A study into the practice of reporting software engineering experiments

Kate Revoredo¹ · Djordje Djurica¹ · Jan Mendling²

Accepted: 21 June 2021 / Published online: 18 August 2021
© The Author(s) 2021

Abstract

It has been argued that reporting software engineering experiments in a standardized way helps researchers find relevant information, understand how experiments were conducted and assess the validity of their results. Various guidelines have been proposed specifically for software engineering experiments. The benefits of such guidelines have often been emphasized, but the actual uptake and practice of reporting have not yet been investigated since the introduction of many of the more recent guidelines. In this research, we utilize a mixed-method study design including sequence analysis techniques for evaluating to which extent papers follow such guidelines. Our study focuses on the four most prominent software engineering journals and the time period from 2000 to 2020. Our results show that many experimental papers miss information suggested by guidelines, that no de facto standard sequence for reporting exists, and that many papers do not cite any guidelines. We discuss these findings and implications for the discipline of experimental software engineering focusing on the review process and the potential to refine and extend guidelines, among others, to account for theory explicitly.

Keywords Guideline for software engineering experiments · Controlled experiments · Process mining · Method mining

Communicated by: Per Runeson

✉ Kate Revoredo
kate.revoredo@wu.ac.at

Djordje Djurica
djordje.djurica@wu.ac.at

Jan Mendling
jan.mendling@hu-berlin.de

¹ Wirtschaftsuniversität Wien, Welthandelsplatz 1, 1020 Vienna, Austria

² Humboldt-Universität zu Berlin, Unter den Linden 6, 10099 Berlin, Germany

1 Introduction

Reporting guidelines are an important concern for software engineering experiments.¹ Arguably, using reporting guidelines makes it easier for the reader understand the experimental design and the validity of the conclusions (Jedlitschka et al. 2008). These benefits have motivated the community to design and refine guidelines that support systematic and consistent reporting (Singer 1999; Wohlin et al. 2000; Juristo and Moreno 2001; Kitchenham et al. 2002; Shaw 2003; Jedlitschka et al. 2008).

Despite these efforts to establish standards, it has been observed that reporting in practice is often heterogeneous and important information is missing (Jedlitschka and Ciolkowski 2004; Sjøberg et al. 2005; Jedlitschka et al. 2008). Indeed, research on reporting guidelines has remained largely prescriptive. We know little about the extent to which reporting guidelines are used and how the uptake has changed over time. This might be because the number of controlled experiments has drastically increased since 2000 and that quantitative analysis of how these are reported is difficult. Still, gaining insights into the actual reporting practice is important to further improve guidelines and reporting practices.

In this paper, we investigate actual reporting practices for controlled experiments with human subjects that have been published in major software engineering journals during the period between the years 2000 and 2020. To this end, we use a mixed-method approach combining coding techniques from qualitative research with a formal analysis of event sequences from process mining. Our analysis reveals the actual reporting path of experiment papers and the degree of conformance for different journals over time. We find that conformance oscillates between 55% and 75% for all covered journals without a clear trend towards increasing. Our consecutive citation analysis shows that roughly one-third of the papers do not refer to any of the experiment reporting guidelines, while replication studies hardly ever refer to the guideline by Carver (2010) for replications. Based on the observed results, we highlight several implications for improving both reporting practices and refining guidelines, among others by more explicitly covering theory.

The remainder of this paper is structured as follows. Section 2 discusses the role of experiments in software engineering with a focus on reporting guidelines. We present hypotheses on the presumable impact of these guidelines. Section 3 describes our mixed-method research design, including paper selection, coding procedures, and analysis techniques. Section 4 presents our analysis results, focusing on the conformance between the reporting sequence of papers and guidelines. Section 5 discusses the implications of our research and threats to validity. Section 6 concludes with a summary and an outlook on future research.

2 Background

In this section, we first discuss the role of experiments in software engineering. Then, we revisit reporting guidelines for experiments in software engineering. Finally, we hypothesize how reporting practice could be expected to develop over time.

¹In the following, we refer to software engineering experiments, but use the term “experiment” without this explicit qualification for brevity.

2.1 Experiments in Software Engineering

Experiments are an important means for generating new scientific insights. Gauch (2003) highlights the strengths of experiments, including control and understanding of causal factors. For these reasons, experiments are also increasingly used in software engineering. Research by Basili (1993) and Basili (1996), Kitchenham et al. (2004), and Wohlin et al. (2000) and Wohlin et al. (2012) laid the foundations for developing the field of empirical software engineering. Basili (1993) and Basili (1996) emphasizes the benefits for software engineering to develop an experimental line of research similar to other scientific disciplines. Specifically, he proposes an experimental approach inspired by the quality improvement paradigm as often adapted in industrial development and engineering as the basis. Kitchenham et al. (2004) highlight the potential of adapting principles from evidence-based medicine to software engineering and discuss both benefits and difficulties of its adoption. Finally, Wohlin et al. (2012) provides an extensive summary of methodological principles for conducting software engineering experiments. They emphasize the importance of experiments, given that the practice of software engineering builds on the interactions between software artifacts and human behavior of software developers and other stakeholders.

Various reviews have investigated how and to which extent experiments are used in software engineering. Shull et al. (2004) emphasize the importance of replication for establishing reliable insights into software engineering. They present guidelines that are meant to support a better transfer of knowledge about experimental designs and execution.

Sjøberg et al. (2005) review controlled experiments in software engineering published between 1993 and 2002, focusing on how relevant information on subjects and tasks is reported. Their findings suggest that reporting is often incomplete and unsystematic, with partially inconsistent terminology. They state that the software engineering community needs guidelines helping researchers to better tackle difficulties of methodological and practical complexity of conducting controlled experiments. They provide a list of experimental details that they consider necessary to be reported. The review by Kampenes et al. (2007) drills down into the aspect of effect sizes. They observe that only 29% of the reviewed experiments reported effect sizes, even though this information is considered essential for interpreting experimental results. Additionally, Dybå et al. (2006) review the quantitative assessment of statistical power in software engineering experiments, which they find to be below established norms. They stress the importance of reporting confidence intervals and effect sizes. Hannay and Sjøberg (2007) review to which extent software engineering experiments build upon a theoretical justification of hypotheses. Their results reveal that out of 103 articles, only 23 build in total on 40 theories. These theories mainly were used for two reasons: to justify research questions and hypotheses as a part of the experimental design and to provide additional explanations of the results. The benefits of building theories and building on theories for software engineering experiments are stressed by Hannay and Sjøberg (2007). As an aid, they provide an extensive overview of the theories used in the papers that they reviewed.

It is important to note that the mentioned reviews stem from the years 2004 to 2007. The weaknesses they uncovered led to a refinement of guidelines for reporting software engineering experiments.

2.2 Experimental Reporting Guidelines in Software Engineering

Reporting has been a concern of research on software engineering experiments since the paper by Singer (1999), and there are several papers afterwards that describe reporting

guidelines. We provide a short description of these guideline papers and a comparison between them.

The first guideline paper by Singer (1999) introduced the APA style guidelines to the field of empirical software engineering. According to Singer (1999), the abstract should summarize the paper including information about hypotheses, population, and results. The paper itself should first present the study's problem and a brief explanation of the research strategy; describe in detail the experimental method, participants, and materials; outline the experimental procedure; and then present the statistical analysis of results with a discussion of the overall findings.

The second guideline of interest is proposed in the book by Wohlin et al. (2000). The authors emphasize the need for describing the scope of the experiment, its goals and foundational definitions. A summary of the experimental planning should cover the context selection for its importance to validity and generalization as well as the hypothesis formulation, including justifications for the selection of variables and subjects. Also the instrumentation is meant to be described. Among others, Wohlin et al. (2000) discuss what threats to validity have to be considered and how they can be addressed. The book also provides guidelines on analyzing data and interpreting results, together with suggestions for writing a summary and conclusion. It is worth mentioning that the second edition (Wohlin et al. 2012) refers to the reporting guideline by Jedlitschka et al. (2008).

Guidelines are also presented in the book by Juristo and Moreno (2001), later reprinted as Juristo and Moreno (2010). These guidelines are motivated by the idea that good experimental documentation needs to describe all phases, including goal definition, experiment design, execution, and analysis at a level of detail that a replication study can be conducted. To this end, the motivation should justify the relevance of the experiment, provide pointers to prior experiments, and describe experimental goals and hypotheses. The experimental design should report the experimental factors, response variables, parameters, blocks, as well as experimental units, subjects, and experimental process. Also, information should be provided about internal replication, randomization procedure if applicable, as well as subject knowledge, experimental schedule, and various factors that may have influenced the experimental result, such as potential learning and boredom effects. Regarding experimental execution, details are warranted about experimenters, instruction to participants, available time for completing the study, as well as experimental deviations and data collection. Finally, the analysis best includes information on data normality, analysis methods, results, and result interpretation.

The guideline by Kitchenham et al. (2002) presents a hands-on approach for reporting the experiment. It suggests describing the context of the experiment at extensive detail. Then, the experimental design is described regarding the planning of subjects, sampling techniques, and the process of allocating and administering treatments. Next, the procedures of experiment execution and data collection are summarized. This leads to the data analysis, presentation, and interpretation of the results, before the report concludes. We have to emphasize that this guideline presents a more general instruction on how to run an experiment, but lacks the instructions on how to report it precisely. The most extensive work on reporting are arguably the guidelines by Jedlitschka et al. (2008), which we will use as a reference in this paper for its level of detail. Note that these guidelines are fairly consistent with other guidelines, but more fine-granular. These guidelines suggest starting with the title and authorship section, which should include the term "controlled experiment" in the title. The structure of the abstract is inspired by abstracts in the medical sciences. The actual paper starts with the introduction section, including the problem statement, research

objectives and context. The related work discussion should summarize prior contributions as well as the technology under investigation and alternative technology. Next, the section on the planning and experimental design covers various aspects. These include research objectives; information on the sampling strategy, population, and sample size; experimental material, objects, tasks and subjects; subsections on hypotheses, experimental design, and the experimental procedure with details on data gathering; as well as a description of the analysis procedure. In turn, the section on the actual experiment execution is followed by the analysis section. Readers should be provided with descriptive statistics, data set preparation procedures, and hypothesis testing results. The discussion and interpretation section should cover results and implications, as well as threats to validity. The conclusion includes a summary and future work propositions.

Table 1 summarizes the guideline structure by Jedlitschka et al. (2008) together with its sections, subsections and a short description. The four columns on the right-hand side of this table describe which of its subsection are also considered by previous guidelines, namely [G2:] Singer (1999); [G3:] Wohlin et al. (2000); [G4:] Juristo and Moreno (2001); and [G5:] Kitchenham et al. (2002). Table 1 also highlights that the guidelines by Jedlitschka et al. (2008) are the most fine-granular ones, and they show substantial overlap with the activities of other guidelines. Table 1 marks those activities with a check if they are explicitly covered in the reporting structure. Furthermore, it is important to mention that there are guidelines that we did not include in this comparison. Jedlitschka et al. (2014) is an extension of Jedlitschka et al. (2008) suggesting the inclusion of additional information for practitioners, such as costs, quality schedule, and productivity in the context of software development. Although widely used in various research fields, we did not consider the introduction, methods, results, and discussion (IMRAD) guideline (Sollaci and Pereira 2004) as it is not specifically designed for software engineering experiments.

2.3 Propositions on the Evolution of Reporting Practices

In this section, we formulate a set of propositions that help us to define clear analysis objectives. Such an approach is consistent with general guidelines on conducting systematic literature reviews (Kitchenham and Charters 2007), which emphasize the need to formulate research questions and objectives. Investigating reporting practices for software engineering experiments is warranted, because the last larger review covering this aspect dates back to Sjöberg et al. (2005) and various proposals for reporting guidelines have been made since then.

Proposition 1 (Patterns) *We expect that patterns of reporting can be observed. Two arguments support this assumption. First, reporting an experiment is a human activity of an author team that becomes routinized by means of social entrainment (Gersick and Hackman 1990). This means that the same author team will likely organize the reporting of a new experiment in such a way as they have done for the previous one. Such persistence of behavior has been studied among others by Kelly and McGrath (1985). Second, experimental reporting is subject to social norms of the scientific process in a particular field. Social norms contribute to the development of recurring patterns of behavior (DiMaggio and Powell 1983). These are further stabilized by mimetic behavior (Gersick and Hackman 1990) of imitating reporting of published experiments in case authors do not have yet established their own way of reporting. All these aspects contribute to the emergence of reporting patterns.*

Table 1 Description of Jedlitschka et al. (2008) guideline (G1) and comparison with previous guidelines: G2: Singer (1999); G3: Wohlin et al. (2000); G4: Juristo and Moreno (2001); G5: Kitchenham et al. (2002)

Jedlitschka et al. (2008) guideline structure (G1)			G2	G3	G4	G5
Introduction	Problem statement	What is the problem? Where does it occur? Who has observed it?	✓	✓		
	Research objectives	What is the research question to be answered by this study?				
	Context	What information is necessary to understand whether the research relates to a specific situation (environment)?				✓
Background	Technology under investigation	Describe the technology that will be evaluated?				
	Alternative technology	How does this research relate to alternative technologies? What is the control treatment?				
	Related studies	How does it relate to state of the research?				
	Relevance to practice	How does it relate to state of the practice?				
Experiment Planning	Goals	Formalization of goals				✓
	Experimental units	From which population will the sample be drawn? How will the groups be formed?				✓
	Experimental material	Which objects are selected and why?				✓
	Tasks	Which tasks have to be performed by the subjects?				✓
	Hypotheses, parameters, variables	what are the constructs and their operationalization?				✓
	Design	What type of experimental design has been chosen?	✓	✓	✓	✓
	Procedure	How will the experiment be performed? What instruments, materials, tools will be used and how?	✓	✓	✓	✓
	Analysis Procedure	How will the data be analyzed?			✓	✓
Execution	Preparation	What has been done to prepare the execution of the experiment				✓
	Deviations	Describe any deviations from the plan			✓	✓
Analysis	Descriptive statistics	What are the results from descriptive statistics?	✓	✓	✓	✓
	Data set preparation	What was done to prepare the data set, why, and how?				
	Hypothesis testing	How was the data evaluated and was the analysis model validated?	✓	✓	✓	✓

Table 1 (continued)

Jedlitschka et al. (2008) guideline structure (G1)			G2	G3	G4	G5
Discussion	Evaluation of results and implications	Explain the results and the relation of the results to earlier research, especially those mentioned in the Background	✓	✓	✓	✓
	Threats to validity	How is validity of the experimental results assured?	✓			✓
	Inferences	Inferences drawn from the data to more general condition				
	Lessons learned	Which experience was collected during the course of the experiment				
Conclusions	Summarize	Concise summary of the research and its results	✓		✓	✓
	Impact	Description of impacts with regard to cost, schedule, and quality, circumstances under which the approach presumably will not yield the expected benefit	✓			✓
	Future work	What other experiments could be run to further investigate the results yielded?				

Proposition 2 (Conformance) *We expect that compliance with reporting guidelines can be observed. DiMaggio and Powell (1983) emphasize that normative pressure is a key factor that explains why organizations have been observed to be rather similar. These pressures are stronger in fields in which formal education and professional networks establish standards. Arguably, these attributes can be associated with empirical software engineering and reporting experiments in this field as well, contributing to compliance with reporting guidelines.*

Proposition 3 (Evolution) *We expect that the reporting patterns have evolved over time. We expect that this evolution is associated with two forces. First, reporting practices might have presumably become more similar over time. Similar observations have been made by Levitt and Nass (1989), who compared the topic sequence in leading textbooks in physics and sociology over time. Levitt and Nass (1989) explain their results with institutional forces, including coercive, mimetic, and normative pressures (DiMaggio and Powell 1983). Such forces are arguably also relevant for reporting software engineering experiments.*

Proposition 4 (Contingency) *We expect that papers with similar reporting can be observed. Burnes (1996) emphasizes that there is often “no one best way” of applying methods because contingent factors require an adaptation to circumstances. Similar observations have influenced situational method engineering (Brinkkemper 1996). This does not mean that reporting is arbitrary, but that differences are systematic and associated with contextual factors. As a consequence, we would be able to observe that certain types of papers would form clusters. Often, when there are hardly patterns overall (Proposition 1), it can still be possible to identify patterns for subgroups, which is investigated for this proposition.*

Proposition 5 (Factors) *We finally expect several factors to be associated with reporting practices. First, we expect that the awareness that papers exhibit concerning the discourse on reporting guidelines is associated with reporting practice. The weakest indication of such awareness is arguably the citation of a reporting guideline. Second, the specific setting of an experiment might have an impact on reporting. Presumably, replication studies might define a context in which specific reporting needs have to be considered Carver (2010).*

Next, we describe how we constructed our dataset with which we aim to investigate these propositions.

3 Method

In this section, we present the research design for investigating Propositions 1–5. To this end, we use a mixed-method approach that combines qualitative and quantitative research methods. More precisely, we apply a sequential mixed-method design (Venkatesh et al. 2013). We first conduct qualitative coding of experiment papers inspired by systematic mapping studies (Kitchenham and Charters 2007) and qualitative coding procedures (Saldaña 2015), which yields structured data that we analyze using computational methods (Berente et al. 2019), namely process mining (van der Aalst 2016) and method mining (Malinova et al. 2019).

We proceed as follows. Section 3.1 defines preliminary concepts that we make use of. Section 3.2 explains our paper selection procedure, and Section 3.3 how we coded the selected papers as event sequences. Section 3.4 describes the analysis techniques we applied, and Section 3.5 provides a summary of which technique is applied to investigate which proposition.

3.1 Preliminaries

Our research method builds on the overall idea that a paper describing a software engineering experiment can be represented as a sequence of sections, and that this sequence can be compared with reporting guidelines by the help of process mining techniques. To this end, we have to map a paper to a structured format representing this sequence of sections. We define this paper structure as follows.

The formal structure of a paper $\mathbb{P} = \langle s_1, \dots, s_n \rangle$ is a sequence of sections and subsections s_i . For all pairs of indexes $i, j \in \mathbb{N}$ with $i < j$, we say that s_i appears before section s_j in the sequence of the paper structure. Each section s_i includes content k_i . A requirement for our analysis is to progress from the formal structure of a paper with its section contents to a logical sequence that is aligned with reporting guidelines. Our dataset ($\mathbb{D}$) is composed of such logical sequences, each corresponding to one paper.

For our analysis, we build on analysis techniques from process mining. Therefore, we recall the classical notions of process mining: event, event sequence and event log. An event is a tuple $e = (c, a, t)$ where c is the case id, a is the activity (event type) and t is the timestamp imposing a temporal order over the events. An event sequence is defined as $\sigma = \langle e_1, \dots, e_{|\sigma|} \rangle$ of events such that $\forall i, j \in \{1, \dots, |\sigma|\} e_i.c = e_j.c$. An event log $\mathbb{L}$ is a multi-set $\{\sigma_1, \dots, \sigma_{|\mathbb{L}|}\}$ of sequences. In our dataset ($\mathbb{D}$), events represent content blocks that match an item of the reporting guidelines, activities define to which reporting activity a content block maps, and timestamps capture the order of how content blocks appear in the text of the paper. We define the alphabet A as the set of all activity types defined by the

Fig. 1 Example of a process model for the process of reporting an experiment

reporting guidelines of Jedlitschka et al. (2008). The content further describes an activity of the guidelines. In particular, we characterize each activity using a set of keywords. The keywords represent plausible terms to be used in the heading of a section. Therefore, an activity is described as a 5-tuple

$$a = (\text{group}, \text{label}, \text{keywords}, \text{description}, \text{required})$$

where *group* is the name of the set of related activity that *a* belongs to, *label* is the name of the activity, *keywords* is a set of terms that define the activity, *description* is a short text that describes its purpose and *required* indicates whether the activity is mandatory or not.

The logical sequence of activities defines the logical structure of reporting an experiment. Figure 1 shows this logical structure as a BPMN process model as an example describing the steps of reporting an experiment. Circles define the start and the end. Rectangles represent the activities and the arcs the sequence of the activities. Diamond shapes represent gateways indicating that consecutive activities can be performed in any order or mutually exclusive. In our example, the background for the experiment and the plan for the experiment can be reported in any order while the activity of defining the execution will only be included if there is a deviation in the experiment.

3.2 Paper Selection

We selected papers according to guidelines for systematic literature reviews (Kitchenham and Charters 2007). We focused on papers reporting controlled software engineering experiments with human participants. We selected papers from the four major software engineering journals with the highest impact factor: *Information and Software Technology* (IST), *IEEE Transactions on Software Engineering* (TSE), *Journal of Systems and Software* (JSS) and *Empirical Software Engineering* (ESS). We conducted a search directly on the publisher's repository for papers with the term "experiment" appearing either in the title or in the abstract and which were published between 2000 and 2020.² The choice for evaluating the 20 years since 2000 was made due to the fact that the first guideline for reporting controlled experiments was published by Singer (1999) in the year before. Therefore, only papers published after 1999 would have had the chance to report their experiments following a guideline.

We used only the term "experiment" in our query to guarantee high recall. We deliberately accepted the risk of including papers with this query that report on any type of experiment. We addressed the challenge of low precision by manually inspecting and

²We included all papers that were available via the publisher's repositories at the time when we closed the selection procedure, which was end of March 2020.

removing papers that (i) do not present controlled experiments with human participants and (ii) do not use an experimental design as a research method.

Table 2 shows the amount of papers retrieved using our search query (#Retrieved) and the number of papers remaining for analysis after the selection criteria were applied (#Analyzed). The last column shows the percentage of papers kept for analysis. It is worth mentioning that the journal Empirical Software Engineering has the highest percentage of kept papers. Our dataset of papers ($\mathbb{D}$) contains 168 papers. A list of all papers is included in Appendix A.

3.3 Coding Procedure

Next, we describe our coding procedure. Input to this procedure is a paper and its output is a structured representation of that paper's structure in terms of a sequence σ . Therefore, we sequentially process all sections and each respective subsection of the input paper in the order as they appear in the text.

Each section or subsection s of the paper is matched with reporting activities of the guideline of Jedlitschka et al. (2008). This matching is done by members of the author team using rules. First, the label terms of s are compared with the keywords of the different reporting activities. If an activity clearly matches, it is chosen. Second, if the label is not clear or ambiguous, the content of the section is read (Holsti 1969). As a result, a set of activities is identified or the section is ignored due to a lack of fitness with the guideline.

The coding of each section or subsection based on their content and the meaning associated with each reporting activity is critical for the validity of our study. Therefore, we adopted the procedure of Recker et al. (2019) and divided the coding into two phases. The first one addressed the consistency of the coding scheme, i.e. the definition of the activities, and the second one the coding itself.

In the first phase, a random sample of 40 papers (approximately 24% of the papers) was selected for defining the coding scheme using keywords and a description for each activity. The refinement of this initial coding scheme was done in four rounds. In each round, two authors coded ten papers, discussed the inconsistencies and improved the coding scheme. After the fourth round, no further refinements of the coding scheme were identified, which completed the first phase. In the second phase, the remaining set of papers was coded by one author.

As done by Recker et al. (2019), we calculated at each round of refinement the agreement between the two authors using Kappa as defined by Cohen (1960) as a measure of inter-coder reliability. Figure 2 depicts the Kappa analysis over the rounds. Figure 2(a) visualizes Kappa's improvement over the rounds with the final round achieving a Kappa of 0.91 indicating almost perfect agreement (Neuendorf 2002, p.145). Figure 2(b) also shows for each

Table 2 Number of papers retrieved for each of the journals and the number of those considered for analysis

Source	#Retrieved	#Analyzed	% Analyzed
Information and Software Technology	324	52	16.05%
IEEE Transactions on Software Engineering	197	35	17.77%
Journal of Systems and Software	458	27	5.90%
Empirical Software Engineering	173	54	31.21%
Total	1152	168	14.58%

Fig. 2 The graph in (a) shows the Kappa value evolution over the rounds. The table in (b) provides the details of the Kappa analyses: the value over the rounds and the total number of codings per round

round the value of Kappa and the number of codings done by each author (column *Rows*). Given that the quantity of sections varies from paper to paper, also the number of codings differs for each round. In the discussions between the authors, we kept track of the number of codes agreed between them. We calculated the number of correct codes for each author, i.e. the number of times their code was eventually chosen. The author with the best coding was chosen to do the rest of the coding of the dataset in the second phase.

Table 3 depicts the final set of activities. There is a one-to-one relation with the content of Jedlitschka et al. (2008) guideline showed in Table 1. The label of each activity is summarized together with its corresponding keywords. The indication whether an activity is mandatory or optional is determined by following the definition of required content presented in Jedlitschka et al. (2008) guideline. Sections considered required in the guideline are mandatory activities, while sections not required are optional. The required contents *Abstract* and *Keyword* of Jedlitschka et al. (2008) guideline are considered as mandatory activities in our approach with labels *Define Abstract* and *Define Keywords*. They were both omitted from Tables 1 and 3 respectively because of space restrictions. Thus, 29 distinct activities are considered. Among them 19 are required and 10 are optional.

As a final step of data preparation, we merged consecutive activities of the same type into one. For instance, a sequence as *<Define Goals, Design Experiment, Design Experiment, Interpret Results, Interpret Results, Interpret Results, Summarize Findings>* is compressed to *<Define Goals, Design Experiment, Interpret Results, Summarize Findings>*. The reduction of the total number of activities in the event log was 16%.

3.4 Applied Process Mining Techniques

We followed the described procedures and obtained a dataset $\mathbb{D}$ that codes a set of papers using the data structure of an event log as used in process mining. For this reason, various analysis techniques from process mining can be readily applied. Such analysis can be used to investigate to which extent reporting guidelines are considered in each paper, which patterns of reporting and which changes of patterns over time can be observed.

Next, we describe the analysis techniques that we include in our analysis. Sections 3.4.1 and 3.4.2 describe automatic process discovery and conformance checking, respectively. Section 3.4.3 explains how we use clustering techniques.

Table 3 The list of activities considered with the final set of keywords (*Define Abstract* and *Define Keywords* omitted for space restrictions)

Group	Label	Keywords	Required?
Introduction	Define Problem	Problem; Problem statement; Issue; Importance; Obstacle; Dispute; Dilemma; This study is concerned with.	✓
	Define Objectives	Research objective; Objective; Goal; Purpose; Aim of this study.	✓
	Define Context	Context; Application type; Application domain; Participant; Subject; Time constraint; Surroundings; Setting; Location; Impact; External factor; Internal factor.	✓
Background	Define Researched Technology	Related work; Other authors; Related; Similar; Technology; Processes; Methods; Technology used.	✓
	Define Alternative Technology	Related work; Other authors; Related; Similar; Technology; Processes; Methods; Alternative.	✓
	Define Related Studies	Related work; Other authors; Related; Similar; Technology; Processes; Methods; Empirical evaluation; Empirical study.	
	Define Relevance to Practice	Related work; Relevance to practice; Real scenario; Real domain.	
Experiment Planning	Define Goals	Goal; Objective; Quality focus; Aims	✓
	Define Experiment Groups	Experiment groups; Subject; Participant; Population; Unit; Group; Sample; Sampling; Sample size.	✓
	Define Experiment Materials	Experiment material; Object; Material; Characteristic; Impact; Work book.	✓
	Define Tasks	Task; Activity; Assignment; Performance; Schedule.	✓
	Define Hypotheses	Hypothesis; Variable; Control; Proposition; Research model.	✓
	Design Experiment	Design; Experimental design; Criteria; Description; Tool; Schedule; Operation; Training.	✓
	Define Procedure	Procedure; Flow; Procedure; Schedule; Task; Participant; Subject; Length; Operation; Operationalization; Instrumentation; Tool.	✓
	Define Analysis Procedure	Analysis procedure; Statistics; Test; Analysis; Data; Tool; Result; Instrumentation.	✓
Execution	Define Experiment Preparation	Execution; Deviation; Execution of experiment; Process; Conduct.	
	Define Experiment Deviations	Execution; Deviation; Execution of experiment; Process; Conduct.	

Table 3 (continued)

Group	Label	Keywords	Required?
Analysis	Explore Data	Result Analysis; Result; Description; Statistics; Quantitative; Qualitative; Analysis; Data; Descriptive statistics; Correlation.	✓
	Prepare Data	Analysis; Data preparation; Data processing; Data cleaning; Data reduction.	
	Test Hypotheses	Hypothesis testing; Test; Hypothesis; Analysis; Validation.	
Discussion	Interpret Results	Evaluation of result; Evaluation of result; Result of the study; Evaluation of findings.	
	Assess Threats to Validity	Threats to validity; Internal validity; External validity; Evaluating validity; External or internal factor; Limitation.	✓
	Infer Results	Inference; Prediction; Inference statistics.	✓
Conclusions	Define Lessons Learned	Lessons learned; Experience acquired.	
	Summarize Findings	Summary; Conclusion; Findings; Discussion.	✓
	Summarize Impacts	Summary; Impact.	
	Outline Future Work	Further work; Future plan; Future process; Future research.	

3.4.1 Process discovery

Process discovery takes an event log as an input and automatically generates a process model representing the sequences of that event log. Figure 3 describes how process discovery works using a simple artificial example. The starting point is the event log shown in Fig. 3(a). It contains three different sequences of activities. Process discovery algorithms construct a process model from such an event log based on behavioral relations between the activities. All sequences have the same two initial activities (*Define Goals*, *Design Experiment*). This pattern is reflected in the output model by including a sequence of these two activities as a mandatory flow. The subsequent behavior is different for the three sequences. Discovery algorithms spot that the first and the second sequence execute the same two activities (*Explore Data*, *Test Hypothesis*), but in a different order and that the third sequence includes a third activity instead (*Interpret Results*). These observations are reflected in the model by exclusive and parallel gateways, respectively, creating different flow options. The suffix is the same for all three sequences and therefore final activity *Summarize Findings* is included as mandatory. Figure 3.(b) shows the discovered model.

Event logs from practice are far more complex than this illustrative example. This implies the challenge of representing the behavior compactly and, specifically, a trade-off between: (i) fitness: the discovered model should allow for the behavior observed in the event log; (ii) precision (avoid underfitting): the discovered model should not allow behavior that was

Fig. 3 Illustration of an event log (a) and the discovered process model (b)

not observed in the event log; (iii) generalization (avoid overfitting): the discovered model should generalize the observed behavior seen in the event log; (iv) simplicity: the discovered model should be as simple as possible (van der Aalst 2016).

If the sequences are similar in terms of their behavior, the derived process model will have high fitness and a clear and simple structure. However, if the set of sequences are substantially different in terms of order and activities, the discovered model is often chaotic; it is also referred to as a spaghetti model (van der Aalst 2016). These spaghetti models are hard to analyze and show the lack of pattern in execution.

In this study, we use process discovery techniques to investigate Proposition 1, i.e. to which extent common reporting patterns exist in the considered papers on software engineering experiments. More specifically, we use the tool Disco³, a widely used commercial process mining tool, to discover a process model from the event log of our dataset (Section 3.3).

3.4.2 Conformance Checking

Conformance checking techniques provide insights into the extent of consistency between a process model and the sequences of an event log. They take an event log and a model as input and identify the extent to which the event sequences deviate from the model. Several techniques exist, e.g. through replaying each sequence against the process model or by calculating an alignment (van der Aalst et al. 2012). Given a sequence and a process model, an alignment maps the event sequence to the process model's best fitting run.

Consider, for instance, the process model in Fig. 3(b). If the event log includes the sequence $\langle \text{Define Goals, Design Experiment, Summarize Findings} \rangle$, a conformance check algorithm will report that the activities *Interpret Results* or *Explore Data* together with *Test Hypothesis* are not observed in the log. In the same way, if a sequence like $\langle \text{Define Goals, Design Experiment, Interpret Results, Define Lessons Learned, Summarize Findings} \rangle$ is observed, conformance checking reports that an unexpected activity (*Define Lessons Learned*) was executed.

³<https://fluxicon.com/disco/>

Fig. 4 Process model capturing the list of activities depicted in Table 3

In our study, we use conformance checking techniques to investigate Proposition 2, i.e. to analyze to which extent software engineering experiments follow the guidelines proposed by Jedlitschka et al. (2008). To that end, we manually created a process model based on these guidelines. Since a paper is written in a sequential way, the list of activities in Table 3 defines a sequence. Optional activities are modeled within XOR-gateway blocks. Figure 4 shows the corresponding process model. We use this model for checking the conformance between reporting sequences of individual papers and the guidelines by Jedlitschka et al. (2008). We calculate conformance with plugin *Replay a Log On Petri Net For Conformance Analysis*⁴ (van der Aalst et al. 2012) of Prom⁵.

Also, we use conformance checking and its evolution over time to evaluate Proposition 3, and partially for Proposition 5 in combination with potential factors associated with reporting practice, such as citation of guidelines and replication.

3.4.3 Cluster Analysis

Cluster analysis allows the identification of groups of sequences in an event log. Two sequences are put into the same cluster if they are similar in terms of a suitable distance function. Various techniques for calculating distances between sequences have been proposed for social sequence analysis (Abbott 1995; Gabadinho et al. 2011) and process mining (Song et al. 2008; De Koninck et al. 2017).

In the context of our work, we use cluster analysis to address Proposition 4, i.e. to investigate whether there are different recurrent patterns for reporting experiments. It is not clear how many clusters can be expected. If all papers considered the reporting guideline structure of Jedlitschka et al. (2008), we would obtain one cluster containing very similar reporting sequences. In case that papers arbitrarily reported experiments, we might obtain a high number of rather dissimilar sequence clusters. It is more plausible to expect only few clusters. In that case, it will be interesting to investigate which are representative sequences for each cluster and in how far they differ.

⁴<https://fddocuments.net/document/replay-a-log-on-petri-net-for-conformance-analysis-plugin-inpdf.html>

⁵<https://www.promtools.org/doku.php>

Table 4 Analysis techniques used to investigate Propositions 1-5

Proposition	Technique	Tool
1. Pattern	Process Discovery	Disco
2. Conformance	Conformance Checking	Prom ⁷ with plugin <i>Replay a Log On Petri Net For Conformance Analysis</i> ⁸
3. Evolution	Conformance checking with statistical analysis over the range of the years of publication	Prom with plugin <i>Replay a Log On Petri Net For Conformance Analysis</i>
4. Contingency	Cluster Analysis	TraMineR ⁹
5. Factors	Conformance Checking with statistical analysis over guidelines citation and experimental replication	Prom with plugin <i>Replay a Log On Petri Net For Conformance Analysis</i>

⁷<https://www.promtools.org/doku.php>⁸<https://fddocuments.net/document/replay-a-log-on-petri-net-for-conformance-analysis-plugin-inpdf.html>⁹<http://traminer.unige.ch/index.shtml>

We use the TraMineR tool⁶ (Gabadinho et al. 2011) for our cluster analysis, an R-package for exploring sequence data. For calculating the sequence distance, we used the optimum matching algorithm (Abbott and Tsay 2000).

3.5 Propositions and Corresponding Techniques

Table 4 summarizes Propositions 1-5, the corresponding analysis techniques that we apply for investigating them, and the corresponding tools used. In Fig. 5, we indicate the input and output for each of the analysis techniques.

4 Results

This section describes the results of our study into reporting practices of software engineering experiments. Section 4.1 provides descriptive statistics of our dataset. Section 4.2 presents the results of analyzing the data using automatic process discovery. Section 4.3 discusses the conformance checking results, which provide insights into how well aligned the articles are with reporting guidelines. Section 4.4 describes the results of clustering articles according to their reporting sequences. Section 4.5 evaluates to which extent is citing guidelines connected with guideline conformance. Finally, Section 4.6 presents observations on how replication studies use reporting guidelines.

4.1 Descriptive statistics

Our event log contains 168 cases (each describing a paper and the sequence of its reporting steps) from the year 2000 until 2020. Figure 6 shows the temporal distribution of these papers for each of the four journals. For every year, there are three or more papers in our analysis.

⁶<http://traminer.unige.ch/index.shtml>

Fig. 5 Research strategies and applied methods

Table 5 shows log statistics about the activities before ($\#Activities$) and after compressing consecutive activities ($\#Activities_c$). The table provides the number of activities per paper (maximum (Max), minimum (Min), and average (Avg)) and the number of distinct activities in the whole log (maximum (Max), minimum (Min), and average (Avg)). It is interesting to note that we did not encounter any sequence that occurred more than once, which means that every paper's reporting sequence was unique. TSE was the only journal, in which not all 29 reporting activities were observed. Three optional activities were missing: *Define Experiment Deviation*, *Define Experiment Preparation*, and *Define Relevance to Practice*. Another interesting observation is that the average of distinct activities per paper is less than the number of required activities (19) for all journals. Furthermore, there are

Fig. 6 Distribution of the papers over time

Table 5 Log statistics

<i>Journal</i>	<i>#Papers</i>	<i>#Activities</i>	<i>#Activities_c</i>	<i>#PerPaper_{Max}</i>	<i>#PerPaper_{Max_c}</i>	<i>#PerPaper_{Min}</i>	<i>#PerPaper_{Min_c}</i>	<i>#PerPaper_{Avg}</i>	<i>#PerPaper_{Avg_c}</i>	<i>#Distinct</i>	<i>#Distinct_{Min}</i>	<i>#Distinct_{Max}</i>	<i>#Distinct_{Avg}</i>
IST	52	1244	1064	40	32	11	10	24	20	29	10	22	18
TSE	35	909	754	45	40	17	12	26	22	26	12	21	17
JSS	27	688	559	35	28	18	14	25	21	29	11	28	18
ESE	54	1564	1291	51	40	17	15	29	24	29	14	21	18
All	168	4405	3668	51	40	11	10	26	22	29	10	28	18

several activities that are repeated in an average paper, i.e. the number of distinct activities per paper ($\#Distinct_{Avg}$) varies from the number of activities per paper ($\#Per\ Paper_{Avg}$). Table 6 shows each activity's frequency in the event log (Log Frequency) and in how many papers each activity appears (Paper Frequency). Also, the corresponding percentage is presented.

4.2 Process discovery

To check if the papers follow recurring patterns of reporting, we applied automatic process discovery for the complete event log using the tool Disco. Figure 7 shows the process model discovered from the set of all papers. In this model, all possible paths are shown. We observe that the complexity of this model is overwhelming, and that it is difficult to spot patterns of recurring behavior.

Table 6 Frequency of each activity in the entire event log and per paper

Activity Label	Log Frequency		Paper Frequency	
Explore Data	350	9.54%	158	94.04%
Test Hypotheses	237	6.46%	124	73.81%
Interpret Results	228	6.22%	142	84.52%
Design Experiment	197	5.37%	142	84.52 %
Assess Threats To Validity	180	4.91%	158	94.04%
Define Problem	177	4.83%	167	99.40%
Summarize Findings	173	4.72%	164	97.62%
Define Objectives	172	4.69%	164	97.62%
Define Abstract	168	4.58%	168	100%
Define Keywords	168	4.58%	168	100%
Define Experiment Groups	160	4.36%	144	85.71%
Define Procedure	158	4.31%	128	76.19%
Define Hypotheses	156	4.25%	131	77.98%
Outline Future Work	154	4.20%	149	88.69%
Define Researched Technology	137	3.74%	117	69.64%
Define Related Studies	133	3.63%	119	70.83%
Define Experiment Materials	126	3.44%	111	66.07%
Define Analysis Procedure	126	3.44%	103	61.31%
Define Context	120	3.27%	112	66.67%
Define Goals	102	2.78%	94	55.95%
Define Tasks	98	2.67%	87	51.79%
Summarize Impacts	49	1.34%	45	26.79%
Define Lessons Learned	33	0.90%	32	19.05%
Define Alternative Technology	22	0.60%	22	13.10%
Prepare Data	12	0.33%	11	6.55%
Infer Results	12	0.33%	11	6.55%
Define Experiment Deviations	8	0.19%	7	4.17%
Define Experiment Preparation	7	0.19%	7	4.17%
Define Relevance To Practice	5	0.14%	5	2.98%

Fig. 7 Process model discovered from the complete event log. All possible paths are represented

The two activities that all the papers consider and that have a clear position in this spaghetti model are the *Define Abstract* and *Define Keywords* activities at the top of the model (not readable in the figure). This might probably not be due to the guidelines, but that paper submission formally enforces the inclusion of abstract and keywords. Therefore, it is not surprising that they are observed in all papers and in this order.

Once we apply filtering techniques provided by Disco to only show the minimum number of paths for connecting all 29 activities, we obtain the process model shown in Fig. 8. Compared with the spaghetti model from Fig. 7, this model is easier to understand and interpret. The darker the color of an activity, the more often this activity occurred in the event log. The Activity *Explore data* is the most observed activity, occurring 350 times (due to repetitions in various papers). The thicker the transition arrows, the more often the corresponding path is observed in the log. The most frequent sequence of activities is from activity *Explore data* to activity *Test Hypotheses* with 204 occurrences (due to repetitions in various papers). In this filtered process model, the frequency associated with an activity is greater or equal to the sum of its outgoing transition arrows frequency, because not all the possible arrows with its correspondent frequencies are shown. This process model shows that papers usually start with the definition of the problem (99% of the 168 papers) followed by the definition of the experiment's objectives (97%). We also notice that many activities and many transitions are only observed for a smaller fraction of papers.

4.3 Conformance Checking

Conformance checking is a group of techniques that facilitates the comparison between the sequences represented in a process model (such as reporting guidelines) and sequences of papers observed in our event log. We conducted such a conformance check for each paper based on the process model shown above in Fig. 4 that captures the reporting guidelines by (Jedlitschka et al. 2008). We used the classical notion of fitness as a measure of conformance (van der Aalst 2016). A sequence fully conforming with the process model has a fitness of 1 while a sequence that does not conform at all has a fitness of 0. We summarize the results for each journal separately and in total. Figure 9 shows the Box plot of this conformance analysis.

The bulk of papers range between 0.6 and 0.7 in terms of conformance. Given that the data is normally distributed, we performed a one-way ANOVA test with no assumption of equal variances. The difference between the mean value of the four journals is statistically significant with 95% of confidence ($F = 6.1574$, num df = 3, denom df = 74.535, p-value = 0.0008497). The journal with the highest average conformance is ESE. This is not surprising given that it is the journal with the highest affinity with controlled experiments. It also has to be noted that we do not observe drastic differences in conformance between the journals. The Box plot of Fig. 9 also highlights some outliers either with outstanding conformance or

Fig. 8 Process model discovered from the complete event log considering the minimum path for connecting all activities

Fig. 9 Conformance analysis of each journal independently and all journal together

very low conformance. The two papers with the highest conformance are from JSS and the third-highest from TSE. All three papers explicitly cited guidelines with two of them citing Jedlitschka et al. (2008) guideline. JSS and TSE are also the journals of the two papers with the lowest conformance of below 0.4. These papers did not cite any guidelines.

We also analyzed the evolution of the conformance over the 20 years in which the papers were published. Figure 10(a) shows a Box plot of the conformance of all papers for each year. We observe a slight increase in the average until the year 2008 when Jedlitschka et al. (2008) was published. Figure 10(b) shows the evolution of the average conformance over the years for each journal and also for the event log with all the papers. All journals show a similar evolution without any clear upward or downward trend over the years. More specifically, we do not observe any noticeable change after the year 2008 when Jedlitschka et al. (2008) was published. The peak of the curve for JSS in 2009 stems from the fact that only one experiment paper was published in that year in this journal and that this paper is an outlier with the highest guideline conformance of the whole set of papers. In summary, Fig. 10(a) and b show the same range of average conformance between 0.6 and 0.7 that we already observed in Fig. 9. Also, the KPSS test (Kwiatkowski et al. 1992) showed that the conformance time series is stationary with 0.05 significance (KPSS Level = 0.16735, Truncation lag parameter = 4, p-value = 0.1) without any clear trend up or down.

4.4 Cluster Analysis

For the cluster analysis, we use the TraMiner tool (Gabadinho et al. 2011). This tool supports clustering based on classical sequence alignment. This means, in essence, that sequences are clustered based on a notion of sequence edit distance. Different number of clusters were evaluated and the best result yielded four clusters. Figure 11 shows the four clusters of sequences. The X-axis represents the position (A) in a sequence. It is scales to 41, which is the maximum number of activities stemming from the paper with the longest sequence

Fig. 10 Conformance analysis of all papers over the years

Fig. 11 Clusters identified by using TramineR

(also compared above in Table 5). The Y-axis represents the relative frequency of color-coded activities. Mind that the area covered by a color bears the relevant information, not necessarily the position of a color dot. Rows in the diagram should not be interpreted as individual papers. The red color corresponds to missing values, which are appended to fill up shorter sequences up to a length of 41. Also, each cluster has the information of how many sequences were grouped. Cluster 1 has 56 sequences, Cluster 2 has 47, Cluster 3 has 45 and Cluster 4 has 20 sequences.

All clusters have low Average Silhouette Width (0.2, 0.09, 0.12, 0.03, respectively) indicating that there is no clear structure of the clusters (Kaufman and Rousseeuw 1990). Also, there are 4, 11, 6 and 5 representative sequences respectively considering a coverage threshold of 25%. The absence of a unique representative sequence for the clusters underlines that the clusters are unstructured. However, it is possible to spot some commonalities. Given that the average length of a sequence is 22, which is almost half of the longest sequence, all clusters have a larger red area indicating missing values. We also observe that all four clusters have a similar start pattern that corresponds to the sequence *Define Abstract*, *Define Keywords* followed by *Define Problem* and *Define Objectives*.

There are also some noticeable differences. Cluster 1 contains the on average shortest sequences. It is also the cluster with the smallest number of explicit *Test Hypothesis* activities. Also less frequent than in the other clusters is the activity *Design Experiment*. The average fitness of this cluster with reporting guidelines is 0.69. In contrast to Cluster 1, Cluster 3 has longer sequences. Most of them range between 20 and 25, while the previous cluster ranges between 15 and 18. Cluster 3 also has a larger relative frequency of *Test Hypothesis* and *Assess Threats to Validity*. Still, the overall proportion of these activities is still low. All this suggests that Cluster 3 includes papers of a more structured reporting style. The average fitness is highest for this cluster with 0.73.

Cluster 2 appears to be the cluster with a rather homogeneous group of papers with a rather structured reporting style. There is ample attention to *Define Task*, *Design Experiment*, and *Explore Data* together with a clear methodological focus on *Test Hypothesis* and *Assess Threats to Validity*. In contrast to the heterogeneous Cluster 4, the papers in Cluster 3 seem to have a clear focus, which results in a detailed, yet compact reporting style. The average fitness of Cluster 2 is 0.70, and for Cluster 4 it is 0.62.

4.5 Guideline Citation Analysis

This section follows up on the question of whether an explicit consideration of guidelines leads to a more structured reporting style. To this end, we analyze to which extent papers cite established guidelines. First, Section 4.5.1 investigates how many papers cite reporting guidelines, including Jedlitschka et al. (2008), and whether this is connected with conformance. Second, Section 4.5.2 analyzes which papers have been citing Jedlitschka et al. (2008).

4.5.1 Guideline Citation of Software Engineering Experiment Papers

One potential explanation of higher conformance with reporting guidelines is whether reporting guideline papers were considered at all. We observed above that the conformance of papers with the guidelines by Jedlitschka et al. (2008) ranges between 0.6 and 0.7. The focus of our analysis here is on three questions: (i) is Jedlitschka et al. (2008) guideline (G1a) or its previous version (Jedlitschka and Pfahl 2005) (G1b) cited? (ii) is the book by Wohlin et al. (2012) (G1c) that points to Jedlitschka et al. (2008) cited? (iii) does a paper consider any other guideline by Singer (1999) (G2), Wohlin et al. (2000) (G3), Juristo and Moreno (2001) (G4), and Kitchenham et al. (2002) (G5)?

We analyzed for all 168 papers which guidelines they cite. As a result we found that 57 papers (34%) do not cite any guideline at all. The remaining 111 papers cite at least one guideline. Table 7 shows the number of citations each guideline received. The percentage is calculated considering the total number of papers that potentially could have cited the respective guideline. This means that a paper published in the year or onward of the publication year of the guideline is a paper that could have potentially cited it.

The guideline most cited is the book by Wohlin et al. (2000) with almost 43% of the potential papers referring to it. Jedlitschka et al. (2008) received 17.5% of the potential papers; however, the book by Wohlin et al. (2012) recommends Jedlitschka et al. (2008) as a guideline and received almost double the citations. Figure 12 presents a comparison of the number of citations that Wohlin et al. (2000), Jedlitschka et al. (2008) and Wohlin et al.

Table 7 Number of citations over the total number of possible citations for each guideline

Journal	G1a	G1b	G1c	G2	G3	G4	G5
IST	4/39	5/43	10/23	0/52	24/52	7/52	12/50
TSE	3/20	1/24	5/12	0/35	12/35	8/32	4/30
JSS	3/16	1/22	4/14	0/27	12/27	5/26	5/25
ESE	11/45	3/52	7/33	0/54	24/54	11/53	13/53
#	21/120	10/141	26/82	0/168	72/168	31/163	34/158
%	17.5	7.1	31.7	0	42.9	19	21.5

#G1a, #G1c and #G3

Fig. 12 Citation comparison between Jedlitschka et al. (2008) (G1a), Wohlin et al. (2012)(G1c) and Wohlin et al. (2000) (G3) guidelines

(2012) received over time. We observe that from 2014 when papers started citing Wohlin et al. (2012), the citations of Jedlitschka et al. (2008) and Wohlin et al. (2012) evolved roughly in parallel to each other, and citation for Wohlin et al. (2000) decreased.

Figure 13 shows a Box plot comparison of the fitness between those papers that cited at least one of the guidelines (YES) and those that did not cite any guidelines at all (NO). We can observe a slightly higher fitness of those papers that cited guidelines. We performed Welch t-test (Ruxton 2006). The difference between the mean value of the two groups is

Fig. 13 Conformance checking comparison between all the papers that cited any of the guidelines and the papers that did not cite any of the guidelines

statistically significant with 95% of confidence ($t = -4.1471$, $df = 132.02$, $p\text{-value} = 5.987e - 05$), showing a correlation between conformance and guideline citation.

Also, 30% of the papers cite only one guideline. Among them, only 7 papers cite either Juristo and Moreno (2001) or Kitchenham et al. (2002) guidelines. The remaining papers cite versions of the Jedlitschka et al. (2008) guidelines (G1a,G1b or G1c) or the Wohlin et al. (2000) guidelines. A pairwise correlation analysis between Jedlitschka et al. (2008) guideline and the other guidelines (G2,G3,G4 and G5) shows a correlation with G2 and no correlation with the others (Pearson’s Chi-squared test with 0.05 of significance). A correlation between the four guidelines (G2,G3,G4 and G5) was also not found. Thus, it was not possible to find a pattern on the citation of guidelines.

4.5.2 Citation from Various Types of Papers

It is surprising that Jedlitschka et al. (2008) is highly cited, but the share of software engineering experiment papers citing it is rather low with 17.5%. Therefore, we investigate which type of papers cite Jedlitschka et al. (2008).

A search on Google Scholar yielded a list of 300 citations. These citations are distributed as follows: 100 conference papers, 103 journal papers and 97 other types of publication, such as books, dissertations and theses. We investigated more deeply into the journal and conference papers and could classify the papers in *Method* (M), *Experiment* (E), or *Survey* (S) papers. Table 8 (a) shows the statistics of papers per category. We notice that the majority of the papers are from the category Experiment (E). Considering the papers published in one of the four journals, we observed that 35 are experiments. We analyzed the 21 papers that cite Jedlitschka et al. (2008). The difference of 14 papers that we did not consider here corresponds to the papers out of our study’s scope (experiment with human participants in which the experiment is the main contribution of the paper). This means, a larger fraction of citations to Jedlitschka et al. (2008) are conference papers and articles that are not published in IST, TSE, JSS, or ESE.

Table 8 Number of journal and conference papers per category (a)

Table 8 Number of journal and conference papers per category				
(a)				
	#Journal	#Conference	#	
M	23	28	51	
E	63	59	122	
S	17	13	30	
#	103	100	203	
(b)				
	#E	#M	#S	#
IST	9	1	3	13
TSE	5	0	0	5
JSS	6	2	2	10
ESE	15	1	2	18
#	35	4	7	46

Number of papers from the four journals considered in this research per category (b)

Number of papers from the four journals considered in this research per category (b)

Table 9 Number of replications for each of the journals and their percentage

Source	#Analysed	#Replication	% Replication
IST	52	10	19.2%
TSE	35	3	8.6%
JSS	27	1	3.7%
ESE	54	11	20.4%
Total	168	25	14.9%

4.6 Replication Analysis

Replication is one of the principles of the scientific method (Brown et al. 2016). Replication studies are an important means to further strengthen the trust in the validity of results obtained by the original experiment. It has been emphasized that a replication study has to describe how the original studies were performed, provide implementation differences of the replication, and a comparison of the results (Carver 2010). It is also important to describe the type of replication (Bettis et al. 2016) and to consider a shared terminology (Baldassarre et al. 2014). Due to these specific aspects, replication experiments require a separate structure of reporting. Table 9 shows the number of replication papers that are part of our analysis.

IST and ESE are the journals with more replication papers. This is consistent with the results of a recent systematic mapping study on replication (Cruz et al. 2020). In their mapping study, Cruz et al. (2020) emphasize the lack of guidelines for reporting replications. They state that guideline proposal by Carver (2010) is the only high-level proposal for reporting replication so far. This lack of guidelines might also explain that we found that nine of the replications were internal replications published together with the original study. Twenty out of the twenty-five replication papers were published after the year 2010 and therefore could have followed Carver's guidelines (Carver 2010). However, only 9 did (45%).

Figure 14 shows to which extent the replication papers conform with Jedlitschka et al. (2008). The fitness of the replication papers is comparable to other papers. The difference between the mean value of the two groups is not statistically significant with 95% of confidence based on Welch t-test (Ruxton 2006) ($t = -0.22898$, $df = 29.114$, $p\text{-value} = 0.8205$), showing there is not a correlation between conformance and replication.

4.7 Summary of Results

We summarize the results of our mix-method study as follows. The descriptive analysis revealed that a consistent reporting style is only followed to a limited extent. The variance is high, with some papers only reporting a fraction of the required reporting activities, others repeating activities, and only a few showing high fitness with reporting guidelines. The application of process mining techniques confirmed the impression of a rather inconsistent reporting practice. Conformance checking revealed that the fitness between the reporting sequences of most papers only partially considers the guidelines by Jedlitschka et al. (2008). The clustering analysis suggested that only Cluster 2 appears to consider most of the important activities. The conformance analysis over time showed that there is no consistent improvement of reporting practice over time. These findings are consistent with our citation analysis, revealing that the highly cited guidelines by Jedlitschka et al. (2008) are often not

Fig. 14 Conformance checking comparison between papers that report a replication (YES) and papers that report an original study (NO)

considered in experiments published in IST, TSE, JSS, and ESE. Furthermore, while replication studies have lower fitness due to special reporting requirements, we observe that they exhibit conformance similar to papers reporting original studies.

5 Discussion

In this section, we first discuss our findings in the light of the formulated propositions. Then, we describe implications for research on software engineering experiments. Finally, we reflect upon threats to validity of our study.

5.1 Propositions and Findings

We formulated several propositions that motivated our study. We discuss the findings for each of them in turn.

Proposition 1 (Patterns) *described the expectation that clear reporting patterns can be identified. We actually found that the process model that we automatically discovered from our event log of all papers is overwhelmingly complex and that not even two papers exhibit the same sequence of reporting activities (Section 4.2). We argued in support of Proposition 1 that routinization of author teams (Gersick and Hackman 1990) and social norms enforcing reporting practices (DiMaggio and Powell 1983) would lead to recurring reporting patterns. As apparently, patterns exist only to a limited extent, there is doubt whether routinization and social norms are effectively in place.*

Proposition 2 (Conformance) *argued that compliance of reporting with established reporting guidelines could be expected. We found that measures of conformance checking are observed in a middle range between 0.6 and 0.7 (Section 4.3). Apparently, institutional*

pressures are not so strong, with the effect that it is possible to publish experiments in the best software engineering journals without strictly following established guidelines.

Proposition 3 (Evolution) *formulated the expectation that reporting practices would have evolved over time. Our findings suggest that an increasing uptake of guidelines has not happened (Section 4.3). There is at least no evidence of an increase in fitness with established guidelines. This observation holds for the overall period from 2000–2020, and also the publication year of the guideline by Jedlitschka et al. (2008) does not seem to have made a difference. According to arguments formulated by Levitt and Nass (1989) on the similarity of textbooks in scientific areas, this could be interpreted that the overall discipline has not matured and converged in terms of reporting.*

Proposition 4 (Contingency) *argued that similar reporting might be observable for subgroups of papers. What we found is that the clusters that we identified are largely connected with the level of guideline compliance (Section 4.4). This suggests that the differences are unsystematic.*

Proposition 5 (Factors) *brought differences of reporting in connection with different factors. We found that papers citing reporting guidelines are more conforming (Section 4.5) and that the factor of whether an experiment is a replication does not imply any difference in fitness (Section 4.6). Also, considering the reporting of effect sizes (Kampenes et al. 2007) as a proxy for quality, the results showed no correlation between quality and conformance with a guideline. These observations suggest that differences in reporting are unsystematic, which means that the only explanation for differences that we found is whether established reporting guidelines were explicitly considered or not.*

5.2 Implications for Reporting Software Engineering Experiments

Our findings have various implications for further developing reporting practices of software engineering experiments.

First, we were surprised by the observation that *variation in reporting sequences is large* and that 57 of the 168 analyzed papers do not refer to any guideline at all. This finding can be interpreted as that there is low institutional pressure (DiMaggio and Powell 1983) to adhere to experiment reporting guidelines in software engineering journals. This raises the question of why that might be the case. It may be speculated that this could be connected with the limited coverage of empirical methods in software engineering curricula. The IEEE Software Engineering Body of Knowledge (Version 3) at least covers experiments and empirical methods on eight pages, however, only in the recently added 15th and last chapter (Bourque et al. 2014). This is already an improvement as compared to the previous version published by Abran et al. (2004) that did not mention empirical methods at all. A direction for improving this situation could be to use more extensively method experts for reviewing experiment papers next to subject-matter experts. Guidelines are available and recognized such as the ones by Jedlitschka et al. (2008), later endorsed by Wohlin et al. (2012), and they can be also helpful guidelines during the review process. It is surprising that many papers are incomplete even in those parts where the guidelines are easy to comply with, like having a structured introduction and conclusion, and an explicit discussion of threats to validity.

Second, our findings also raise the question of whether some *potential weaknesses of guidelines* could explain at least a share of the non-conformance. We observe that the

description of hypotheses and variables can be found in different sections, although Jedlitschka et al. (2008) suggest always presenting them as part of the experimental planning section. Also, not many papers used the execution section to describe deviations. The ones that used this section were often misinterpreting its purpose and describing an experimental procedure without providing any details about experiment deviations. Potentially, such misinterpretations can be addressed by sharpening the section's title or its description. But it must be stated in defense of Jedlitschka et al. (2008) that it is also on the authors to carefully read and reflect upon the purpose of each reporting activity.

Third, guidelines like the ones by Jedlitschka et al. (2008) should be further evaluated regarding their *completeness*. Indeed, as our study design focuses largely on the conformance between Jedlitschka et al. (2008) and individual papers, we cannot observe directly to which extent authors felt the need to report experiment-related content that was not covered by these guidelines. We recognize the increasing importance of theorizing for empirical research in software engineering (Hannay and Sjøberg 2007; Wohlin et al. 2015; Ralph 2018; Ralph et al. 2021). In the neighboring field of information systems research, theorizing is given precedence over research design (Recker 2012), and experimental guidelines emphasize the importance of theory for justifying the overall research design (Burton-Jones et al. 2009). The reporting promoted by Jedlitschka et al. (2008) might be too restricted to provide appropriate space and attention for theorizing. A theoretical discussion would be a useful addition to the background section, which in Jedlitschka et al. (2008) only includes research technology, alternative technology, related studies, and relevance to practice. Future research, for instance, an interview-based study on guideline usage, can help to leverage a deeper understanding of guideline usage and its challenges.

5.3 Threats to Validity

As for any empirical study, threats to validity have to be considered for our study. We focus on the conclusion, construct and external validity. As our design does not single out causality, any threats to internal validity of correlational studies apply. Beyond the reflection that follows, we made a replication package available⁷, which is meant to help other researchers to criticize and to build upon our arguments.

Conclusion validity is defined as a degree to which the conclusions we reach about the relationship of the study's data are considered reasonable Trochim and Donnelly (2001). Frequent threats to conclusion validity emerge from the erroneous interference of relationships that do not have the backing of the data, measurements, and analysis results (Trochim and Donnelly 2001). Our study established a correlation argument related to the connection between the publication of reporting guidelines for software engineering experiments and the practice of reporting. Various weaknesses of correlational studies in terms of conclusion validity, such as ones previously mentioned, apply. However, correlational studies such as surveys are appropriate for exploring, describing, and explaining a phenomenon (Recker 2012) such as experimental reporting in software engineering as in our study. We iterated our coding protocol to address potential threats to conclusion validity until we achieved high inter-rater reliability in terms of Kappa (0.91). Furthermore, we conducted several in-depth analyses on our sample, which gives us confidence that the conclusions drawn from the data are credible and reasonable. A limitation of our

⁷The replication package can be found under DOI: [10.5281/zenodo.4559141](https://doi.org/10.5281/zenodo.4559141)

design is the lack of insights into the rationale why authors and reviewers do not adapt or enforce guidelines more strongly. A qualitative research agenda is needed to understand this phenomenon better.

Construct Validity is relevant for how we constructed our dataset (Wohlin et al. 2012). Although we adopted established procedures for coding research papers for literature reviews as applied by Recker et al. (2019), we can identify three construct validity threats related to the measurement of conformance between papers and guidelines. The first one is how sections of the paper describe what their respective titles promise to do. This threat can be mitigated by reading and coding the section contents. We refrained to code according to the explicit formal structure of sections, which means that relevant content is not visible to our coding procedure if it is not emphasized in a section title. The second threat is that papers might have cited guidelines other than the ones considered in our study. We focused on guidelines from software engineering. If guidelines from other fields would be found useful by authors, these could serve as a basis for revising software engineering guidelines.

Finally, the third threat relates to the situation in which the reporting guideline is cited but not followed. This issue can be addressed by performing conformance checking on the subset of the papers reported using at least one guideline.

External Validity is concerned with the generalization of the findings Trochim and Donnelly (2001). In turn, a threat to external validity points to limitations of generalizing the results to a larger context. Our study sampled papers that reported experiments from the four most prestigious software engineering journals published between 2000 and 2020. A possible limitation is that our study does not include experiment papers published in other software engineering journals and presented at conferences during the observed period. As our citation analysis of Jedlitschka et al. (2008) guideline in Section 4.5.2 has shown, this guideline is cited in other experiment journals and conference papers that may be potentially associated to the scope of this study. This restricts our findings and conclusions to the selected journals and the chosen time frame. However, we can still argue that our results are relevant for the whole field of software engineering experiments, because highly ranked journals can be regarded as best-in-class examples for other journals and conferences. Another threat to external validity is that we manually selected papers for inclusion in our study. To tackle this, we have followed guidelines by Kitchenham and Charters (2007) and defined formal criteria to make this manual selection process transparent.

6 Conclusion

This paper has presented findings of a mixed-method study into the reporting practice of controlled experiments published in major software engineering journals over the period from 2000 to 2020. To this end, we combined coding techniques from qualitative research with process mining techniques. We find that (1) many experimental papers in top software engineering journals do not report all of the information that published guidelines suggest to be required, (2) there is no de facto standard sequence for reporting these items, (3) many experimental papers do not cite any reporting guidelines, and (4) that these findings, on average, hold for both early as well as for recent papers.

Our findings have important implications both for the practice of reporting software engineering experiments and for guideline design. First, we discussed the potential of more explicitly using guidelines during the review process. Second, we also suggest partially

refining them and for giving the theory a more explicit place. Some of the excellent experiment papers that consider reporting guidelines diligently can nicely serve as role models for future experiment papers in software engineering.

There are several important questions for which our research design did not offer an answer. First, a central point of discussion during the review process was the connection between adherence to guidelines and the paper's research quality. By focusing on the four top journals only, we assumed that quality must have been assured in the review process. The variation in guideline conformance leaves doubt and calls for a research strategy that builds on the expert judgment of research quality. Second, our analysis does not offer insights into the motives of authors to either adopt or ignore reporting guidelines. A qualitative research strategy is required to investigate this point further. We hope that our appendix and the replication package corresponding to this paper will help to address these points in future research.

Appendix A: Papers used in the analysis

In this section we list all 168 papers we used in our analysis grouped by journal.

A.1 Papers from the ESE journal

1. Brooks, A., Utbult, F., Mulligan, C., & Jeffery, R. (2000). Early lifecycle work: influence of individual characteristics, methodological constraints, and interface constraints. *Empirical Software Engineering*, 5(3), 269-285.
<https://doi.org/10.1023/A:1026594716872>
2. Burkhardt, J. M., Détienne, F., & Wiedenbeck, S. (2002). Object-oriented program comprehension: Effect of expertise, task and phase. *Empirical Software Engineering*, 7(2), 115-156.
<https://doi.org/10.1023/A:1015297914742>
3. Anda, B., & Sjøberg, D. I. K. (2005). Investigating the role of use cases in the construction of class diagrams. *Empirical Software Engineering*, 10(3), 285-309.
<https://doi.org/10.1007/s10664-005-1289-3>
4. Verelst, J. (2005). The influence of the level of abstraction on the evolvability of conceptual models of information systems. *Empirical Software Engineering*, 10(4), 467-494.
<https://doi.org/10.1007/s10664-005-3863-0>
5. Zettel, J. (2005). Methodology support in CASE tools and its impact on individual acceptance and use: A controlled experiment. *Empirical Software Engineering*, 10(3), 367-394.
<https://doi.org/10.1007/s10664-005-1287-5>
6. Maldonado, J. C., Carver, J., Shull, F., Fabbri, S., Dória, E., Martimiano, L., Mendonça, M., & Basili, V. (2006). Perspective-based reading: A replicated experiment focused on individual reviewer effectiveness. *Empirical Software Engineering*, 11(1), 119-142.
<https://doi.org/10.1007/s10664-006-5967-6>
7. Natt Och Dag, J., Thelin, T., & Regnell, B. (2006). An experiment on linguistic tool support for consolidation of requirements from multiple sources in market-driven

- product development. *Empirical Software Engineering*, 11(2), 303-329.
<https://doi.org/10.1007/s10664-006-6405-5>
8. Bunse, C. (2006). Using patterns for the refinement and translation of UML models: A controlled experiment. *Empirical Software Engineering*, 11(2), 227-267.
<https://doi.org/10.1007/s10664-006-6403-7>
 9. Knodel, J., Muthig, D., & Naab, M. (2008). An experiment on the role of graphical elements in architecture visualization. *Empirical Software Engineering*, 13, 693-726.
<https://doi.org/10.1007/s10664-008-9069-5>
 10. Babar, M. A., Kitchenham, B., & Jeffery, R. (2008). Comparing distributed and face-to-face meetings for software architecture evaluation: A controlled experiment. *Empirical Software Engineering*, 13(1), 39-62.
<https://doi.org/10.1007/s10664-007-9052-6>
 11. Budgen, D., Kitchenham, B. A., Charters, S. M., Turner, M., Brereton, P., & Linkman, S. G. (2008). Presenting software engineering results using structured abstracts: A randomised experiment. *Empirical Software Engineering*, 13(4), 435-468.
<https://doi.org/10.1007/s10664-008-9075-7>
 12. Cruz-Lemus, J. A., Genero, M., Manso, M. E., Morasca, S., & Piattini, M. (2009). Assessing the understandability of UML statechart diagrams with composite states-A family of empirical studies. *Empirical Software Engineering*, 14(6), 685-719.
<https://doi.org/10.1007/s10664-009-9106-z>
 13. Gresse Von Wangenheim, C., Thiry, M., & Kochanski, D. (2009). Empirical evaluation of an educational game on software measurement. *Empirical Software Engineering*, 14(4), 418-452.
<https://doi.org/10.1007/s10664-008-9092-6>
 14. De Lucia, A., Oliveto, R., & Tortora, G. (2009). Assessing IR-based traceability recovery tools through controlled experiments. *Empirical Software Engineering*, 14(1), 57-92.
<https://doi.org/10.1007/s10664-008-9090-8>
 15. Sfetsos, P., Stamelos, I., Angelis, L., & Deligiannis, I. (2009). An experimental investigation of personality types impact on pair effectiveness in pair programming. *Empirical Software Engineering*, 14(2), 187-226.
<https://doi.org/10.1007/s10664-008-9093-5>
 16. El-Attar, M., & Miller, J. (2009). A subject-based empirical evaluation of SSUCD's performance in reducing inconsistencies in use case models. *Empirical Software Engineering*, 14(5), 477-512.
<https://doi.org/10.1007/s10664-008-9101-9>
 17. De Lucia, A., Gravino, C., Oliveto, R., & Tortora, G. (2010). An experimental comparison of ER and UML class diagrams for data modeling. *Empirical Software Engineering*, 15(5), 455-492.
<https://doi.org/10.1007/s10664-009-9127-7>
 18. Hofman, R. (2011). Behavioral economics in software quality engineering. *Empirical Software Engineering*, 16(2), 278-293.
<https://doi.org/10.1007/s10664-010-9140-x>
 19. Budgen, D., Burn, A. J., & Kitchenham, B. (2011). Reporting computing projects through structured abstracts: A quasi-experiment. *Empirical Software Engineering*, 16(2), 244-277.
<https://doi.org/10.1007/s10664-010-9139-3>

20. Martens, A., Koziolok, H., Prechelt, L., & Reussner, R. (2011). From monolithic to component-based performance evaluation of software architectures : A series of experiments analysing accuracy and effort. *Empirical Software Engineering*, 16, 587-622. <https://doi.org/10.1007/s10664-010-9142-8>
21. Wnuk, K., Höst, M., & Regnell, B. (2012). Replication of an experiment on linguistic tool support for consolidation of requirements from multiple sources. *Empirical Software Engineering*, 17(3), 305-344. <https://doi.org/10.1007/s10664-011-9174-8>
22. Kosar, T., Mernik, M., & Carver, J. C. (2012). Program comprehension of domain-specific and general-purpose languages: Comparison using a family of experiments. *Empirical Software Engineering*, 17(3), 276-304. <https://doi.org/10.1007/s10664-011-9172-x>
23. Feigenspan, J., Kästner, C., Apel, S., Liebig, J., Schulze, M., Dachzelt, R., Papendieck, M., Leich, T., & Saake, G. (2013). Do background colors improve program comprehension in the #ifdef hell? *Empirical Software Engineering*, 18, 699-745. <https://doi.org/10.1007/s10664-012-9208-x>
24. Jurkiewicz, J., Nawrocki, J., Ochodek, M., & Głowacki, T. (2013). HAZOP-based identification of events in use cases: An empirical study. *Empirical Software Engineering*, 20(1), 82-109. <https://doi.org/10.1007/s10664-013-9277-5>
25. Polančič, G., Jošt, G., & Heričko, M. (2013). An experimental investigation comparing individual and collaborative work productivity when using desktop and cloud modeling tools. *Empirical Software Engineering*, 20(1), 142-175. <https://doi.org/10.1007/s10664-013-9280-x>
26. Itkonen, J., & Mäntylä, M. V. (2014). Are test cases needed? Replicated comparison between exploratory and test-case-based software testing. *Empirical Software Engineering*, 19(2), 303-342. <https://doi.org/10.1007/s10664-013-9266-8>
27. Reinhartz-Berger, I., & Sturm, A. (2014). Comprehensibility of UML-based software product line specifications A controlled experiment. *Empirical Software Engineering*, 19, 678-713. <https://doi.org/10.1007/s10664-012-9234-8>
28. Apa, C., Dieste, O., Espinosa G., E. G., & Fonseca C., E. R. (2014). Effectiveness for detecting faults within and outside the scope of testing techniques: An independent replication. *Empirical Software Engineering*, 19(2), 378-417. <https://doi.org/10.1007/s10664-013-9267-7>
29. Hanenberg, S., Kleinschmager, S., Robbes, R., Tanter, É., & Stefik, A. (2014). An empirical study on the impact of static typing on software maintainability. *Empirical Software Engineering*, 19, 1335-1382. <https://doi.org/10.1007/s10664-013-9289-1>
30. Siegmund, J., Kästner, C., Liebig, J., Apel, S., & Hanenberg, S. (2014). Measuring and modeling programming experience. *Empirical Software Engineering*, 19(5), 1299-1334. <https://doi.org/10.1007/s10664-013-9286-4>
31. Fucci, D., & Turhan, B. (2014). On the role of tests in test-driven development: A differentiated and partial replication. *Empirical Software Engineering*, 19(2), 277-302. <https://doi.org/10.1007/s10664-013-9259-7>

32. Guerrouj, L., Di Penta, M., Guéhéneuc, Y. G., & Antoniol, G. (2014). An experimental investigation on the effects of context on source code identifiers splitting and expansion. *Empirical Software Engineering*, 19(6), 1706-1753.
<https://doi.org/10.1007/s10664-013-9260-1>
33. Albayrak, Ö., & Carver, J. C. (2014). Investigation of individual factors impacting the effectiveness of requirements inspections: A replicated experiment. *Empirical Software Engineering*, 19(1), 241-266.
<https://doi.org/10.1007/s10664-012-9221-0>
34. Wu, W., Serveaux, A., Guéhéneuc, Y. G., & Antoniol, G. (2015). The impact of imperfect change rules on framework API evolution identification: an empirical study. *Empirical Software Engineering*, 20(4), 1126-1158.
<https://doi.org/10.1007/s10664-014-9317-9>
35. Afzal, W., Ghazi, A. N., Itkonen, J., Torkar, R., Andrews, A., & Bhatti, K. (2015). An experiment on the effectiveness and efficiency of exploratory testing. *Empirical Software Engineering*, 20(3), 844-878.
<https://doi.org/10.1007/s10664-014-9301-4>
36. Mäder, P., & Egyed, A. (2015). Do developers benefit from requirements traceability when evolving and maintaining a software system? *Empirical Software Engineering*, 20(2), 413-441.
<https://doi.org/10.1007/s10664-014-9314-z>
37. Tu, Y. C., Tempero, E., & Thomborson, C. (2016). An experiment on the impact of transparency on the effectiveness of requirements documents. *Empirical Software Engineering*, 21, 1035-1066.
<https://doi.org/10.1007/s10664-015-9374-8>
38. Fernández-Sáez, A. M., Genero, M., Caivano, D., & Chaudron, M. R. V. (2016). Does the level of detail of UML diagrams affect the maintainability of source code?: a family of experiments. *Empirical Software Engineering*, 21(1), 212-259.
<https://doi.org/10.1007/s10664-014-9354-4>
39. Grigera, J., Garrido, A., Panach, J. I., Distant, D., & Rossi, G. (2016). Assessing refactorings for usability in e-commerce applications. *Empirical Software Engineering*, 21(3), 1224-1271.
<https://doi.org/10.1007/s10664-015-9384-6>
40. King, J., Stallings, J., Riaz, M., & Williams, L. (2017). To log, or not to log: using heuristics to identify mandatory log events - a controlled experiment. *Empirical Software Engineering*, 22(5), 2684-2717.
<https://doi.org/10.1007/s10664-016-9449-1>
41. Johanson, A. N., & Hasselbring, W. (2017). Effectiveness and efficiency of a domain-specific language for high-performance marine ecosystem simulation: a controlled experiment. *Empirical Software Engineering*, 22(4), 2206-2236.
<https://doi.org/10.1007/s10664-016-9483-z>
42. Sakhini, V., Mich, L., & Berry, D. M. (2017). Group versus individual use of power-only EPMcreate as a creativity enhancement technique for requirements elicitation. *Empirical Software Engineering*, 22(4), 2001-2049.
<https://doi.org/10.1007/s10664-016-9475-z>
43. Jbara, A., & Feitelson, D. G. (2017). How programmers read regular code: a controlled experiment using eye tracking. *Empirical Software Engineering*, 22(3), 1440-1477.
<https://doi.org/10.1007/s10664-016-9477-x>

44. Tosun, A., Dieste, O., Fucci, D., Vegas, S., Turhan, B., Erdogmus, H., Santos, A., Oivo, M., Toro, K., Jarvinen, J., & Juristo, N. (2017). An industry experiment on the effects of test-driven development on external quality and productivity. *Empirical Software Engineering*, 22(6), 2763-2805.
<https://doi.org/10.1007/s10664-016-9490-0>
45. Riaz, M., King, J., Slankas, J., Williams, L., Massacci, F., Quesada-López, C., & Jenkins, M. (2017). Identifying the implied: Findings from three differentiated replications on the use of security requirements templates. *Empirical Software Engineering*, 22(4), 2127-2178.
<https://doi.org/10.1007/s10664-016-9481-1>
46. Scanniello, G., Gravino, C., Genero, M., Cruz-Lemus, J. A., Tortora, G., Risi, M., & Doderio, G. (2018). Do software models based on the UML aid in source-code comprehensibility? Aggregating evidence from 12 controlled experiments. *Empirical Software Engineering*, 23(5), 2695-2733.
<https://doi.org/10.1007/s10664-017-9591-4>
47. Kosar, T., Gaberc, S., Carver, J. C., & Mernik, M. (2018). Program comprehension of domain-specific and general-purpose languages: replication of a family of experiments using integrated development environments. *Empirical Software Engineering*, 23(5), 2734-2763.
<https://doi.org/10.1007/s10664-017-9593-2>
48. Przybyłek, A. (2018). An empirical study on the impact of AspectJ on software evolvability. In *Empirical Software Engineering*, 23, 2018-2050.
<https://doi.org/10.1007/s10664-017-9580-7>
49. Ricca, F., Torchiano, M., Leotta, M., Tiso, A., Guerrini, G., & Reggio, G. (2018). On the impact of state-based model-driven development on maintainability: a family of experiments using UniMod. *Empirical Software Engineering*, 23(3), 1743-1790.
<https://doi.org/10.1007/s10664-017-9563-8>
50. Hannebauer, C., Hesenius, M., & Gruhn, V. (2018). Does syntax highlighting help programming novices? *Empirical Software Engineering*, 704.
<https://doi.org/10.1145/3180155.3182554>
51. McChesney, I., & Bond, R. (2019). Eye tracking analysis of computer program comprehension in programmers with dyslexia. *Empirical Software Engineering*, 24, 1109-1154.
<https://doi.org/10.1007/s10664-018-9649-y>
52. Salman, I., Turhan, B., & Vegas, S. (2019). A controlled experiment on time pressure and confirmation bias in functional software testing. *Empirical Software Engineering*, 24, 1727-1761.
<https://doi.org/10.1007/s10664-018-9668-8>
53. Amálio, N., Briand, L., & Kelsen, P. (2020). An experimental scrutiny of visual design modelling: VCL up against UML+OCL. *Empirical Software Engineering*, 25(2), 1205-1258.
<https://doi.org/10.1007/s10664-019-09784-9>
54. Viticchié, A., Regano, L., Basile, C., Torchiano, M., Ceccato, M., & Tonella, P. (2020). Empirical assessment of the effort needed to attack programs protected with client/server code splitting. *Empirical Software Engineering*, 25(1), 1-48.
<https://doi.org/10.1007/s10664-019-09738-1>

A.2 Papers from the IST journal

1. Calero, C., Piattini, M., & Genero, M. (2001). Empirical validation of referential integrity metrics. *Information and Software Technology*, 43(15), 949–957. [https://doi.org/10.1016/S0950-5849\(01\)00202-6](https://doi.org/10.1016/S0950-5849(01)00202-6)
2. Thelin, T., Runeson, P., & Regnell, B. (2001). Usage-based readingan experiment to guide reviewers with use cases. *Inf. Softw. Technol.*, 43(15), 925–938. [https://doi.org/10.1016/S0950-5849\(01\)00201-4](https://doi.org/10.1016/S0950-5849(01)00201-4)
3. Sabaliauskaite, G., Matsukawa, F., Kusumoto, S., & Inoue, K. (2003). Further investigations of reading techniques for object-oriented design inspection. *Information and Software Technology*, 45(9), 571–585. [https://doi.org/10.1016/S0950-5849\(03\)00044-2](https://doi.org/10.1016/S0950-5849(03)00044-2)
4. Kamsties, E., von Knethen, A., & Reussner, R. (2003). A controlled experiment to evaluate how styles affect the understandability of requirements specifications. *Information and Software Technology*, 45(14), 955–965. [https://doi.org/10.1016/S0950-5849\(03\)00098-3](https://doi.org/10.1016/S0950-5849(03)00098-3)
5. Sabaliauskaite, G., Kusumoto, S., & Inoue, K. (2004). Assessing defect detection performance of interacting teams in object-oriented design inspection. *Information and Software Technology*, 46(13), 875–886. <https://doi.org/10.1016/j.infsof.2004.03.004>
6. Snook, C. F., & Harrison, R. (2004). Experimental comparison of the comprehensibility of a Z specification and its implementation in Java. *Information and Software Technology*, 46(14), 955–971. <https://doi.org/10.1016/j.infsof.2004.04.003>
7. Pfahl, D., Laitenberger, O., Ruhe, G., Dorsch, J., & Krivobokova, T. (2004). Evaluating the learning effectiveness of using simulations in software project management education: results from a twice replicated experiment. *Information and Software Technology*, 46(2), 127–147. [https://doi.org/10.1016/S0950-5849\(03\)00115-0](https://doi.org/10.1016/S0950-5849(03)00115-0)
8. George, B., & Williams, L. (2004). A structured experiment of test-driven development. *Information and Software Technology*, 46(5), 337–342. <https://doi.org/10.1016/j.infsof.2003.09.011>
9. Glezer, C., Last, M., Nachmany, E., & Shoval, P. (2005). Quality and comprehension of UML interaction diagrams-an experimental comparison. *Information and Software Technology*, 47(10), 675–692. <https://doi.org/10.1016/j.infsof.2005.01.003>
10. Kabeli, J., & Shoval, P. (2005). Comprehension and quality of analysis specifications—a comparison of FOOM and OPM methodologies. *Information and Software Technology*, 47(4), 271–290. <https://doi.org/10.1016/j.infsof.2004.08.009>
11. Ardimento, P., Baldassarre, M. T., Caivano, D., & Visaggio, G. (2006). Assessing multiview framework (MF) comprehensibility and efficiency: A replicated experiment. *Information and Software Technology*, 48(5), 313–322. <https://doi.org/10.1016/j.infsof.2005.09.010>
12. Abraha, S., & Poels, G. (2007). Experimental evaluation of an object-oriented function point measurement procedure. 49, 366–380. <https://doi.org/10.1016/j.infsof.2006.06.001>

13. Bipp, T., Lepper, A., & Schmedding, D. (2008). Pair programming in software development teams – An empirical study of its benefits. *Information and Software Technology*, 50(3), 231–240.
<https://doi.org/10.1016/j.infsof.2007.05.006>
14. Webb, R., Hevner, A. R., Walton, G. H., & Linger, R. C. (2008). The impacts of function extraction technology on program comprehension : A controlled experiment. 50, 1165–1179.
<https://doi.org/10.1016/j.infsof.2008.04.001>
15. Colosimo, M., Lucia, A. De, Scanniello, G., & Tortora, G. (2009). Evaluating legacy system migration technologies through empirical studies. *Information and Software Technology*, 51(2), 433–447.
<https://doi.org/10.1016/j.infsof.2008.05.012>
16. Ricca, F., Torchiano, M., Penta, M. Di, Ceccato, M., & Tonella, P. (2009). Using acceptance tests as a support for clarifying requirements: A series of experiments. *Information and Software Technology*, 51(2), 270–283.
<https://doi.org/10.1016/j.infsof.2008.01.007>
17. Eichelberger, H., & Schmid, K. (2009). Guidelines on the aesthetic quality of UML class diagrams. *Information and Software Technology*, 51(12), 1686–1698.
<https://doi.org/10.1016/j.infsof.2009.04.008>
18. Opdahl, A. L., & Sindre, G. (2009). Experimental comparison of attack trees and misuse cases for security threat identification. *Information and Software Technology*, 51(5), 916–932.
<https://doi.org/10.1016/j.infsof.2008.05.013>
19. Nugroho, A. (2009). Level of detail in UML models and its impact on model comprehension: A controlled experiment. *Information and Software Technology*, 51(12), 1670–1685.
<https://doi.org/10.1016/j.infsof.2009.04.007>
20. Huang, L., & Holcombe, M. (2009). Empirical investigation towards the effectiveness of Test First programming. *Information and Software Technology*, 51(1), 182–194.
<https://doi.org/10.1016/j.infsof.2008.03.007>
21. Olav, F., Inge, A., & Arisholm, E. (2009). Improving the effectiveness of root cause analysis in post mortem analysis : A controlled experiment. 51, 150–161.
<https://doi.org/10.1016/j.infsof.2008.02.003>
22. Inge, A., & Arisholm, E. (2009). The effect of task order on the maintainability of object-oriented software. *Information and Software Technology*, 51(2), 293–305.
<https://doi.org/10.1016/j.infsof.2008.03.005>
23. Madeyski, L. (2010). The impact of Test-First programming on branch coverage and mutation score indicator of unit tests: An experiment. *Information and Software Technology*, 52(2), 169–184.
<https://doi.org/10.1016/j.infsof.2009.08.007>
24. Jørgensen, M. (2010). Identification of more risks can lead to increased over-optimism of and over-confidence in software development effort estimates. *Information and Software Technology*, 52(5), 506–516.
<https://doi.org/10.1016/j.infsof.2009.12.002>
25. Ochodek, M., Alchimowicz, B., Jurkiewicz, J., & Nawrocki, J. (2011). Improving the reliability of transaction identification in use cases. *Information and Software Technology*, 53(8), 885–897.
<https://doi.org/10.1016/j.infsof.2011.02.004>

26. Nguyen, V., Boehm, B., & Danphitsanuphan, P. (2011). A controlled experiment in assessing and estimating software maintenance tasks. *Information and Software Technology*, 53(6), 682–691.
<https://doi.org/10.1016/j.infsof.2010.11.003>
27. Cruz-Lemus, J. A., Genero, M., Caivano, D., Abrahão, S., Insfrán, E., & Carsí, J. A. (2011). Assessing the influence of stereotypes on the comprehension of UML sequence diagrams: A family of experiments. *Information and Software Technology*, 53(12), 1391–1403.
<https://doi.org/10.1016/j.infsof.2011.07.002>
28. Ciglaric, M. (2011). Impact of test-driven development on productivity , code and tests : A controlled experiment. 53, 557–573.
<https://doi.org/10.1016/j.infsof.2011.02.002>
29. Teruel, M. A., Navarro, E., López-Jaquero, V., Montero, F., Jaen, J., & González, P. (2012). Analyzing the understandability of Requirements Engineering languages for CSCW systems: A family of experiments. *Information and Software Technology*, 54(11), 1215–1228.
<https://doi.org/10.1016/j.infsof.2012.06.001>
30. Ng, T. H., Yu, Y. T., Cheung, S. C., & Chan, W. K. (2012). Human and program factors affecting the maintenance of programs with deployed design patterns. *Information and Software Technology*, 54(1), 99–118.
<https://doi.org/10.1016/j.infsof.2011.08.002>
31. Mäntylä, M. V., & Itkonen, J. (2013). More testers – The effect of crowd size and time restriction in software testing. *Information and Software Technology*, 55(6), 986–1003.
<https://doi.org/10.1016/j.infsof.2012.12.004>
32. Martínez, Y., Cachero, C., & Meliá, S. (2013). MDD vs. traditional software development: A practitioner’s subjective perspective. *Information and Software Technology*, 55(2), 189–200.
<https://doi.org/10.1016/j.infsof.2012.07.004>
33. Hadar, I., Reinhartz-Berger, I., Kuflik, T., Perini, A., Ricca, F., & Susi, A. (2013). Comparing the comprehensibility of requirements models expressed in Use Case and Tropos: Results from a family of experiments. *Information and Software Technology*, 55(10), 1823–1843.
<https://doi.org/10.1016/j.infsof.2013.05.003>
34. Karpati, P., Redda, Y., Opdahl, A. L., & Sindre, G. (2014). Comparing attack trees and misuse cases in an industrial setting. *Information and Software Technology*, 56(3), 294–308.
<https://doi.org/10.1016/j.infsof.2013.10.004>
35. Fernández-sáez, A. M., Genero, M., Chaudron, M. R. V, Ramos, I., Fernández-sáez, A. M., Genero, M., & Ramos, I. (2014). Are Forward Designed or Reverse-Engineered UML Diagrams More Helpful for Code Maintenance ?: A Family of Experiments. *INFORMATION AND SOFTWARE TECHNOLOGY*.
<https://doi.org/10.1016/j.infsof.2014.05.014>
36. Lytra, I., Gaubatz, P., & Zdun, U. (2015). Two controlled experiments on model-based architectural decision making. *Information and Software Technology*, 63, 58–75.
<https://doi.org/10.1016/j.infsof.2015.03.006>
37. Felderer, M., & Herrmann, A. (2015). Manual test case derivation from UML activity diagrams and state machines: A controlled experiment. *Information and Software*

- Technology, 61, 1–15.
<https://doi.org/10.1016/j.infsof.2014.12.005>
38. Ignacio, J., España, S., Dieste, Ó., Pastor, Ó., & Juristo, N. (2015). In search of evidence for model-driven development claims : An experiment on quality , effort , productivity and satisfaction. *INFORMATION AND SOFTWARE TECHNOLOGY*, 62, 164–186.
<https://doi.org/10.1016/j.infsof.2015.02.012>
 39. Häser, F., Felderer, M., & Breu, R. (2016). Is business domain language support beneficial for creating test case specifications: A controlled experiment. *Information and Software Technology*, 79, 52–62.
<https://doi.org/10.1016/j.infsof.2016.07.001>
 40. Marculescu, B., Poulding, S., Feldt, R., Petersen, K., & Torkar, R. (2016). Tester interactivity makes a difference in search-based software testing: A controlled experiment. *Information and Software Technology*, 78, 66–82.
<https://doi.org/10.1016/j.infsof.2016.05.009>
 41. Sedrakyan, G., Poelmans, S., & Snoeck, M. (2017). Assessing the influence of feedback-inclusive rapid prototyping on understanding the semantics of parallel UML statecharts by novice modellers. *Information and Software Technology*, 82, 159–172.
<https://doi.org/10.1016/j.infsof.2016.11.001>
 42. Dolado, J., & Otero, M. C. (2017). Evaluation of the comprehension of dynamic modelling in UML Evaluation of the comprehension of the dynamic modeling in UML. 5849(November).
[https://doi.org/10.1016/S0950-5849\(03\)00108-3](https://doi.org/10.1016/S0950-5849(03)00108-3)
 43. Bulegon, J., Mendling, J., Bouzeghoub, A., Heloisa, L., Palazzo, J., & Oliveira, M. De. (2017). An experiment on an ontology-based support approach for process modeling. *Information and Software Technology*, 83, 94–115.
<https://doi.org/10.1016/j.infsof.2016.11.005>
 44. Reinhartz-berger, I., Figl, K., & Haugen, Ø. (2017). US CR. *Information and Software Technology*.
<https://doi.org/10.1016/j.infsof.2017.01.012>
 45. Siqueira, F. L. (2018). Comparing the comprehensibility of requirements models: An experiment replication. *Information and Software Technology*, 96, 1–13.
<https://doi.org/10.1016/j.infsof.2017.11.002>
 46. Souza, E., Moreira, A., Abrah, S., Insfran, E., Silva, D., Abrah, S., & Insfran, E. (2018). Accepted Manuscript.
<https://doi.org/10.1016/j.infsof.2018.08.001>
 47. Campusano, M., Fabry, J., & Bergel, A. (2019). Live programming in practice: A controlled experiment on state machines for robotic behaviors. *Information and Software Technology*, 108, 99–114.
<https://doi.org/10.1016/j.infsof.2018.12.008>
 48. Abrahão, S., Insfran, E., González-Ladrón-de-Guevara, F., Fernández-Diego, M., Cano-Genoves, C., & de Oliveira, R. P. (2019). Assessing the effectiveness of goal-oriented modeling languages: A family of experiments. *Information and Software Technology*, 116, 106171.
<https://doi.org/10.1016/j.infsof.2019.08.003>
 49. Zubcoff, J., Garrigós, I., Casteleyn, S., Mazón, J.-N., Aguilar, J.-A., & Gomariz-Castillo, F. (2019). Evaluating different i*-based approaches for selecting functional

- requirements while balancing and optimizing non-functional requirements: A controlled experiment. *Information and Software Technology*, 106, 68–84.
<https://doi.org/10.1016/j.infsof.2018.09.004>
50. Cachero, C., Meliá, S., & Hermida, J. M. (2019). Impact of model notations on the productivity of domain modelling: An empirical study. *Information and Software Technology*, 108, 78–87.
<https://doi.org/10.1016/j.infsof.2018.12.005>
 51. Oliveira, R., Mello, R. De, Fernandes, E., Garcia, A., & Lucena, C. (2019). Collaborative or individual identification of code smells ? On the effectiveness of novice and professional developers. *Information and Software Technology*, 120(December), 106242.
<https://doi.org/10.1016/j.infsof.2019.106242>
 52. Ferreira, J. M., Acuña, S. T., Dieste, O., Vegas, S., Santos, A., Rodríguez, F., & Juristo, N. (2020). Impact of usability mechanisms: An experiment on efficiency, effectiveness and user satisfaction. *Information and Software Technology*, 117, 106195.
<https://doi.org/10.1016/j.infsof.2019.106195>

A.3 Papers from the JSS journal

1. Laitenberger, O., Atkinson, C., Schlich, M., & Emam, K. El. (2000). An experimental comparison of reading techniques for defect detection in UML design documents. *Journal of Systems and Software*, 53(2), 183–204.
[https://doi.org/10.1016/S0164-1212\(00\)00052-2](https://doi.org/10.1016/S0164-1212(00)00052-2)
2. Zendler, A., Pfeiffer, T., Eicks, M., & Lehner, F. (2001). Experimental comparison of coarse-grained concepts in UML, OML, and TOS. *Journal of Systems and Software*, 57(1), 21–30.
[https://doi.org/10.1016/S0164-1212\(00\)00114-X](https://doi.org/10.1016/S0164-1212(00)00114-X)
3. Prechelt, L., Unger, B., Philippsen, M., & Tichy, W. (2003). A controlled experiment on inheritance depth as a cost factor for code maintenance. *Journal of Systems and Software*, 65(2), 115–126.
[https://doi.org/10.1016/S0164-1212\(02\)00053-5](https://doi.org/10.1016/S0164-1212(02)00053-5)
4. Biffl, S. (2003). Evaluating defect estimation models with major defects. *Journal of Systems and Software*, 65(1), 13–29.
[https://doi.org/10.1016/S0164-1212\(02\)00025-0](https://doi.org/10.1016/S0164-1212(02)00025-0)
5. Deligiannis, I. (2004). A controlled experiment investigation of an object-oriented design heuristic A controlled experiment investigation of an object-oriented design heuristic for maintainability. 1212(July).
[https://doi.org/10.1016/S0164-1212\(03\)00240-1](https://doi.org/10.1016/S0164-1212(03)00240-1)
6. Mu, M. M. (2005). Two controlled experiments concerning the comparison of pair programming to peer review. 78, 166–179.
<https://doi.org/10.1016/j.jss.2004.12.019>
7. Meso, P., Madey, G., Troutt, M. D., & Liegle, J. (2006). The knowledge management efficacy of matching information systems development methodologies with application characteristics—an experimental study. *Journal of Systems and Software*, 79(1), 15–28.
<https://doi.org/10.1016/j.jss.2004.11.002>
8. Staron, M., Kuzniarz, L., & Wohlin, C. (2006). Empirical assessment of using stereotypes to improve comprehension of UML models: A set of experiments. *Journal of*

- Systems and Software, 79(5), 727–742.
<https://doi.org/10.1016/j.jss.2005.09.014>
9. Canfora, G., Cimitile, A., Garcia, F., Piattini, M., & Visaggio, C. A. (2007). Evaluating performances of pair designing in industry. *Journal of Systems and Software*, 80(8), 1317–1327.
<https://doi.org/10.1016/j.jss.2006.11.004>
 10. Kristin, A., & Thomas, R. (2007). Comprehension strategies and difficulties in maintaining object-oriented systems : An explorative study. 80, 1541–1559.
<https://doi.org/10.1016/j.jss.2006.10.041>
 11. Keil, M., Li, L., Mathiassen, L., & Zheng, G. (2008). The influence of checklists and roles on software practitioner risk perception and decision-making. *Journal of Systems and Software*, 81(6), 908–919.
<https://doi.org/10.1016/j.jss.2007.07.035>
 12. Berander, P., & Svahnberg, M. (2009). The Journal of Systems and Software Evaluating two ways of calculating priorities in requirements hierarchies – An experiment on hierarchical cumulative voting. *The Journal of Systems & Software*, 82(5), 836–850.
<https://doi.org/10.1016/j.jss.2008.11.841>
 13. Ottensooser, A., Fekete, A., Reijers, H. A., Mendling, J., & Menictas, C. (2012). The Journal of Systems and Software Making sense of business process descriptions : An experimental comparison of graphical and textual notations. *The Journal of Systems & Software*, 85(3), 596–606.
<https://doi.org/10.1016/j.jss.2011.09.023>
 14. Maia, M. D. A., & Lafetá, R. F. (2013). The Journal of Systems and Software On the impact of trace-based feature location in the performance of software maintainers. *The Journal of Systems & Software*, 86(4), 1023–1037.
<https://doi.org/10.1016/j.jss.2012.12.032>
 15. Fernandez, A., Abrahão, S., & Insfran, E. (2013). The Journal of Systems and Software Empirical validation of a usability inspection method for model-driven Web development. *The Journal of Systems & Software*, 86(1), 161–186.
<https://doi.org/10.1016/j.jss.2012.07.043>
 16. Kumar, A., & Brügge, B. (2013). The Journal of Systems and Software A mixed-method approach for the empirical evaluation of the issue-based variability modeling. *The Journal of Systems & Software*, 86(7), 1831–1849.
<https://doi.org/10.1016/j.jss.2013.01.038>
 17. El-attar, M. (2014). The Journal of Systems and Software Using SMCD to reduce inconsistencies in misuse case models : A subject-based empirical evaluation. *The Journal of Systems & Software*, 87, 104–118.
<https://doi.org/10.1016/j.jss.2013.10.017>
 18. Miguel, J., Navarro, E., Sánchez, P., & Alonso, D. (2015). The Journal of Systems and Software A controlled experiment to evaluate the understandability of KAOS and i* for modeling Teleo-Reactive systems. *The Journal of Systems & Software*, 100, 1–14.
<https://doi.org/10.1016/j.jss.2014.10.010>
 19. Sabatucci, L., Ceccato, M., Marchetto, A., & Susi, A. (2015). The Journal of Systems and Software Ahab ’ s legs in scenario-based requirements validation : An experiment to study communication mistakes. *The Journal of Systems & Software*, 109, 124–136.
<https://doi.org/10.1016/j.jss.2015.07.039>
 20. Lehtinen, T. O. A., Mäntylä, M. V., Itkonen, J., & Vanhanen, J. (2015). The Journal of Systems and Software Diagrams or structural lists in software project retrospectives –

- An experimental comparison. 103, 17–35.
<https://doi.org/10.1016/j.jss.2015.01.020>
21. Karpati, P., Opdahl, A. L., & Sindre, G. (2015). The Journal of Systems and Software Investigating security threats in architectural context : Experimental evaluations of misuse case maps. *The Journal of Systems & Software*, 104, 90–111.
<https://doi.org/10.1016/j.jss.2015.02.040>
 22. Bernárdez, B., Durán, A., Parejo, J. A., & Cortés, A. R. (2016). An experimental replication on the effect of the practice of mindfulness in conceptual modeling performance R. 0, 1–20.
<https://doi.org/10.1016/j.jss.2016.06.104>
 23. Jørgensen, M. (2016). The Journal of Systems and Software Unit effects in software project effort estimation : Work-hours gives lower effort estimates than workdays. 117, 274–281.
<https://doi.org/10.1016/j.jss.2016.03.048>
 24. Løhre, E., & Jørgensen, M. (2016). The Journal of Systems and Software Numerical anchors and their strong effects on software development effort estimates. 116, 49–56.
<https://doi.org/10.1016/j.jss.2015.03.015>
 25. Dolado, J., & Otero, M. C. (2017). An empirical comparison of the dynamic modeling in OML and UML An empirical comparison of the dynamic modeling in OML and UML. August 2005.
<https://doi.org/10.1016/j.jss.2004.11.022>
 26. Caivano, D., Fernández-ropero, M., Pérez-castillo, R., Piattini, M., & Scalera, M. (2018). The Journal of Systems & Software Artifact-based vs . human-perceived understandability and modi fi ability of refactored business processes : An experiment. 144(October 2017), 143–164.
<https://doi.org/10.1016/j.jss.2018.06.026>
 27. Avila, L. F. D., Farias, K., Luis, J., & Barbosa, V. (2020). The Journal of Systems and Software Effects of contextual information on maintenance effort : A controlled experiment. 159.
<https://doi.org/10.1016/j.jss.2019.110443>

A.4 Papers from the TSE journal

1. Maxion, R. A. (2000). Eliminating exception handling errors with dependability cases: a comparative, empirical study. *IEEE Transactions on Software Engineering*, 26(9), 888–906.
<https://doi.org/10.1109/32.877848>
2. Peleg, M., & Dori, D. (2000). The model multiplicity problem: Experimenting with real-time specification methods. *IEEE Transactions on Software Engineering*, 26(8), 742–759.
<https://doi.org/10.1109/32.879812>
3. Briand, L. C., Bunse, C., & Daly, J. W. (2001). A controlled experiment for evaluating quality guidelines on the maintainability of object-oriented designs. *IEEE Transactions on Software Engineering*, 27(6), 513–529.
<https://doi.org/10.1109/32.926174>
4. Prechelt, L., Unger, B., Tichy, W. F., Brössler, P., & Votta, L. G. (2001). A controlled experiment in maintenance comparing design patterns to simpler solutions. *IEEE Transactions on Software Engineering*, 27(12), 1134–1144.
<https://doi.org/10.1109/32.988711>

5. Prechelt, L., & Unger, B. (2001). An experiment measuring the effects of Personal Software Process (PSP) training. *IEEE Transactions on Software Engineering*, 27(5), 465–472.
<https://doi.org/10.1109/32.922716>
6. Bratthall, L., & Wohlin, C. (2002). Is it possible to decorate graphical software design and architecture models with qualitative information? - An experiment. *IEEE Transactions on Software Engineering*, 28(12), 1181–1193.
<https://doi.org/10.1109/TSE.2002.1158290>
7. Prechelt, L., Unger-Lamprecht, B., Philippsen, M., & Tichy, W. F. (2002). Two controlled experiments assessing the usefulness of design pattern documentation in program maintenance. *IEEE Transactions on Software Engineering*, 28(6), 595–606.
<https://doi.org/10.1109/TSE.2002.1010061>
8. Vitharana, P., & Ramamurthy, K. (2003). Computer-mediated group support, anonymity, and the software inspection process: An empirical investigation. *IEEE Transactions on Software Engineering*, 29(2), 167–180.
<https://doi.org/10.1109/TSE.2003.1178054>
9. Bandi, R. K., Vaishnavi, V. K., & Turk, D. E. (2003). Object-Oriented Design Complexity Metrics. *IEEE Transactions on Software Engineering*, 29(1), 77–87.
<http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=1166590>
10. Thelin, T., Runeson, P., & Wohlin, C. (2003). An experimental comparison of usage-based and checklist-based reading. *IEEE Transactions on Software Engineering*, 29(8), 687–704.
<https://doi.org/10.1109/TSE.2003.1223644>
11. Arisholm, E., & Sjøberg, D. I. K. (2004). Evaluating the effect of a delegated versus centralized control style on the maintainability of object-oriented software. *IEEE Transactions on Software Engineering*, 30(8), 521–534.
<https://doi.org/10.1109/TSE.2004.43>
12. Briand, L. C., Labiche, Y., Di Penta, M., & Yan-Bondoc, H. (2005). An experimental investigation of formality in UML-based development. *IEEE Transactions on Software Engineering*, 31(10), 833–849.
<https://doi.org/10.1109/TSE.2005.105>
13. Erdogmus, H., Morisio, M., & Torchiano, M. (2005). On the effectiveness of the test-first approach to programming. *IEEE Transactions on Software Engineering*, 31(3), 226–237.
<https://doi.org/10.1109/TSE.2005.37>
14. Arisholm, E., Briand, L. C., Hove, S. E., & Labiche, Y. (2006). The impact of UML documentation on software maintenance: An experimental evaluation. *IEEE Transactions on Software Engineering*, 32(6), 365–381.
<https://doi.org/10.1109/TSE.2006.59>
15. Arisholm, E., Gallis, H., Dybå, T., & Sjøberg, D. I. K. (2007). Evaluating pair programming with respect to system complexity and programmer expertise. *IEEE Transactions on Software Engineering*, 33(2), 65–86.
<https://doi.org/10.1109/TSE.2007.17>
16. Dzidek, W. J., Arisholm, E., & Briand, L. C. (2008). A realistic empirical evaluation of the costs and benefits of UML in software maintenance. *IEEE Transactions on Software Engineering*, 34(3), 407–432.
<https://doi.org/10.1109/TSE.2008.15>

17. Carver, J. C., Nagappan, N., & Page, A. (2008). The impact of educational background on the effectiveness of requirements inspections: An empirical study. *IEEE Transactions on Software Engineering*, 34(6), 800–812.
<https://doi.org/10.1109/TSE.2008.49>
18. Lui, K. M., Chan, K. C. C., & Nosek, J. (2008). The effect of pairs in program design tasks. *IEEE Transactions on Software Engineering*, 34(2), 197–211.
<https://doi.org/10.1109/TSE.2007.70755>
19. Ricca, F., Di Penta, M., Torchiano, M., Tonella, P., & Ceccato, M. (2010). How developers' experience and ability influence web application comprehension tasks supported by UML stereotypes: A series of four experiments. *IEEE Transactions on Software Engineering*, 36(1), 96–118.
<https://doi.org/10.1109/TSE.2009.69>
20. Perepletchikov, M., & Ryan, C. (2011). A controlled experiment for evaluating the impact of coupling on the maintainability of service-oriented software. *IEEE Transactions on Software Engineering*, 37(4), 449–465.
<https://doi.org/10.1109/TSE.2010.61>
21. Cornelissen, B., Zaidman, A., & Deursen, A. Van. (2011). A controlled experiment for program comprehension through trace visualization. *IEEE Transactions on Software Engineering*, 37(3), 341–355.
<https://doi.org/10.1109/TSE.2010.47>
22. Mouchawrab, S., Briand, L. C., Labiche, Y., & Di Penta, M. (2011). Assessing, comparing, and combining state machine-based testing and structural testing: A series of experiments. *IEEE Transactions on Software Engineering*, 37(2), 161–187.
<https://doi.org/10.1109/TSE.2010.32>
23. Prechelt, L. (2011). Plat-Forms: A Web development platform comparison by an exploratory experiment searching for emergent platform properties. *IEEE Transactions on Software Engineering*, 37(1), 95–108.
<https://doi.org/10.1109/TSE.2010.22>
24. Ramasubbu, N., Kemerer, C. F., & Hong, J. (2012). Structural complexity and programmer team strategy: An experimental test. *IEEE Transactions on Software Engineering*, 38(5), 1054–1068.
<https://doi.org/10.1109/TSE.2011.88>
25. Wilkerson, J. W., Nunamaker, J. F., & Mercer, R. (2012). Comparing the defect reduction benefits of code inspection and test-driven development. *IEEE Transactions on Software Engineering*, 38(3), 547–560.
<https://doi.org/10.1109/TSE.2011.46>
26. Abrahão, S., Gravino, C., Insfran, E., Scanniello, G., & Tortora, G. (2013). Assessing the effectiveness of sequence diagrams in the comprehension of functional requirements: Results from a family of five experiments. *IEEE Transactions on Software Engineering*, 39(3), 327–342.
<https://doi.org/10.1109/TSE.2012.27>
27. Latorre, R. (2014). Effects of developer experience on learning and applying unit test-driven development. *IEEE Transactions on Software Engineering*, 40(4), 381–395.
<https://doi.org/10.1109/TSE.2013.2295827>
28. Krein, J. L., Prechelt, L., Juristo, N., Nanthamornphong, A., Carver, J. C., Vegas, S., Knutson, C. D., Seppi, K. D., & Eggett, D. L. (2016). A Multi-Site Joint Replication of a Design Patterns Experiment Using Moderator Variables to Generalize across

- Contexts. *IEEE Transactions on Software Engineering*, 42(4), 302–321.
<https://doi.org/10.1109/TSE.2015.2488625>
29. Borstler, J., & Paech, B. (2016). The Role of Method Chains and Comments in Software Readability and Comprehension-An Experiment. *IEEE Transactions on Software Engineering*, 42(9), 886–898.
<https://doi.org/10.1109/TSE.2016.2527791>
 30. Aranda, A. M., Dieste, O., & Juristo, N. (2016). Effect of Domain Knowledge on Elicitation Effectiveness: An Internally Replicated Controlled Experiment. *IEEE Transactions on Software Engineering*, 42(5), 427–451.
<https://doi.org/10.1109/TSE.2015.2494588>
 31. Panach Navarrete, J. I., Dieste, O., Marin, B., Espana, S., Vegas, S., Pastor, O., & Juristo, N. (2018). Evaluating Model-Driven Development Claims with respect to Quality: A Family of Experiments. *IEEE Transactions on Software Engineering*, c, 1–18.
<https://doi.org/10.1109/TSE.2018.2884706>
 32. Karac, E. I., Turhan, B., & Juristo, N. (2019). A Controlled Experiment with Novice Developers on the Impact of Task Description Granularity on Software Quality in Test-Driven Development. *IEEE Transactions on Software Engineering*, 14(8), 1–1.
<https://doi.org/10.1109/tse.2019.2920377>
 33. Tosun, A., Dieste, O., Vegas, S., Pfahl, D., Rungi, K., & Juristo, N. (2019). Investigating the Impact of Development Task on External Quality in Test-Driven Development: An Industry Experiment. *IEEE Transactions on Software Engineering*, PP(X), 1–1.
<https://doi.org/10.1109/tse.2019.2949811>
 34. Czepa, C., & Zdun, U. (2020). On the Understandability of Temporal Properties Formalized in Linear Temporal Logic, Property Specification Patterns and Event Processing Language. *IEEE Transactions on Software Engineering*, 46(1), 100–112.
<https://doi.org/10.1109/TSE.2018.2859926>
 35. Fucci, D., Scanniello, G., Romano, S., & Juristo, N. (2020). Need for Sleep: The Impact of a Night of Sleep Deprivation on Novice Developers' Performance. *IEEE Transactions on Software Engineering*, 46(1), 1–19.
<https://doi.org/10.1109/TSE.2018.2834900>

Funding Open access funding provided by Vienna University of Economics and Business (WU).

Open Access This article is licensed under a Creative Commons Attribution 4.0 International License, which permits use, sharing, adaptation, distribution and reproduction in any medium or format, as long as you give appropriate credit to the original author(s) and the source, provide a link to the Creative Commons licence, and indicate if changes were made. The images or other third party material in this article are included in the article's Creative Commons licence, unless indicated otherwise in a credit line to the material. If material is not included in the article's Creative Commons licence and your intended use is not permitted by statutory regulation or exceeds the permitted use, you will need to obtain permission directly from the copyright holder. To view a copy of this licence, visit <http://creativecommons.org/licenses/by/4.0/>.

References

- van der Aalst W (2016) *Process mining - data science in action*, 2nd edn. Springer, Berlin
- van der Aalst W, Adriansyah A, van Dongen B (2012) Replaying history on process models for conformance checking and performance analysis. *WIREs Data Min Knowl Discov* 2(2):182–192.
<https://doi.org/10.1002/widm.1045>
- Abbott A (1995) Sequence analysis: New methods for old ideas. *Ann Rev Sociol* 21(1):93–113

- Abbott A, Tsay A (2000) Sequence analysis and optimal matching methods in sociology: Review and prospect. *Sociol Methods Res* 29(1):3–33. <https://doi.org/10.1177/0049124100029001001>
- Abran A, Moore JW, Bourque P, Dupuis R, Tripp LL (2004) Software engineering body of knowledge. IEEE Computer Society, Angela Burgess
- Baldassarre MT, Carver J, Dieste O, Juristo N (2014) Replication types: Towards a shared taxonomy. In: Proceedings of the 18th international conference on evaluation and assessment in software engineering, association for computing machinery, New York, NY, USA, EASE '14. <https://doi.org/10.1145/2601248.2601299>
- Basili VR (1993) The experimental paradigm in software engineering. In: Experimental software engineering issues: critical assessment and future directions. Springer, pp 1–12
- Basili VR (1996) The role of experimentation in software engineering: past, current, and future. In: Proceedings of IEEE 18th international conference on software engineering. IEEE, pp 442–449
- Berente N, Seidel S, Safadi H (2019) Research commentary—data-driven computationally intensive theory development. *Inf Syst Res* 30(1):50–64
- Bettis RA, Helfat CE, Shaver JM (2016) The necessity, logic, and forms of replication. *Strateg Manag J* 37(11):2193–2203. <https://doi.org/10.1002/smj.2580>
- Bourque P, Fairley RE, Abran A, Garbajosa J, Keeni G, Shen B, April A (2014) Guide to the software engineering body of knowledge
- Brinkkemper S (1996) Method engineering: engineering of information systems development methods and tools. *Inf Softw Technol* 38(4):275–280
- Brown SA, Dennis AR, Samuel BM, Tan B, Valacich JS, Whitley EA (2016) Replication research: Opportunities, experiences and challenges. In: 2016 International conference on information systems, ICIS 2016. Association for Information Systems
- Burnes B (1996) No such thing as... a "one best way" to manage organizational change. *Manag Decis* 34(10):11–18
- Burton-Jones A, Wand Y, Weber R (2009) Guidelines for empirical evaluations of conceptual modeling grammars. *J Assoc Inf Syst* 10(6):1
- Carver JC (2010) Towards reporting guidelines for experimental replications: A proposal. In: Proceedings of the 1st international workshop on replication in empirical software engineering research
- Cohen J (1960) A coefficient of agreement for nominal scales. *Educ Psychol Meas* 20(1):37–46. <https://doi.org/10.1177/001316446002000104>
- Cruz M, Bernárdez B, Durán A, Galindo JA, Ruiz-Cortés A (2020) Replication of studies in empirical software engineering: A systematic mapping study, from 2013 to 2018. *IEEE Access* 8:26773–26791
- De Koninck P, De Weerd P, vanden Broucke SK (2017) Explaining clusterings of process instances. *Data Min Knowl Discov* 31(3):774–808
- DiMaggio PJ, Powell WW (1983) The iron cage revisited: Institutional isomorphism and collective rationality in organizational fields. *Am Sociol Rev* :147–160
- Dybå T, Kampenes VB, Sjøberg DI (2006) A systematic review of statistical power in software engineering experiments. *Inf Softw Technol* 48(8):745–755
- Gabadinho A, Ritschard G, Müller N, Studer M (2011) Analyzing and visualizing state sequences in r with traminer. *J Stat Softw* 40(4):1–37. <https://doi.org/10.18637/jss.v040.i04>
- Gauch HG (2003) Scientific method in practice. Cambridge University Press, Cambridge
- Gersick CJ, Hackman JR (1990) Habitual routines in task-performing groups. *Organ Behav Hum Decis Process* 47(1):65–97
- Hannay JE, Sjøberg DI (2007) A systematic review of theory use in software engineering experiments. *IEEE Trans Softw Eng* 33(2):87–107
- Holsti O (1969) Content analysis for the social sciences and humanities. Addison-Wesley Publishing Company, Boston. <https://books.google.at/books?id=IMpCAAAAIAAJ>
- Jedlitschka A, Ciolkowski M (2004) Towards evidence in software engineering. In: Proceedings. 2004 International symposium on empirical software engineering, 2004. ISESE'04. IEEE, pp 261–270
- Jedlitschka A, Pfahl D (2005) Reporting guidelines for controlled experiments in software engineering. In: [Host publication title missing], IEEE - Institute of Electrical and Electronics Engineers Inc., pp 95–104. <https://doi.org/10.1109/ISESE.2005.1541818>
- Jedlitschka A, Ciolkowski M, Pfahl D (2008) Reporting experiments in software engineering. Springer London, London, pp 201–228
- Jedlitschka A, Juristo N, Rombach D (2014) Reporting experiments to satisfy professionals' information needs. *Empir Softw Eng* 19(6):1921–1955
- Juristo N, Moreno A (2001) Basics of software engineering experimentation. Springer Nature Book Archives Millennium. Springer US, New York. <https://books.google.at/books?id=ovWfOeW653EC>

- Juristo N, Moreno AM (2010) Basics of Software Engineering Experimentation 1st edn. Springer Publishing Company, Incorporated
- Kampenes VB, Dybå T, Hannay JE, Sjøberg DI (2007) A systematic review of effect size in software engineering experiments. *Inf Softw Technol* 49(11-12):1073–1086
- Kaufman L, Rousseeuw PJ (1990) Finding groups in data: an introduction to cluster analysis. John Wiley. <https://doi.org/10.1002/9780470316801>
- Kelly JR, McGrath JE (1985) Effects of time limits and task types on task performance and interaction of four-person groups. *J Pers Soc Psychol* 49(2):395
- Kitchenham B, Charters S (2007) Guidelines for performing systematic literature reviews in software engineering. Tech. rep.
- Kitchenham BA, Pfleeger SL, Pickard LM, Jones PW, Hoaglin DC, Emam KE, Rosenberg J (2002) Preliminary guidelines for empirical research in software engineering. *IEEE Trans Softw Eng* 28(8):721–734. <https://doi.org/10.1109/TSE.2002.1027796>
- Kitchenham BA, Dyba T, Jorgensen M (2004) Evidence-based software engineering. In: Proceedings. 26th International conference on software engineering. IEEE, pp 273–281
- Kwiatkowski D, Phillips PC, Schmidt P, Shin Y (1992) Testing the null hypothesis of stationarity against the alternative of a unit root: How sure are we that economic time series have a unit root? *J Econom* 54(1):159–178. [https://doi.org/10.1016/0304-4076\(92\)90104-Y](https://doi.org/10.1016/0304-4076(92)90104-Y). <https://www.sciencedirect.com/science/article/pii/030440769290104Y>
- Levitt B, Nass C (1989) The lid on the garbage can: Institutional constraints on decision making in the technical core of college-text publishers. *Adm. Sci. Q.* :190–207
- Malinova M, Gross S, Mendling J (2019) Researching information systems methods using method mining - A case study on process improvement methods. In: Proceedings of the 40th international conference on information systems, ICIS 2019, Munich, Germany, December 15-18, 2019. https://aisel.aisnet.org/icis2019/general_topics/general_topics/12
- Neuendorf K (2002) The content analysis guidebook. SAGE Publications, Thousand Oaks. <https://books.google.at/books?id=huPVtmu4sigC>
- Ralph P (2018) Toward methodological guidelines for process theories and taxonomies in software engineering. *IEEE Trans Softw Eng* 45(7):712–735
- Ralph P, bin Ali N, Baltes S, Bianculli D, Diaz J, Dittrich Y, Ernst N, Felderer M, Feldt R, Filieri A, de França BBN, Furia CA, Gay G, Gold N, Graziotin D, He P, Hoda R, Juristo N, Kitchenham B, Lenarduzzi V, Martínez J, Melegati J, Mendez D, Menzies T, Moller J, Pfahl D, Robbes R, Russo D, Saarimäki N, Sarro F, Taibi D, Siegmund J, Spinellis D, Staron M, Stol K, Storey MA, Taibi D, Tamburri D, Torchiano M, Treude C, Turhan B, Wang X, Vegas S (2021) Empirical standards for software engineering research. *arXiv:2010.03525*
- Recker J (2012) Scientific research in information systems: a beginner's guide. Springer Publishing Company Incorporated, Berlin
- Recker J, Indulska M, Green P, Burton-Jones A, Weber R (2019) Information systems as representations: A review of the theory and evidence. *J Assoc Inf Syst* 20(6). Article number: 5 735–786. <https://doi.org/10.17705/1jais.00550>. <https://eprints.qut.edu.au/123305/>
- Ruxton GD (2006) The unequal variance t-test is an underused alternative to Student's t-test and the Mann-Whitney U test. *Behav Ecol* 17(4):688–690. <https://doi.org/10.1093/beheco/ark016>, <https://academic.oup.com/beheco/article-pdf/17/4/688/17275561/ark016.pdf>
- Saldaña J (2015) The coding manual for qualitative researchers. Sage, Thousand Oaks
- Shaw M (2003) Writing good software engineering research papers: Minitutorial. In: Proceedings of the 25th international conference on software engineering, ICSE '03. IEEE Computer Society, Washington, pp 726–736. <http://dl.acm.org/citation.cfm?id=776816.776925>
- Shull F, Mendonça MG, Basili V, Carver J, Maldonado JC, Fabbri S, Travassos GH, Ferreira MC (2004) Knowledge-sharing issues in experimental software engineering. *Empir Softw Eng* 9(1-2):111–137
- Singer J (1999) Using the apa style guidelines to report experimental results. In: Proceedings of workshop on empirical studies in software maintenance, pp 71–75
- Sjøberg DI, Hannay JE, Hansen O, Kampenes VB, Karahasanovic A, Liborg NK, Rekdal AC (2005) A survey of controlled experiments in software engineering. *IEEE Trans Softw Eng* 31(9):733–753
- Sollaci LB, Pereira MG (2004) The introduction, methods, results, and discussion (imrad) structure: a fifty-year survey. *J Med Libr Assoc* 92(3):364
- Song M, Günther CW, Van der Aalst WM (2008) Trace clustering in process mining. In: International conference on business process management. Springer, pp 109–120
- Trochim WM, Donnelly JP (2001) Research methods knowledge base, vol 2. Atomic Dog Pub, Cincinnati
- Venkatesh V, Brown SA, Bala H (2013) Bridging the qualitative-quantitative divide: Guidelines for conducting mixed methods research in information systems. *MIS Q* :21–54

- Wohlin C, Runeson P, Höst M, Ohlsson MC, Regnell B, Wesslén A (2000) Experimentation in software engineering: an introduction. Kluwer Academic Publishers, Norwell
- Wohlin C, Runeson P, Höst M, Ohlsson MC, Regnell B, Wesslén A (2012) Experimentation in software engineering. Springer Publishing Company Incorporated, Berlin
- Wohlin C, Šmite D, Moe NB (2015) A general theory of software engineering: Balancing human, social and organizational capitals. *J Syst Softw* 109:229–242

Publisher's note Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.

Kate Revoredo is an Assistant Professor at the Institute for Information Business at Wirtschaftsuniversität Wien (WU Vienna), Austria. Her main research interest is on data-centric approaches for information systems. This includes topics in the area of business process management, data science, semantic web and knowledge representation. She has published more than 80 research papers in various outlets, among others in *Computers in Industry*, *Information Systems* and *Machine Learning*. She is a member of the Brazilian Commission in Artificial Intelligence and a reviewer for several journals and conferences, among others *Business & Information Systems Engineering*, *The Knowledge Engineering Review*, *International Joint Conference on Artificial Intelligence (IJCAI)* and *Conference on Artificial Intelligence (AAAI)*.

Djordje Djurica is a Teaching and Research Associate at Vienna University of Economics and Business (WU Wien). He received his bachelor's degree in Economics from the University of Novi Sad and a master's degree in Information Systems from WU Wien. Djordje's research is focused on conceptual modeling. In particular, he is interested in human interaction with conceptual models (i.e., business process and decision models) and how such models can be improved by using cognitive effective techniques.

Jan Mendling is the Einstein-Professor of Process Science with the Department of Computer Science at Humboldt-Universität zu Berlin, Germany. His research interests include various topics in the area of business process management and information systems. He has published more than 450 research papers and articles, among others in *MIS Quarterly*, *ACM Transactions on Software Engineering and Methodology*, *IEEE Transactions on Software Engineering*, *Journal of the Association of Information Systems* and *Decision Support Systems*. He is a department editor for *Business and Information Systems Engineering*, member of the board of the Austrian Society for Process Management, one of the founders of the Berlin BPM Community of Practice, and a member of the IEEE Task Force on Process Mining. He is co-author of the textbooks *Fundamentals of Business Process Management*, Second Edition, and *Wirtschaftsinformatik*, 12th Edition.