A Systematic Literature Review of Software Visualization Evaluation

L. Merino\textsuperscript{a}, M. Ghafari\textsuperscript{a}, C. Anslow\textsuperscript{b}, O. Nierstrasz\textsuperscript{a}

\textsuperscript{a}Software Composition Group, University of Bern, Switzerland
\textsuperscript{b}School of Engineering and Computer Science, Victoria University of Wellington, New Zealand

Abstract

Context: Software visualizations can help developers to analyze multiple aspects of complex software systems, but their effectiveness is often uncertain due to the lack of evaluation guidelines.

Objective: We identify common problems in the evaluation of software visualizations with the goal of formulating guidelines to improve future evaluations.

Method: We review the complete literature body of 387 full papers published in the SOFTVIS/VISSOFT conferences, and study 181 of those from which we could extract evaluation strategies, data collection methods, and other aspects of the evaluation.

Results: Of the proposed software visualization approaches, 62\% lack a strong evaluation. We argue that an effective software visualization should not only boost time and correctness but also recollection, usability, engagement, and other emotions.

Conclusion: We call on researchers proposing new software visualizations to provide evidence of their effectiveness by conducting thorough (i) case studies for approaches that must be studied \textit{in situ}, and when variables can be controlled, (ii) experiments with randomly selected participants of the target audience and real-world open source software systems to promote reproducibility and replicability. We present guidelines to increase the evidence of the effectiveness of software visualization approaches, thus improving their adoption rate.

Published in: Journal of Systems and Software, https://doi.org/10.1016/j.jss.2018.06.027

Keywords: software visualisation, evaluation, literature review

1. Introduction

Software visualizations are useful for analyzing multiple aspects of complex software systems. Software visualization tools have been proposed to help analysts make sense of multivariate data [25], to support programmers in comprehending the architecture of systems [31], to help researchers analyze version control repositories [9], and to aid developers of software product lines [16]. However, most developers are still unaware of which existing visualization approaches are suitable to adopt for their needs. We conjecture that the low adoption of software visualization results from their unproved effectiveness and lack of evaluations. Indeed, researchers adopt varying strategies to evaluate software visualization approaches, and therefore the quality of the evidence of their effectiveness varies. We believe that a characterization of the evaluation of software visualization approaches will (i) assist researchers in the field to improve the quality of evaluations, and (ii) increase the adoption of visualization among developers.

We consider previous research to be an important step to characterizing the evidence of the effectiveness of software visualization approaches. However, we reflect that previous research has failed to define what is an effective software visualization, and consequently comparing the effectiveness of visualization approaches is not possible. Moreover, we believe that some studies have used a loose definition of “case studies” and include many usage scenarios of visualization instead that present little evidence of the effectiveness of an approach. In our investigation we perform a subtler analysis of the characteristics of evaluations to elucidate these concerns. Consequently, we formulated the following research questions:

\textbf{RQ1.} What are the characteristics of evaluations that validate the effectiveness of software visualization approaches?

\textbf{RQ2.} How appropriate are the evaluations that are conducted to validate the effectiveness of software visualization?

We believe that answering these questions will assist researchers in the software visualization field to improve the quality of evaluations by identifying evaluation strategies and methods and their common pitfalls. In particular, we reviewed 181 full papers of the 387 papers published in SOFTVIS/VISSOFT. We identified evaluation strategies such as surveys, case studies, and experiments, as well as characteristics such as tasks, participants, and systems used in evaluations. We found that 62\% (i.e., 113) of the proposed software visualization approaches either do not include any evaluation, or include a weak evaluation (i.e., anecdotal evidence, usage scenarios). Almost all of them (i.e., 110) introduce a new software visualization approach. The remaining three discuss an existing approach but without providing a stronger evaluation. We also found that 29\% of the studies (i.e., 53) conducted experiments in which 30\% (i.e., 16) corresponded to visualizations that target the novice developer...
They observed that even though the proportion of publications only a few researchers conducted experiments (i.e., visualizations are evaluated via case studies published between 2000 and 2012. They found that most visualizations are evaluated via case studies and include an appropriate participant. The remaining 70% proposed visualizations for developers with various levels of experience. However, amongst them only 30% included experienced developers, and the remaining 70% (i.e., 37) included in experiments only students and academics of a convenience sample who are vulnerable to selection bias and hence hinder generalization. We found that 7% (i.e., 12) of the studies conducted a case study that involved (i) professional developers from industry, and (ii) real-world software systems. Finally, 3% (i.e., 4) of studies conducted a survey. Even though we are not aware of a similar quantitative report of the state of the art in information visualization, a review of the practice of evaluation [12] found similar issues.

We believe that for software visualization approaches to be adopted by developers, visualizations not only must prove their effectiveness via evaluations, but evaluations should also include participants of the target audience, and be based on real-world software systems. Finally, we recommend researchers in the field to conduct surveys that can help them to identify what are the frequent and complex problems that affect developers.

This paper makes the following contributions:

1. A study of the characteristics of evaluations performed in the literature of software visualization.
2. Guidelines for researchers in the visualization field who need to evaluate software visualization approaches.
3. A publicly available data set including the information of the studies and classifications.1

The remainder of the paper is structured as follows: Section 2 presents related work. Section 3 describes the main concepts that are addressed in the characterization. Section 4 describes the methodology that we followed to collect and select relevant studies proposed in the software visualization field. Section 5 presents our results by classifying evaluations based on adopted strategies, methods, and their characteristics. Section 6 discusses our research questions and threats to validity of our findings, and Section 7 concludes and presents future work.

2. Related Work

A few studies have attempted to characterize the evaluation of software visualization approaches via a literature review. For instance, Schots and Werner [35] reviewed 36 papers published between 1993 and 2012 and proposed an extended taxonomy that includes evidence of the applicability of a software visualization as a dimension [34]. They found that papers lacked a clear description of information related to the evidence on the use of visualization. Seriai et al. [38] analyzed 87 papers published between 2000 and 2012. They found that most visualizations are evaluated via case studies (i.e., 78.16%), and only a few researchers conducted experiments (i.e., 16.09%).

They observed that even though the proportion of publications that include an evaluation is fairly constant over time, they lack rigor. Mattila et al. [19] included 83 papers published between 2010 and 2015 in their analysis. They also found that only a few researchers conducted experiments (i.e., 13.25%), some performed case studies (i.e., 22.89%), and the rest used other evaluation methods. In our investigation we cover a much larger body of literature (i.e., 181 full papers) that spans up to 2017. We not only characterize the state-of-the-art in software visualization evaluation, but we also propose guidance to researchers in the field by detecting common pitfalls, and by elaborating on guidelines to conduct evaluation of software visualization approaches.

Other studies have opted to evaluate software visualization tools and have reported guidelines. For example, Storey et al. [41] evaluated 12 software visualization tools, and proposed an evaluation framework based on intent, information, presentation, interaction, and effectiveness. Sensalire et al. [36, 37] evaluated 20 software visualization tools proposed for maintenance based via experiments, and elaborated various lessons learned. They identified a number of dimensions that are critical for organizing an evaluation, and then analyzing the results. Müller et al. [27] proposed a structured approach for conducting controlled experiments in envisioned 3D software visualization tools. Instead of concentrating on rather limited number of tools, we chose a meta analysis by analyzing the reports of the evaluation of proposed visualization tools. In this way we could analyze the state-of-the-art in the practice of software visualization evaluation, and consequently elaborate guidelines for defining what is an effective software visualization.

A few reviews of the software visualization literature that focus on various domains have tangentially analyzed the evaluation aspect. Lopez-Herrejon et al. [16] analyzed evaluation strategies used in visualizations proposed for software product line engineering, and they found that most approaches used case studies. They also found that only a few performed experiments, and a few others did not explicitly describe an evaluation. Shahin et al. [39] discussed the evaluation of visualization approaches proposed to support software architecture, and classified the evidence of the evaluation using a 5-step scale [1]. The analysis of the results showed that almost half of the evaluations represent toy examples or demonstrations. The other half correspond to industrial case studies, and a very few others described experiments and anecdotal evidence of tool adoption. Novais et al. [30] investigated the evaluations of approaches that proposed visualization to analyze software evolution. In most of the analyzed studies evaluation consisted in usage examples that were demonstrated by the authors of the study. In a few of them, the demonstration was carried out by external users. Evaluation strategies based on experiments were found to be extremely rare. In almost 20% of the studies they did not find an explicit evaluation. Since the main focus of these mentioned studies is not on evaluation (as opposed to ours), they only characterize the evaluation of the analyzed studies, and offer little advice for researchers who need to perform their own evaluations of software visualizations.

Similar efforts have been made in the information visualization field. Amar and Stasko [2] proposed a task-based frame-
work for the evaluation of information visualizations. Forsell [8] proposed a guide to scientific evaluation of information visualization that focuses on quantitative experimental research. The guide contains recommendations for (a) designing, (b) conducting, (c) analyzing results, and (d) reporting on experiments. Lam et al. [15] proposed seven scenarios for empirical studies in information visualization. Isenberg et al. [12] reviewed 581 papers to analyze the practice of evaluating visualization. Some of the pitfalls they found are that in some evaluations (i) participants do not belong to the target audience, (ii) goals are not explicit, (iii) the strategy and analysis method is not appropriate, and (iv) the level of rigor is low. Elmqvist and Yi [6] proposed patterns for visualization evaluation that present solutions to common problems encountered when evaluating a visualization system. We observed that advice given in the context of information visualization can also be applied to software visualization evaluation; however, we also observed that there are particularities in software visualization that require a tailored analysis, which is an objective of our investigation.

3. Background

The strategies that researchers adopt to evaluate the effectiveness of a software visualization approach can be classified into two main categories:

i) Theoretical principles from information visualization that provide researchers support to justify a chosen visual encoding [28]. For instance, the effectiveness of perceptual channels depends on the data type (i.e., categorical, ordered, or quantitative) [17].

ii) Empirical evidence gathered from the evaluation of a technique, method or tool. Amongst them we find a) exploratory evaluations that involve high-level real-world tasks, for which identifying the aspects of the tool that boosted the effectiveness is complex; and b) explanatory evaluations in which high-level tasks are dissected into low-level (but less realistic) tasks that can be measured in isolation to identify the cause of an increase in the effectiveness of an approach [44].

Amongst the strategies used in empirical evaluations we find (a) surveys [45] that allow researchers to collect data from developers who are the users of a system, and hence analyze the collected data to generalize conclusions; (b) experiments [40] that provide researchers with a high level of control to manipulate some variables while controlling others (i.e., controlled experiments) with randomly assigned subjects (when it is not possible to ensure randomness the strategy is called “quasi-experiment”); and (c) case studies [33] that help researchers to investigate a phenomenon in its real-life context (i.e., the case), hence giving researchers a lower level of control than an experiment but enabling a deeper analysis.

Several methods exist for collecting data in each evaluation strategy. The two most common methods [7] are (i) questionnaires in which the researcher provides instructions to participants to answer a set of questions that can range from loosely structured (e.g., exploratory survey) to closed and fully structured (e.g., to collect data of the background of participants in an experiment), and (ii) interviews in which a researcher can ask a group of subjects a set of closed questions in a fixed order (i.e., fully structured), a mix of open and closed questions (i.e., semi-structured), and open-ended questions (i.e., unstructured). Less frequent methods for collecting data are observational ones such as (iii) think-aloud in which researchers ask participants to verbalize their thoughts while performing the evaluation. Besides, recent experiments have collected data using (iv) video recording to capture the behavior of participants during the evaluation; (v) sketch drawing to evaluate recollection; and (vi) eye tracking to measure the browsing behavior of eye’s movement.

Finally, there are several statistical tests that are usually used to analyze quantitative data collected from an experiment. For discrete or categorical data, tests such as Chi-square and Cohen’s kappa are suitable. For questions that analyze the relationships of independent variables, regression analysis can be applied. For correlation analysis of dependent variables one has to first analyze if the parametric assumptions holds. That is, if the data is (i) collected from independent and unbiased samples, (ii) normally distributed (Shapiro-Wilk test is suggested and proven more powerful than Kolmogorov-Smirnov [32]), and (iii) present equal variances (e.g., Levene’s test, Mauchly’s test). Parametric data can be analyzed with Pearson’s r, while non-parametric with Spearman’s Rank Correlation. For the analysis of differences of parametric data collected from two groups Student’s unpaired t-test, Paired t-test, and Hotelling’s T-square are appropriate. For the non-parametric case Mann-Whitney U and Wilcoxon Rank sum test are suitable. In the case of analysis that involves more than two groups of parametric data ANOVA is a frequent choice, which is usually followed by a post-hoc test such as Tukey HSD. When data is non-parametric Kruskal-Wallis test and Friedman test are suitable as well.

4. Methodology

We applied the Systematic Literature Review approach, a rigorous and auditable research methodology for Evidence-Based Software Engineering. We followed Keele’s comprehensive guidelines [14], which make it less likely that the results of the literature survey will be biased. The method offers a means for evaluating and interpreting relevant research to a topic of interest by evidence, which is robust and transferable. We defined a review protocol to ensure rigor and reproducibility, in which we determine (i) research questions, (ii) data sources and search strategy, (iii) inclusion and exclusion criteria, (iv) quality assessment, (v) data extraction, and (vi) selected studies.

4.1. Data sources and search strategy

Systematic literature reviews often define as their data source digital libraries such as ACM DL\(^2\) or IEEE Xplore.\(^3\) To find

\(^{2}\)http://dl.acm.org/

\(^{3}\)http://ieeexplore.ieee.org
suitable primary studies for analysis, they define a search strategy that typically is based on keywords. Instead, we decided to adopt as data source the complete set of papers published by the SOFTVIS and VISSOFT conferences. We believe the sixteen editions and hundreds of papers dedicated especially to software visualization offer a sound body of literature used in previous studies [26]. We based our decision on (i) the good \( B \) classification that they obtain in the CORE ranking\(^4\) (which considers citation rates, paper submission and acceptance rates among other indicators), (ii) related work that concluded that results from the analysis of software visualization evaluation in papers published by other venues do not differ from those published by SOFTVIS/VISSOFT [19, 38]. Although we observe that publications in better ranked venues might require stronger evaluations, we believe that analyzing a collection of studies that have been accepted for publication according to fairly similar criteria will support a more objective comparison, and will provide a suitable baseline for future investigations.

4.2. Inclusion and exclusion criteria

We reviewed the proceedings and programs of the venues to include full papers and exclude other types of papers that due to limited space are unlikely to contain enough detail. In particular, from the 387 papers we excluded 178 papers that corresponded to: (i) 61 poster, (ii) 52 new ideas and emerging results (NIER), (iii) 44 tool demo (TD), (iv) 8 keynote, (v) 8 position, and (vi) 5 challenge papers.

4.3. Quality assessment

We then assessed the quality of the remaining 209 papers. We classified the studies according to the categories proposed by Munzner [28], in which a visualization paper can be classified into one of five categories:

- **a) Evaluations** describe how a visualization is used to deal with tasks in a problem domain. Evaluations are often conducted via user studies in laboratory settings in which participants solve a set of tasks while variables are measured.
- **b) Design studies** show how existing visualization techniques can be usefully combined to deal with a particular problem domain. Typically, design studies are evaluated through case studies and usage scenarios.
- **c) Systems** elaborate on the architectural design choices of a proposed visualization tool and the lessons learned from observing its use.
- **d) Techniques** focus on novel algorithms that improve the effectiveness of visualization. Techniques are often evaluated using benchmarks that measure performance.
- **e) Models** include Commentary papers in which an expert in the field advocate a position and argue to support it; Formalism papers present new models, definitions or terminology to describe techniques; and Taxonomy papers propose categories that help researchers to analyze the structure of a domain.

For each paper, we first read the abstract, second the conclusion, and finally, in the cases where we still were not sure of their main contribution, we read the rest of the paper. Although some papers might exhibit characteristics of more than one type, we classified them by focusing on their primary contribution.

We observed that model papers in which the main contribution is a commentary, a formalism or a taxonomy, usually do not describe explicit evaluations. Consequently, we excluded twenty-eight papers that we classified in those categories: (i) six commentary, (ii) seven taxonomy, and (iii) fifteen formalism papers.

Figure 1a provides an overview of the selection process. Figure 1b summarizes the 387 collected papers and highlights the 181 included in the study. Figure 1c shows the outcome of our classification. We observe that the two venues have a slightly different focus. SOFTVIS papers focus mostly on design studies, while VISSOFT papers focus mainly on techniques. A frequent critique of visualization papers is a lack of evaluation. Indeed, papers in which the main contribution is an evaluation are unusual (i.e., 10%). The chart also shows that the two main paper types in visualization are design study and technique.

The collection of 181 full papers includes studies from six to eleven pages in length. Initially, we were reluctant to include six-page papers, but we observed that in two editions of the conferences all full papers were of that length. Consequently, we analyzed the distribution of research strategies used to evaluate software visualization approaches by paper length. We did not find any particular trend, and so decided to include them.

4.4. Data extraction

To accelerate the process of finding and extracting the data from the studies, we collected keywords that authors commonly use to describe evaluations iteratively. That is, we started the process by searching for the following keywords in each paper: “evaluation”, “survey”, “experiment”, “case study”, and “user study”. When we did not find these keywords, we manually inspected the paper and looked for other new representative keywords to expand our set. During the manual inspection when we did not find an explicit evaluation we labeled the papers accordingly. In the end, we collected the following set of keywords:

\[
\text{evaluation}, \text{survey}, \text{case}\{\text{user}\}\text{study}\{\text{ies}\}, \text{application}\ | \text{usage}\ | \text{analysis}\ | \text{example}\{\text{s}\}, \text{use}\text{case}\{\text{s}\}, \text{application}\ \text{scenario}\{\text{s}\}, \text{controlled}\ | \text{user}\text{experiment}, \text{demonstration}, \text{user}\text{scenario}\{\text{s}\}, \text{example}\ | \text{of}\ | \text{use}, \text{usage}\text{scenario}\{\text{s}\}, \text{example}\ \text{scenario}\{\text{s}\}, \text{demonstrative}\ \text{result}\{\text{s}\}}
\]

We investigated whether evaluations that involve users are conducted with end users from the expected target audience (i.e., representative sample) to ensure the generality of results. Therefore, in studies that used this type of evaluation, we extracted who conducted the evaluation, and what subject systems were involved. We extracted these data by scanning the evaluation section of papers. In particular, we extracted (i) data...
Inclusion Criteria
N = 387
Keynote [N=8]
Challenge [N=5]
NIER [N=52]
TD [N=44]
Poster [N=6]  
Commentary [N=6]
Taxonomy [N=7]
Formalism [N=15]
Quality Assessment
N = 181

(a) Stages of the search process and number of selected studies in each stage.

(b) The 181 included papers from the collection of 387 papers published in SOFTVIS/VISSOFT venues.

(c) Classification of the 181 SOFTVIS/VISSOFT full papers by type.

Figure 1: The 181 SOFTVIS/VISSOFT full papers included.

collection methods (e.g., think-aloud, interview, questionnaire); (ii) number of participants and their background, (iii) tasks, (iv) subject system, (v) dependent variables, and (vi) statistical tests.

4.5. Selected studies

We included in our study the 181 papers listed in Tables 1 and 2. The papers are identified by venue and evaluation strategy.

5. Results

We report the characteristics of the extracted data and the categories used to classify them for quantitative analysis. Figure 2 shows the distribution of the studies categorized by paper type [28] and research strategy used to evaluate visualizations. Table 3 presents our classification of the evaluation strategy adopted by papers into one of three main categories: (i) theoretical, (ii) no explicit evaluation, and (iii) empirical. For evaluations that used an empirical strategy, we classified them into one of five categories: (i) anecdotal evidence, (ii) usage scenarios, (iii) survey, (iv) case study, and (v) experiment.

We report on characteristics of experiments such as data collection methods, type of analysis, visual tasks, dependent variables, statistical tests, and scope. The complete classification of the 181 included studies is displayed in Tables 4, 5, 6, 7, 8, and 9.

5.1. Data Collection Methods

In Table 4 we list the various methods that researchers used to collect data from experiments. The most frequent were questionnaires, which are normally used to collect data of the background of participants at the beginning of experiments and final observations at the end. Questionnaires are found across all types of evaluation strategies (i.e., survey, experiment, case study). Interviews are fairly frequent and found mostly in case studies. We also found traditional observational methods (e.g., think-aloud), but also fairly new methods (e.g., eye tracking).
as future work to conduct an experimental evaluation. The remaining 155 studies (i.e., 86%) adopted an empirical strategy to evaluate software visualization approaches. Amongst them, we found that multiple strategies were used. We investigated the evidence of the effectiveness of visualization approaches provided by those strategies.

Figure 3 shows the relation between the data collection methods used in evaluation strategies. We observe that most case studies do not describe the methods used to collect data; however, we presume they are observational ones, such as one [S90] that reported to have conducted interviews. The few surveys in the analysis collected data using interviews and questionnaires. One survey [S113] did not describe the method to collect data. Experiments use multiple methods to collect data. They mainly use questionnaires, interviews, and the think-aloud protocol. Recent experiments have used video recording, and other methods such as sketch drawing, eye tracking, log analysis, and emotion cards.

### 5.2.1. Anecdotal Evidence

We found six studies (i.e., 3%) that support the claim of effectiveness of visualizations on anecdotal evidence of tool adoption. Two papers [S55, S73] proposed a visualization to support the student audience and reported that tools were successfully used in software engineering courses. The remaining four studies [S70, S91, S97, S103] that focused on the developer audience reported that visualizations were used intensively and obtained positive feedback.

### 5.2.2. Usage Scenarios

Eighty-three studies (i.e., 46%) evaluated software visualizations via usage scenarios. In this type of evaluation, authors posed envisioned scenarios and elaborated on how the visualization was expected to be used. Usually, they selected an open-source software system as the subject of the visualization. The most popular systems that we found were written in (i) Java, such as ArgoUML (4×), Ant (4×), JHotDraw (3×), Java SDK (2×), and Weka (2×); (ii) C++, such as Mozilla (7×), VTK (2×), and GNOME (2×); and, (iii) Smalltalk Pharo (4×). We found that several names were used among the studies to describe this strategy. We observed that sixty-seven studies (i.e., 37%) labeled evaluations as case studies, while twenty-six (i.e., 14%) presented them as use cases. In the rest of the cases, authors used titles such as: “application examples”, “usage examples”,

Table 3: Research strategies used to evaluate software visualization approaches.

<table>
<thead>
<tr>
<th>Category</th>
<th>Strategy</th>
<th>Reference #</th>
</tr>
</thead>
<tbody>
<tr>
<td>Theoretical</td>
<td>S1, S12</td>
<td></td>
</tr>
<tr>
<td>No Explicit Evaluation</td>
<td>S4, S7, S15, S16, S17, S18, S19, S20, S29, S44, S46, S49, S51, S53, S54, S65, S67, S79, S89, S92, S93, S111, S124, S132</td>
<td>24</td>
</tr>
<tr>
<td>Empirical</td>
<td>Survey S13, S71, S100, S113</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Anecdotal Evidence S55, S70, S73, S91, S97, S103</td>
<td>6</td>
</tr>
<tr>
<td></td>
<td>Case Study S52, S56, S58, S59, S64, S90, S105, S114, S115, S129, S151, S167</td>
<td>12</td>
</tr>
</tbody>
</table>

Table 4: Data collection methods used to evaluate software visualization approaches.

<table>
<thead>
<tr>
<th>Method</th>
<th>Reference #</th>
</tr>
</thead>
<tbody>
<tr>
<td>Think-Aloud</td>
<td>S40, S50, S100, S112, S117, S118, S123, S125, S126, S135, S141, S148, S150, S169, S173, S179, S180</td>
</tr>
<tr>
<td>Interview</td>
<td>S33, S71, S78, S90, S100, S106, S123, S127, S153, S174, S177, S180</td>
</tr>
<tr>
<td>Video Recording</td>
<td>S33, S50, S117, S125, S127, S140, S141, S144, S180</td>
</tr>
<tr>
<td>Sketch Drawing</td>
<td>S117, S127, S180</td>
</tr>
<tr>
<td>Others</td>
<td>Eye Tracking (S144), Log Analysis (S166), Feelings Cards (S180)</td>
</tr>
</tbody>
</table>
“application scenarios”, “analysis example”, “example of use”, “usage scenarios”, “application scenarios”, and “usage example”.

5.2.3. Survey

Only four studies (i.e., 2%) performed a survey, which is consistent with the findings of related work [19, 38]. Three of them [S13,S71,S100] surveyed developers to identify complex problems and collect requirements to design a proposed visualization approach: one focused on supporting development teams who use version control systems [S13], another asked former students of a course what they considered the most difficult subject in the lecture [S71], and another was concerned with understanding exception-handling constructs [S100]. In one study [S113] students who used a visualization approach were surveyed to collect anecdotal evidence of its usefulness. Two surveys [S71,S113] were conducted for visualization approaches that target the student audience in a software engineering course, while the remaining two [S13,S100] target the developer audience.

We found that surveys are used to identify frequent and complex problems that affect developers: such problems are then interpreted as requirements for a new visualization approach. We conjecture whether the low number of surveys has an effect on the disconnect between the proposed software visualization approaches and the needs of developers that we found in the past [23].

5.2.4. Case Study

We classified twelve papers (i.e., 7%) in the case study category. Usually, case studies are conducted to evaluate visualization approaches that target professional developers working on real-world projects in an industrial setting. The case of the study describes the context of the project in which difficulties arise, and shows how a visualization approach provides developers support for tackling them. We observed that in three studies [S56,S90,S114] some or all authors of the study come from industry, while in the rest there seems to be a strong relation of authors with industrial companies. In all of them, the evaluation involved professional developers.

5.2.5. Experiment

Fifty-three studies (i.e., 29%) evaluated software visualization via experiments. Although the level of detail varies, we identified a number of characteristics such as (i) data collection methods; (ii) type of analysis; (iii) participants; (iv) tasks; (v) dependent variables; and (vi) statistical tests. In the following we describe the results of the extracted data.

i) Participants. We observed a high variance in the number of participants in experiments (shown in Figure 4). The highest number of participants is found in a study [S25] that included 157 students. The minimum number corresponds to a study [S100] that involved three participants (graduate students with experience in industry). The median was 13 participants. A similar analysis of participants in the evaluation of information visualization approaches [12] shows similar results. Most evaluations of information visualization approaches involve 1–5 participants (excluding evaluations that do not report on the number of participants). The second most popular group includes 11–20 participants, and the group that includes 6–10 is the third most popular. Overall the median is 9 participants. Although many evaluations in software visualization included a number of participants in that range, the most popular ones are 6–10 and 11–20, followed by 21–30. One reason that might explain the difference could be that in our analysis we only included full papers that might present more thorough evaluations including a higher number of participants.

We noticed that experiments to evaluate software visualization approaches for teaching software engineering (e.g., algorithms and data structures) include a high number of participants since they usually involve a whole course and sometimes several of them. This type of experiment typically evaluates the effect of introducing visualization tools as a means for helping students to learn the subject of the
course. All of them found that visualizations do help students. However, they do not provide insights into whether the particular visualization technique tested in the experiment is the most suitable one. All experiments include participants selected from a convenience sample. Normally, they are students and academics at various levels with little experience working in industry.

ii) Type of Analysis. Table 5 presents our classification of the type of analysis adopted in experiments. We categorized the type of analysis into one of two categories: quantitative and qualitative. We found thirteen studies that adopted a quantitative analysis, while twenty-two used a qualitative one. In eighteen studies there was both a quantitative and qualitative analysis. Common examples of quantitative analyses in experiments include the measure of quantitative variables such as time and correctness. Typically, experiments were described as being formative or exploratory, and adopted a qualitative analysis of results (i.e., 75%). Several experiments also used a quantitative analysis to report evidence that supports the effectiveness of software visualization approaches. Although reporting on early results of preliminary evaluations has contributed important knowledge to the software visualization field, we believe that for software visualization approaches to become an actionable choice for developers, they have to present sound evidence of their effectiveness via surveys, controlled experiments, and case studies.

iii) Dependent Variables. Table 7 lists the dependent variables that were measured in experiments. We adopted the classification proposed by Lam et al. [15] and classified the dependent variables based on two of the proposed scenarios for evaluation of the understanding of visualizations: user performance and user experience. We found 35 (i.e., 66%) studies that evaluated user performance, 8 (i.e., 15%) evaluated user experience, and 10 (i.e., 19%) that evaluated variables of both. To evaluate performance most experiments defined as dependent variables correctness and time, some others specified that the experiment aimed at evaluating effectiveness without presenting details, and a few described multiple variables such as recollection, visual effort, scalability, and efficiency. To evaluate user experience researchers asked participants their perception of various variables such as usability, engagement, understandability, and emotions.

iv) Statistical Tests. Table 8 summarizes the statistical tests used in experiments for the quantitative analysis of data. We observed that the choice of the test is governed primarily by the number of dependent variables, their treatment and the type of the collected data (i.e., categorical, ordinal, interval). For instance, a questionnaire that uses a 5-step Likert scale to ask participants how suitable they find particular characteristics of a software visualization approach for a certain task would be ordinal. In that case, there would be one dependent variable, with five levels of ordinal data, for which the Kruskal-Wallis test would be a suitable match. Also, ANOVA is a common choice to test hypotheses. However, we observed that in some cases researchers found that parametric assumptions do not hold. Although there are alternative tests for non-parametric data, we observe that for data that do not follow a normal distribution, they can perform an Aligned Rank Transform [43] [S177].

v) Task. In table 9 the column Task summarizes exemplary tasks that we extracted from the design of each experiment. In almost half of the experiments (i.e., 26) we found explicit tasks that we identify with a check mark ✓. The remaining tasks that we list correspond to rationales that we inferred from analyzing the goals of experiments. We observed that in several studies participants were asked to use a visualization to lookup some aspects of the system. Although in some cases a database query might be a more effective tool than a visualization, we observed that these tasks are often used as a stepping stone towards complex tasks, in which developers certainly benefit from visualizing the context. For instance, participants used a visualization to answer questions where they had to:

a) count elements such as “how many packages are in the Java API?” [S125], “what is the number of packages?” [S164], “determine the total number of packages this system has” [S180], “how many methods does the largest class have (in terms of LOC)” [S144], and
b) find outliers such as “find the process with the longest duration.” [S32], “who are the top three most active code contributors?” [S108], “what are the two largest classes?” [S141], “name three applications that have a high fan-in” [S162], “find the three classes with the highest NOA” [S180].
We also observe that most studies build on these answers and ask participants to complete tasks that require them to explore. We believe that visualizations inherently excel in such tasks in contrast to text-based approaches. For instance, participants used visualizations to answer questions that involve:

a) Feature location such as “which method contains the logic to increase the speed?” [S50], “locate the feature that implements the logic: users are reminded that their accounts will be deleted if they do not log in after a certain number of months” [S117],

b) Change impact analysis such as “which classes of the package dependency will be directly affected by this change?” [S108], “analyze the impact of adding items to a playlist” [S78],

c) Analyze the rationale of an artifact such as “find the purpose of the given application” [S117], “what is the purpose of the application” [S162], and

d) Pattern detection such as “can you identify some interactions that are identical, along time, between groups of classes?” [S168], “find the most symmetric subtree in the tree” [S169], “locate the best candidate for the god class smell” [S180].

Moreover, we classify these tasks according to the taxonomy proposed by Munzner [29]. In it, she proposed that the task that motivates a visualization be classified using the following dimensions:

a) Analyze. The goal of a visualization can be to consume, that is, to discover new knowledge, present already discovered knowledge, and enjoy it; or it can be to create new material, which could be to annotate elements in the visualization, record visualization elements, and derive data elements from the existing ones.

b) Search. All analyses require users to search. However, the type of search can differ depending on whether the target of the search and the location of that target are known. When both the target and its location are known, it is called lookup. When the target is known but not its location, it is called locate. When the target is unknown but its location is known, it is called browse. Finally, when both target and its location are unknown, it is called explore.

c) Query. Once the searched targets are found, users query them. In tasks that involve a single target, the type of query is referred to as to identify. In tasks that involve two targets, it is referred to as to compare. Finally, in tasks that involve more than two targets, it is referred as to summarize.

We classify all tasks collected from the studies into the discovery category. The results of the classification in the remaining two dimensions is presented in Table 6. We observed that most of the tasks were designed to explore and summarize, that is, participants have to summarize many targets that they neither know, nor for which they know the location in the visualization. Almost half of the twenty-seven tasks in this category were explicitly described in the studies, while for the other half we only found a rationale. Tasks in this category tackle:

a) Comprehension [S23], [S24], [S25], [S32], [S33], [S40], [S61], [S96], [S106], [S148], [S154], [S174];

b) Change impact analysis [S50], [S78], [S118];

c) Debugging [S144], [S150], [S181];

d) Code Structure [S140], [S157];

e) Project Management [S166], [S169];

f) Rationale [S13], [S117], [S127], [S162]; and
g) Refactoring [S135].

We found seven other studies with tasks in which participants were asked to summarize targets but in which the targets were known, and therefore we classified them in the locate category. Studies in this category involve tasks that deal with:

a) Comprehension [126];

b) Debugging [S21], [S71];

c) Dependencies [100], [149];

d) Code structure [112]; and
e) Project Management [S179].

Only five studies involved tasks that asked participants to compare two targets. All of these tasks related to comprehension. Finally, the tasks of ten studies involved identifying a single target. These tasks deal with:

a) Comprehension [S11], [S101], [S173], [S180];

b) Change impact analysis [S177]; and
c) Debugging [S66], [S123], [S131], [S137], [S153].

Table 6: Classification of tasks used in experiments according to Munzner [29]

<table>
<thead>
<tr>
<th>Query</th>
<th>Search</th>
<th>Identify</th>
<th>Compare</th>
<th>Summarize</th>
</tr>
</thead>
<tbody>
<tr>
<td>Lookup</td>
<td>—</td>
<td>S5, S125</td>
<td>S108</td>
<td></td>
</tr>
<tr>
<td>Locate</td>
<td>S123, S168</td>
<td>S21, S71, S131, S137, S153, S177, S180</td>
<td>S100, S112, S126, S149, S179</td>
<td></td>
</tr>
<tr>
<td>Browse</td>
<td>S66, S47</td>
<td>S107, S141, S101</td>
<td>S164</td>
<td></td>
</tr>
</tbody>
</table>

10
6. Discussion

We now revisit our research questions. Firstly, we discuss the main characteristics that we found amongst the analyzed evaluations. Secondly, we discuss whether the conducted evaluations are appropriate considering their scope. Finally, we discuss the threats to the validity of our investigation.

*RQ1.* What are the characteristics of evaluations that validate the effectiveness of software visualization approaches?

Beyond traditional data collection methods. The methods used to collect data during the evaluation have to facilitate the subsequent analysis. Consequently, in a formative experiment researchers interview participants to freely explore aspects of complex phenomena. In a case study researchers can interview developers in their work environment, which can help researchers to formulate hypotheses that can be tested in experiments. Questionnaires can be used in surveys for exploration, reaching a higher number of participants who can provide researchers feedback of past experiences. We observed that several studies record sessions with participants. Afterwards, these records are used to dissect a user’s performance (e.g., correctness of answers and their completion time) and experience (e.g., level of engagement of participants with a tool). We observed that few non-traditional methods are used: (i) eye tracking to capture how participants see the elements in visualizations; (ii) log analysis to investigate how participants navigate visualizations; and (iii) emotion cards to help participants to report their feelings in a measurable fashion. Finally, we believe that the capabilities of recent devices used to display visualizations [21] (e.g., mobile phones, tablets, head-mounted displays [22]) can complement the standard computer screen, and provide researchers with useful data for investigating both user performance and user experience.

Thorough reports of anecdotal evidence and usage scenarios. Tool adoption can be considered the strongest evidence of the usability of an application [1]. However, we observe a lack of rigor amongst studies that reported anecdotal evidence. Normally, these studies report that tools were used, but often they do not specify the context, for instance, whether the tools are freely adopted or enforced as a requirement in a software engineering teaching course. Moreover, they describe subjective feedback from users using expressions such as “the tool was used with much success” [555], “feedback was positive” [597]. We propose that also reporting objective evidence, for instance number of downloads, would help them in making a stronger case to support the effectiveness of visualizations.

We also observed that one third of studies employed usage scenarios to demonstrate the effectiveness of the software visualization approaches. Typically they describe how the approach can answer questions about a software system. Normally, usage scenarios are carried out by the researchers themselves. Although researchers in the software visualization field are frequently both experts in software visualization and also experienced software developers, we believe they are affected by construction bias to perform the evaluation. Usage scenarios can help researchers to illustrate the applicability of a visualization approach. In fact, use cases that drive usage scenarios can reveal insights into the applicability of an visualization approach in an early stage [10]. Nonetheless, we believe they must involve external developers of the target audience who can produce a less biased evaluation, though related work [11] found that software engineering students can be used instead of professional software developers under certain conditions. We found multiple subject systems in usage scenarios, of which the most popular are open source. We reflect that open source software systems provide researchers an important resource for evaluating their proposed visualization approaches. They allow researchers to replicate evaluations in systems of various characteristics (e.g., size, complexity, architecture, language, domain). They also ease the reproducibility of studies. However, we think that defining a set of software systems to be used in benchmarks would facilitate comparison across software visualization evaluation [18, 21].

The value of visualizations beyond time and correctness. We believe that it is necessary to identify the requirements of developers and evaluate whether the functionality offered by a visualization tool is appropriate to the problem. Indeed, past research has found a large gap between the desired aspects and the features of current software visualization tools [3]. A later study [36] analyzed desirable features of software visualization tools for corrective maintenance. A subsequent study [13] analyzed the requirements of visualization tools for reverse engineering. We observed, however, little adoption of the proposed requirements. Usability is amongst them the most adopted one. Scalability was adopted only in one study [532]. Others such as interoperability, customizability, adoptability, integration, and query support were not found amongst the variables measured in experiments (see Table 7). We observed that even though none of the studies proposed that users of software visualizations should find answers quickly (i.e., time) and accurately (i.e., correctness), there are many evaluations that only considered these two variables.

We observed that evaluations in most studies aimed at proving the effectiveness of software visualization approaches. However, some studies do not specify how the effectiveness of the visualization is defined. Since something effective has “the power of acting upon the thing designated” [5], we reflect that effective visualization should fulfill its designated requirements. Then we ask what are the requirements of software visualization? We extract requirements from the dependent variables analyzed in experiments. We observed that the two main categories are user performance and user experience. Indeed, practitioners who adopt a visualization approach expect to find not only correct answers to software concerns, they expect that the visualization approach is also efficient (i.e., uses a minimal amount of resources), and helps them to find answers in a short amount of time [42]. However, they also aim at obtaining a good ex-

Table 7: A summary of the dependent variables found in experiments.

<table>
<thead>
<tr>
<th>Dependent Variable</th>
<th>References</th>
<th>#</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>User Performance</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Not Explicit</td>
<td>S96, S108</td>
<td>2</td>
</tr>
<tr>
<td>Time</td>
<td>S5, S11, S32, S40, S71, S107, S125, S137, S144, S162, S164, S173, S174, S177, S180</td>
<td>15</td>
</tr>
<tr>
<td>Effectiveness</td>
<td>S13, S21, S50, S66, S72, S78, S100, S101, S112, S127, S131, S141, S148, S157, S162, S164, S166</td>
<td>17</td>
</tr>
<tr>
<td>Completion</td>
<td>S50, S164</td>
<td>2</td>
</tr>
<tr>
<td>Recollection</td>
<td>S150, S180</td>
<td>2</td>
</tr>
<tr>
<td>Others</td>
<td>Visual Effort (S144), Scalability (S32), Efficiency (S32)</td>
<td>3</td>
</tr>
<tr>
<td><strong>User Experience</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Not Explicit</td>
<td>S96, S126, S49</td>
<td>3</td>
</tr>
<tr>
<td>Usability</td>
<td>S11, S13, S32, S40, S61, S117, S137, S140, S49, S153, S164, S169, S177, S181</td>
<td>14</td>
</tr>
<tr>
<td>Engagement</td>
<td>S154, S177</td>
<td>2</td>
</tr>
<tr>
<td>Understandability</td>
<td>S118, S181</td>
<td>2</td>
</tr>
<tr>
<td>Feeling</td>
<td>Enjoyment (S32), Intuitive (S137), Satisfaction (S164), Confidence (S107, S126)</td>
<td>5</td>
</tr>
<tr>
<td>Others</td>
<td>Acceptability (S164), Learnability (S164), Difficulty (S180)</td>
<td>3</td>
</tr>
</tbody>
</table>

Table 8: Statistical tests used to analyze data from experiments.

<table>
<thead>
<tr>
<th>Id.</th>
<th>Test</th>
<th>Reference</th>
<th>#</th>
</tr>
</thead>
<tbody>
<tr>
<td>T1</td>
<td>ANOVA</td>
<td>S25, S32, S40, S107, S144, S164, S174, S177, S180</td>
<td>9</td>
</tr>
<tr>
<td>T2</td>
<td>Pearson</td>
<td>S25, S40, S50, S107, S108, S150</td>
<td>6</td>
</tr>
<tr>
<td>T3</td>
<td>Cohen</td>
<td>S107, S150</td>
<td>2</td>
</tr>
<tr>
<td>T4</td>
<td>Wilcoxon</td>
<td>S101, S107, S126, S150, S164</td>
<td>5</td>
</tr>
<tr>
<td>T5</td>
<td>Student T</td>
<td>S5, S72, S101, S137, S162</td>
<td>5</td>
</tr>
<tr>
<td>T6</td>
<td>Shapiro-Wilk</td>
<td>S107, S162, S177, S180</td>
<td>4</td>
</tr>
<tr>
<td>T7</td>
<td>Kruskal-Wallis</td>
<td>S25, S108, S180</td>
<td>3</td>
</tr>
<tr>
<td>T8</td>
<td>Mann-Whitney</td>
<td>S25, S107, S168</td>
<td>3</td>
</tr>
<tr>
<td>T9</td>
<td>Descriptive Statistics and Charts</td>
<td>S24, S78, S118, S125, S131, S141, S154, S173, S179</td>
<td>9</td>
</tr>
<tr>
<td>T10</td>
<td>Levene</td>
<td>S162, S180</td>
<td>2</td>
</tr>
<tr>
<td>T11</td>
<td>Tukey (S180), Mauchly (S174), Greenhouse-Geisser (S174), Friedman (S21), Hotelling (S71), Kolmogorov-Smirnov (S72), Spearman (S25), Regression Analysis (S24)</td>
<td>8</td>
<td></td>
</tr>
</tbody>
</table>

- We believe that effective software visualization approaches must combine various complementary variables, which depend on the objective of the visualization. That is, variables used to explicitly define effectiveness relate to the domain problem and the tasks required by a particular target audience. We think that a deeper understanding of the mapping between users’ desired variables to usage scenarios of visualization can bring insights for defining quality metrics [4] in the software visualization field.

**The case in case studies.** We classified twelve papers into the case study category. In these papers, we identified a case that is neither hypothetical nor a toy example, but a concrete context that involves a real world system in which developers adopted a visualization approach to support answering complex questions. In only one paper [S90] did we find a thorough evaluation that describes the use of various research methods to collect data such as questionnaires and interviews. In contrast, in others we found less detail and no explicit description of the methods employed to collect data. In particular, in three papers [S52, S114, S151] a reference was given to a paper that contains more details. We observed that in studies in which authors come from industry [S56, S90, S114] there are many details provided as part of the evaluation. In all of them, (i) users who evaluated the proposed visualization approach were senior developers from industry, and (ii) the evaluation adopted a qualitative analysis. Case studies are often accused of lack of rigor since biased views of participants can influence the direction of the findings and conclusions [46]. Moreover, since they focus on a small number of subjects, they provide little basis for generalization.

In summary, we reflect on the need for conducting more case studies that can deliver insights into the benefits of software visualization approaches, and highlight the compulsion of identifying a concrete real-world case.
The scope of experiments in software visualization. Table 9 summarizes our extension to the framework proposed by Wohlin et al. [45] to include key characteristics of software visualizations. We believe that the extended framework can serve as a starting point for researchers who are planning to evaluate a software visualization approach. Each row in the table can be read as follows:


We used the framework to describe the scope of a recent experiment of 3D visualization in immersive augmented reality [20].

RQ2.) How appropriate are the evaluations that are conducted to validate the effectiveness of software visualization?

Explicit goal of evaluations. We observed that studies often do not explicitly specify the goal of the evaluation. They formulate sentences such as “To evaluate our visualization, we conducted interviews ...” [S100]. We investigate what aspects of the visualization are evaluated. We reflect that a clear and explicit formulation of the goal of the evaluation would help developers to assess if the evaluation provides them enough evidence that support the claimed benefits of a proposed visualization approach. Although in most studies we infer that the goal is to evaluate the effectiveness of a visualization, in only a few studies is there a definition of effectiveness. For instance, one study [S131] defines effectiveness of a visualization in terms of the number of statements that need to be read before identifying the location of an error; however, we believe this definition suits better the definition of efficiency. Indeed, practitioners will benefit from effective and efficient software visualization. Nonetheless, we believe the game-changing attribute of a visualization resides in the user experience, for which multiple variables should be included in evaluations (e.g., usability, engagement, emotions).

Experiments’ tasks must be in-line with evaluations’ goal. Software visualizations are proposed to support developers in tasks dealing with multiple development concerns. A problem thus arises for developers willing to adopt a visualization but who need to match a suitable visualization approach to their particular task at hand [24]. We investigate how suitable a visualization approach is for the tasks used in evaluations. We reflect that proving a software visualization approach to be effective for tasks for which there exist other more appropriate tools (but not included in the evaluation) can lead to misleading conclusions. Since many evaluations included in our analysis do not state an explicit goal, and some of the remaining ones refer to rather generic terms (e.g., effectiveness, usability) without providing a definition, understanding whether the tasks used in experiments are in-line with the goals of evaluations is still uncertain.

Beyond usage scenarios. Related work concluded that describing a case study is the most common strategy used to evaluate software visualization approaches. Indeed, we found many papers that contain a section entitled case study; however, we observed that most of them correspond to usage scenarios used to demonstrate how the proposed visualization approach is expected to be useful. In all of them, the authors (who usually are also developers) select a subject system and show how visualizations support a number of use cases. For example, one study [S158] describes the presence of independent judges, but without providing much detail about them. In the past, such a self-evaluation, known as an assertion [48], has been used in many studies, and is not considered an accepted research method for evaluation [44]. Instead, we prefer to refer to them as usage scenarios (as they are called in many studies). This name has also been adopted in the information visualization community [12], and therefore its adoption in software visualization will ease comparison across the two communities. Nonetheless, usage scenarios do not represent solid evidence of the benefits of proposed software visualization, and should be used only as a starting point to adjust requirements, and improve an approach.

Surveys to collect software visualization requirements. We observed that surveys are adequate to identifying requirements for software visualizations. Through a survey, the problems that arise in the development tasks carried out by a target audience that involve a particular data set can be collected as assessed as potential candidates for visualization. Then, researchers can propose an approach that defines the use of a visualization technique displayed in a medium. We observed that a main threat in software visualization is the disconnect between the development concerns that are the focus of visualization, and the most complex and frequent problems that arise during real-life development.

Report on thorough experiments. Although formative evaluations can be useful at an early stage, evidence of the user performance and user experience of a software visualization approach should be collected via thorough experiments (when variables included in the evaluation can be controlled). Experiments should include participants of a random sample of the target audience and real-world software systems. Experiments should aim at reproducibility, for which open source software projects are suitable. Moreover, open source projects boost replicability of evaluations across systems of various characteristics. The tasks used in experiments should be realistic, and as already discussed, consistent with the goal of the evaluation, otherwise conclusions can be misleading. Finally, we observed that standardizing evaluations via benchmarks would promote their comparison.

In summary, we observed that the main obstacles that prevent researchers from doing more appropriate evaluations are (i) the lack of a ready-to-use evaluation infrastructure, e.g., visualization tools to compare with; (ii) the lack of benchmarks.
<table>
<thead>
<tr>
<th>S</th>
<th>Task</th>
<th>Experiment</th>
<th>Scope</th>
<th>Participants</th>
<th>Environment</th>
<th>Visualization</th>
<th>Correctness</th>
<th>Time</th>
<th>Usefulness</th>
<th>Robustness</th>
<th>Usability</th>
<th>Effectiveness</th>
<th>Trade-offs</th>
</tr>
</thead>
<tbody>
<tr>
<td>S101</td>
<td>saUML</td>
<td>SCS</td>
<td>Correctness</td>
<td>All 20 CS students</td>
<td>T4</td>
<td>UML SCS</td>
<td>Correctness</td>
<td>All 20 CS students</td>
<td>T4</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>S108</td>
<td>Version Tree vs Augur</td>
<td>Windows</td>
<td>Performance</td>
<td>All 45 CS students</td>
<td>T1-T4,T6</td>
<td>SCS</td>
<td>Correctness, Time</td>
<td>All 45 CS students</td>
<td>T1-T4,T6</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>S117</td>
<td>CodeMap</td>
<td>SCS</td>
<td>Correctness</td>
<td>All 12 participants (ind. &amp; acad.)</td>
<td>T4</td>
<td>SCS</td>
<td>Correctness</td>
<td>All 12 participants (ind. &amp; acad.)</td>
<td>T4</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>S126</td>
<td>StenchBlossom</td>
<td>SCS</td>
<td>Correctness</td>
<td>All 8 par. (CS stud. &amp; resear.)</td>
<td>T8</td>
<td>SCS</td>
<td>Correctness</td>
<td>All 8 par. (CS stud. &amp; resear.)</td>
<td>T8</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>S127</td>
<td>Sfei</td>
<td>SCS</td>
<td>Correctness</td>
<td>All 12 participants (ind. &amp; acad.)</td>
<td>T4</td>
<td>SCS</td>
<td>Correctness</td>
<td>All 12 participants (ind. &amp; acad.)</td>
<td>T4</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>S13</td>
<td>Xia</td>
<td>SCS</td>
<td>Correctness</td>
<td>All 8 participants (ind. &amp; acad.)</td>
<td>T9</td>
<td>SCS</td>
<td>Correctness</td>
<td>All 8 participants (ind. &amp; acad.)</td>
<td>T9</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>S144</td>
<td>SeeIT3D</td>
<td>SCS</td>
<td>Correctness</td>
<td>All 8 participants (ind. &amp; acad.)</td>
<td>T9</td>
<td>SCS</td>
<td>Correctness</td>
<td>All 8 participants (ind. &amp; acad.)</td>
<td>T9</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>S148</td>
<td>ChronoTwigger</td>
<td>SCS</td>
<td>Correctness</td>
<td>All 8 participants (ind. &amp; acad.)</td>
<td>T9</td>
<td>SCS</td>
<td>Correctness</td>
<td>All 8 participants (ind. &amp; acad.)</td>
<td>T9</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>S149</td>
<td>FIt</td>
<td>SCS</td>
<td>Correctness</td>
<td>All 6 par. (CS stud. &amp; resear.)</td>
<td>T9</td>
<td>SCS</td>
<td>Correctness</td>
<td>All 6 par. (CS stud. &amp; resear.)</td>
<td>T9</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>S153</td>
<td>SIFEI</td>
<td>SCS</td>
<td>Correctness</td>
<td>All 6 par. (CS stud. &amp; resear.)</td>
<td>T9</td>
<td>SCS</td>
<td>Correctness</td>
<td>All 6 par. (CS stud. &amp; resear.)</td>
<td>T9</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>S154</td>
<td>TiledGrace</td>
<td>SCS</td>
<td>Correctness</td>
<td>All 6 par. (CS stud. &amp; resear.)</td>
<td>T9</td>
<td>SCS</td>
<td>Correctness</td>
<td>All 6 par. (CS stud. &amp; resear.)</td>
<td>T9</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>S164</td>
<td>VIMETRIK</td>
<td>SCS</td>
<td>Correctness</td>
<td>All 6 par. (CS stud. &amp; resear.)</td>
<td>T9</td>
<td>SCS</td>
<td>Correctness</td>
<td>All 6 par. (CS stud. &amp; resear.)</td>
<td>T9</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>S166</td>
<td>CuboidMatrix</td>
<td>SCS</td>
<td>Correctness</td>
<td>All 6 par. (CS stud. &amp; resear.)</td>
<td>T9</td>
<td>SCS</td>
<td>Correctness</td>
<td>All 6 par. (CS stud. &amp; resear.)</td>
<td>T9</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>S168</td>
<td>CuboidMatrix</td>
<td>SCS</td>
<td>Correctness</td>
<td>All 6 par. (CS stud. &amp; resear.)</td>
<td>T9</td>
<td>SCS</td>
<td>Correctness</td>
<td>All 6 par. (CS stud. &amp; resear.)</td>
<td>T9</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>S169</td>
<td>Perquimans</td>
<td>SCS</td>
<td>Correctness</td>
<td>All 6 par. (CS stud. &amp; resear.)</td>
<td>T9</td>
<td>SCS</td>
<td>Correctness</td>
<td>All 6 par. (CS stud. &amp; resear.)</td>
<td>T9</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>S172</td>
<td>GrowingSquares</td>
<td>SCS</td>
<td>Correctness</td>
<td>All 6 par. (CS stud. &amp; resear.)</td>
<td>T9</td>
<td>SCS</td>
<td>Correctness</td>
<td>All 6 par. (CS stud. &amp; resear.)</td>
<td>T9</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>S173</td>
<td>iTraceVis</td>
<td>SCS</td>
<td>Correctness</td>
<td>All 6 par. (CS stud. &amp; resear.)</td>
<td>T9</td>
<td>SCS</td>
<td>Correctness</td>
<td>All 6 par. (CS stud. &amp; resear.)</td>
<td>T9</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>S179</td>
<td>S179 iTraceVis</td>
<td>SCS</td>
<td>Correctness</td>
<td>All 6 par. (CS stud. &amp; resear.)</td>
<td>T9</td>
<td>SCS</td>
<td>Correctness</td>
<td>All 6 par. (CS stud. &amp; resear.)</td>
<td>T9</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>S181</td>
<td>MethodExecutionReports</td>
<td>SCS</td>
<td>Correctness</td>
<td>All 6 par. (CS stud. &amp; resear.)</td>
<td>T9</td>
<td>SCS</td>
<td>Correctness</td>
<td>All 6 par. (CS stud. &amp; resear.)</td>
<td>T9</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
that ease comparison across tools, e.g., quality metrics; (iii) the tradeoff between the effort of conducting comprehensive evaluations and little added value to paper acceptance; and (iv) the difficulties to involve industrial partners willing to share resources, e.g., include participants of the target audience.

6.1. Threats to Validity

Construct validity. Our research questions may not provide complete coverage of software visualization evaluation. We mitigated this threat by including questions that focus on the two main aspects that we found in related work: (I) characterization of the state-of-the-art, and (2) appropriateness of adopted evaluations.

Internal validity. We included papers from only two venues, and may have missed papers published in other venues that require more thorough evaluations. We mitigated this threat by identifying relevant software visualization papers that ensure an unbiased paper selection process. Therefore, we selected papers from the most frequently cited venue dedicated to software visualization: SOFTVIS/VISSOFT. We argue that even if we would have included papers from other venues the trend of the results would be similar. Indeed, related work did not find important differences when comparing software visualization evaluation in papers published in SOFTVIS/VISSOFT to papers published in other venues [19, 38]. Moreover, our results are in line with the conclusions of related work that have included papers from multiple venues [16, 30, 39]. We also mitigated the paper selection bias by selecting peer-reviewed full papers. We assessed the quality of these papers by excluding model papers (i.e., commentary, formalism, taxonomy) that are less likely to include an evaluation. However, since software visualization papers do not specify their types, we may have missed some. We mitigated this threat by defining a cross-checking procedure and criteria for paper type classification.

External validity. We selected software visualization papers published between 2002 to 2017 in SOFTVIS/VISSOFT. The excluded papers from other venues or published before 2002 may affect the generalizability of our results.

Conclusion validity. Bias in the data collection procedure could obstruct reproducibility of our study. We mitigated this threat by establishing a protocol to extract the data of each paper equally, and by maintaining a spreadsheet to keep records, normalize terms, and identify anomalies.

7. Conclusion

We reviewed 181 full papers of the 387 that were published to date in the SOFTVIS/VISSOFT conferences. We extracted evaluation strategies, data collection methods and other various aspects of evaluations. We found that 62% (i.e., 113) of the proposed software visualization approaches do not include a strong evaluation. We identified several pitfalls that must be avoided in the future of software visualization: (i) evaluations with fuzzy goals (or without explicit goals), for which the results are hard to interpret; (ii) evaluations that pursue effectiveness without defining it, or that limit the assessment to time, correctness (user performance) and usability (user experience) while disregarding many other variables that can contribute to effectiveness (e.g., recollection, engagement, emotions); (iii) experiment tasks that are inconsistent with the stated goal of the evaluation; (iv) lack of surveys to collect requirements that explain the disconnect between the problem domains on which software visualization have focused and the domains that get the most attention from practitioners; and (v) lack of rigor when designing, conducting, and reporting on evaluation.

We call researchers in the field to collect evidence of the effectiveness of software visualization approaches by means of (I) case studies (when there is a case that must be studied in situ), and (2) experiments (when variables can be controlled) including participants of a random sample of the target audience and real-world open source software systems that promote reproducibility and replicability.

We believe that our study will help (a) researchers to reflect on the design of appropriate evaluations for software visualization, and (b) developers to be aware of the evidence that supports the claims of benefit of the proposed software visualization approaches. We plan in the future to encapsulate the characterization and insights from this study in a software visualization ontology that will allow developers to find suitable visualizations for development concerns as well as researchers to reflect on the domain.

Acknowledgments

We gratefully acknowledge the financial support of the Swiss National Science Foundation for the project “Agile Software Analysis” (SNSF project No. 200020-162352, Jan 1, 2016 - Dec. 30, 2018). Merino has been partially funded by CONICYT BCH/Doctorado Extranjero 72140330.

References
