.: Verbal Reports as Data. Psychological Review **87**(2), 215–251 (1980)
20. Feigenspan, J.: Empirical Comparison of FOSD Approaches Regarding Program Comprehension – A Feasibility Study. Master's thesis, University of Magdeburg (2009)
21. Feigenspan, J., Kästner, C., Liebig, J., Apel, S., Hanenberg, S.: Measuring Programming Experience. In: Proc. Int'l Conf. Program Comprehension (ICPC), pp. 73–82. IEEE CS (2012)
22. Feigenspan, J., Siegmund, N., Fruth, J.: On the Role of Program Comprehension in Embedded Systems. In: Proc. Workshop Software Reengineering (WSR), pp. 34–35 (2011). `http://wwwiti.cs.uni-magdeburg.de/iti\_db/publikationen/ps/auto/FeSiFr11.pdf`
23. Friedman, M.: The use of ranks to avoid the assumption of normality implicit in the analysis of variance. Journal of the American Statistical Association **32**(200), 675–701 (1937)
24. Fry, Z., Weimer, W.: A human study of fault localization accuracy. In: Proc. Int'l Conf. Software Maintenance (ICSM), pp. 1–10. IEEE CS (2010)
25. Goldstein, B.: Sensation and Perception, fifth edn. Cengage Learning Services (2002)
26. Gong, L., Lai, J.: Shall we mix synthetic speech and human speech impact on users' performance, perception, and attitude. In: Proc. Conf. Human Factors in Computing Systems (CHI), pp. 158–165. ACM Press (2001)
27. Goodwin, J.: Research in Psychology: Methods and Design, second edn. Wiley Publishing, Inc. (1999)
28. Grigoreanu, V., Cao, J., Kulesza, T., Bogart, C., Rector, K., Burnett, M., Wiedenbeck, S.: Can feature design reduce the gender gap in end-user software development environments? In: Proc. Symposium Visual Languages and Human-Centric Computing (VLHCC), pp. 149–156. IEEE CS (2008)
29. Güleşir, G., Berg, K., Bergmans, L., Akşit, M.: Experimental evaluation of a tool for the verification and transformation of source code in event-driven systems. Empirical Softw. Eng. **14**(6), 720–777 (2009)
30. Hu, W., Lee, H., Zhang, Q., Liu, T., Geng, L., Seghier, M., Shakeshaft, C., Twomey, T., Green, D., Yang, Y., , Price, C.: Developmental Dyslexia in Chinese and English Populations: Dissociating the Effect of Dyslexia from Language Differences. Brain **133**(6), 1694–1706 (2010)
31. Ishihara, S.: Test for Colour-Blindness. Kanehara Shuppan Co. (1972)
32. Jablonski, P., Hou, D.: Aiding Software Maintenance with Copy-and-Paste Clone-Awareness. In: Proc. Int'l Conf. Program Comprehension (ICPC), pp. 170–179. IEEE CS (2010)
33. Jäger, A., Süß, H.M., Beauducel, A.: *Berliner Intelligenzstruktur-Test*. Hogrefe (1997)
34. Jedlitschka, A., Ciolkowski, M., Pfahl, D.: Reporting Experiments in Software Engineering. In: Guide to Advanced Empirical Software Engineering, pp. 201–228. Springer (2008)
35. Jensen, E.: Teaching with the Brain in Mind. Atlantic Books (1998)
36. Juristo, N., Moreno, A.: Basics of Software Engineering Experimentation. Kluwer (2001)
37. Kampenes, V., Dybå, T., Hannay, J., Sjøberg, D.: A Systematic Review of Quasi-Experiments in Software Engineering. Information and Software Technology **51**(1), 71–82 (2009)
38. Ko, A., Myers, B., Coblenz, M., Aung, H.: An Exploratory Study of How Developers Seek, Relate, and Collect Relevant Information during Software Maintenance Tasks. IEEE Trans. Softw. Eng. **32**(12), 971–987 (2006)
39. Ko, A., Uttl, B.: Individual Differences in Program Comprehension Strategies in Unfamiliar Programming Systems. In: Proc. Int'l Workshop Program Comprehension (IWPC), pp. 175–184. IEEE CS (2003)
40. von Mayrhauser, A., Vans, M.: Program Comprehension During Software Maintenance and Evolution. Computer **28**(8), 44–55 (1995)

41. von Mayrhauser, A., Vans, M., Howe, A.: Program Understanding Behaviour during Enhancement of Large-scale Software. Journal of Software Maintenance: Research and Practice **9**(5), 299–327 (1997)
42. McConnell, S.: What does 10x Mean? Measuring Variations in Programmer Productivity. In: Making Software, pp. 567–574. O'Reilly & Associates, Inc. (2011)
43. McQuiggan, S.W., Rowe, J.P., Lester, J.C.: The Effects of Empathetic Virtual Characters on Presence in Narrative-Centered Learning Environments. In: Proc. Conf. Human Factors in Computing Systems (CHI), pp. 1511–1520. ACM Press (2008)
44. Miller, G.: The Magical Number Seven, Plus or Minus Two: Some Limits on Our Capacity for Processing Information. Psychological Review **63**(2), 81–97 (1956)
45. Mook, D.: Motivation: The Organization of Action, second edn. W.W. Norton & Co. (1996)
46. Neumann, J.v.: *First Draft of a Report on the EDVAC* (1945)
47. Oberauer, K., Süß, H.M., Schulze, R., Wilhelm, O., Wittmann, W.: Working memory capacity—facets of a cognitive ability construct. Personality and Individual Differences **29**(6), 1017–1045 (2000)
48. Oezbek, C., Prechelt, L.: Jtourbus: Simplifying program understanding by documentation that provides tours through the source code. In: Proc. Int'l Conf. Software Maintenance (ICSM), pp. 64–73. IEEE CS (2007)
49. Pennington, N.: Stimulus Structures and Mental Representations in Expert Comprehension of Computer Programs. Cognitive Psychologys **19**(3), 295–341 (1987)
50. Raven, J.: Mental Tests Used in Genetic Studies: The Performances of Related Individuals in Tests Mainly Educative and Mainly Reproductive. Master's thesis, University of London (1936)
51. Roethlisberger, F.: Management and the Worker. Harvard University Press (1939)
52. Rosenthal, R., Jacobson, L.: Teachers' Expectancies: Determinants of Pupils' IQ Gains. Psychological Reports **19**(1), 115–118 (1966)
53. Sackman, H., Erikson, W., Grant, E.: Exploratory Experimental Studies Comparing Online and Offline Programming Performance. Commun. ACM **11**(1), 3–11 (1968)
54. Schlaug, G.: The Brain of Musicians. A Model for Functional and Structural Adaptation. Annals of the New York Acadamy of Sciences **930**, 281–299 (2001)
55. Shadish, W., Cook, T., Campbell, D.: Experimental and Quasi-Experimental Designs for Generalized Causal Inference. Houghton Mifflin Company (2002)
56. Shaft, T., Vessey, I.: The Relevance of Application Domain Knowledge: The Case of Computer Program Comprehension. Information Systems Research **6**(3), 286–299 (1995)
57. Sharafi, Z., Soh, Z., Guéhéneuc, Y.G., Antoniol, G.: Women and men–different but equal: On the impact of identifier style on source code reading. In: Proc. Int'l Conf. Program Comprehension (ICPC), pp. 27–36. IEEE CS (2012)
58. Sharif, B., Maletic, J.: An Empirical Study on the Comprehension of Stereotyped UML Class Diagram Layouts. In: Proc. Int'l Conf. Program Comprehension (ICPC), pp. 268–272. IEEE CS (2009)
59. Sharif, B., Maletic, J.: An Eye Tracking Study on camelCase and under_score Identifier Styles. In: Proc. Int'l Conf. Program Comprehension (ICPC), pp. 196–205. IEEE CS (2010)
60. Shneiderman, B., Mayer, R.: Syntactic/Semantic Interactions in Programmer Behavior: A Model and Experimental Results. Int'l Journal of Parallel Programming **8**(3), 219–238 (1979)
61. Siegmund, J.: Framework for Measuring Program Comprehension. Ph.D. thesis, School of Computer Science, University of Magdeburg (2012)
62. Sjøberg, D., Hannay, J., Hansen, O., Kampenes, V.B., Karahasanovic, A., Liborg, N.K., Rekdal, A.: A Survey of Controlled Experiments in Software Engineering. IEEE Trans. Softw. Eng. **31**(9), 733–753 (2005)
63. Soloway, E., Ehrlich, K.: Empirical Studies of Programming Knowledge. IEEE Trans. Softw. Eng. **10**(5), 595–609 (1984)
64. Standish, T.: An Essay on Software Reuse. IEEE Trans. Softw. Eng. **SE–10**(5), 494–497 (1984)
65. Tiarks, R.: What Programmers Really Do: An Observational Study. In: Proc. Workshop Software Reengineering (WSR), pp. 36–37 (2011)

66. Torchiano, M.: Empirical Assessment of UML Static Object Diagrams. In: Proc. Int'l Workshop Program Comprehension (IWPC), pp. 226–230. IEEE CS (2004)
67. Vitharana, P., Ramamurthy, K.: Computer-Mediated Group Support, Anonymity, and the Software Inspection Process: An Empirical Investigation. IEEE Trans. Softw. Eng. **29**(2), 167–180 (2003)
68. Wechsler, D.: The Measurement of Adult Intelligence, third edn. American Psychological Association (1950)
69. Wohlin, C., Runeson, P., Höst, M., Ohlsson, M., Regnell, B., Wesslén, A.: Experimentation in Software Engineering: An Introduction. Kluwer Academic Publishers (2000)
70. Wundt, W.: Grundzüge der Physiologischen Psychologie. Engelmann (1874)