# A Systematic Literature Review of Software Visualization Evaluation

L. Merino[a], M. Ghafari[a], C. Anslow[b], O. Nierstrasz[a]

[a] *Software Composition Group, University of Bern, Switzerland*
[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 *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:

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

*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 audience, and included appropriate participants. 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-

[1] http://scg.unibe.ch/research/softvis-eval

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.

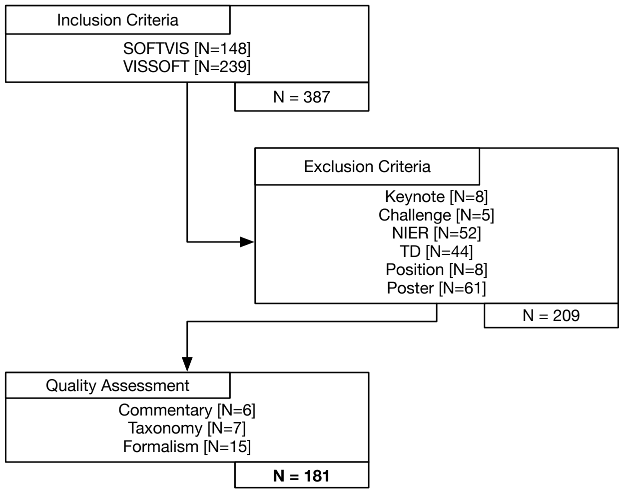
### 4.4. Data extraction

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:

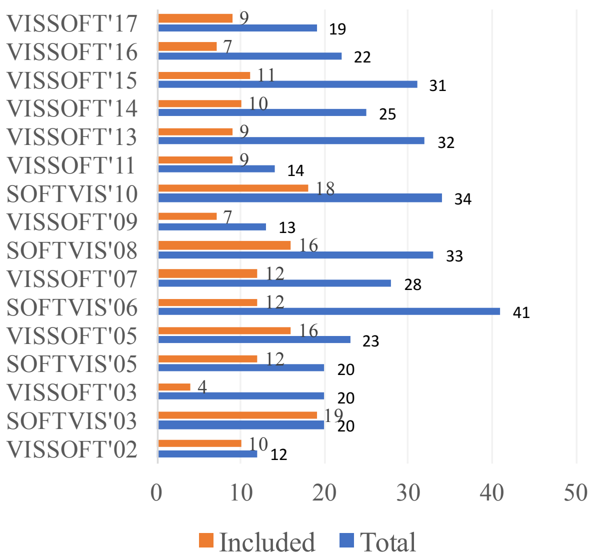
> *{evaluation, survey, [case|user] stud[y|ies], [application | usage | analysis] example[s], use case[s], application scenario[s], [controlled | user] experiment, demonstration, user scenario[s], example of use, usage scenario[s], example scenario[s], demonstrative result[s]}*

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

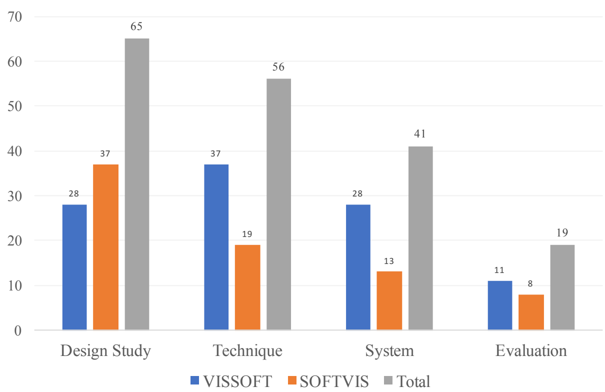
[4] http://portal.core.edu.au/conf-ranks/




(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.

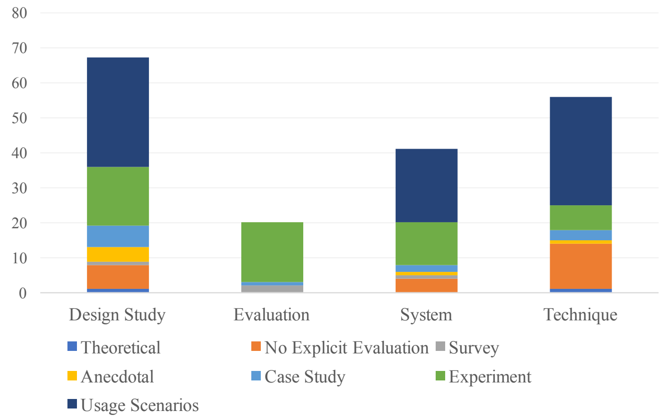



Figure 2: The distribution of the 181 included papers categorized by paper types and research strategy used to evaluate software visualization approaches.

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).

Table 1: The papers included in the study [S1-S107].

| Id and Reference | Venue | Evaluation |
|---|---|---|
| [S1] Aesthetics of class diagrams, **Eichelberger, H.** | V'02 | Theorical |
| [S2] Specifying algorithm visualizations in terms of dat..., **Francik, J.** | V'02 | Usage Scenario |
| [S3] View definitions for language-independent multipl..., **Sajaniemi, J.** | V'02 | Usage Scenario |
| [S4] The CONCEPT project - applying source code analysis to..., **Rilling, J. *et al.*** | V'02 | - |
| [S5] UML collaboration diagram syntax: an empir..., **Purchase, H.C. *et al.*** | V'02 | Experiment |
| [S6] Runtime visualisation of object oriented soft..., **Smith, M.P. *et al.*** | V'02 | Usage Scenario |
| [S7] Reification of program points for visual execution , **Diehl, S. *et al.*** | V'02 | - |
| [S8] Metrics-based 3D visualization of large obj..., **Lewerentz, C. *et al.*** | V'02 | Usage Scenario |
| [S9] Analogical representations of programs, **Ploix, D.** | V'02 | Usage Scenario |
| [S10] Revision Towers, **Taylor, C.M.B. *et al.*** | V'02 | Usage Scenario |
| [S11] Self-Organizing Maps Applied in Visualising ..., **Brittle, J. *et al.*** | V'03 | Experiment |
| [S12] KScope: A Modularized Tool for 3D Visualizati..., **Davis, T.A. *et al.*** | V'03 | Theorical |
| [S13] Visualization to Support Version Control Software..., **Wu, X. *et al.*** | V'03 | Experiment |
| [S14] Techniques for Reducing the Complexity o..., **Hamou-Lhadj, A. *et al.*** | V'03 | Usage Scenario |
| [S15] A topology-shape-metrics approach for the automa..., **Eiglsperger, M. *et al.*** | S'03 | - |
| [S16] A new approach for visualizing UML class diagrams, **Gutwenger, C. *et al.*** | S'03 | - |
| [S17] Visualizing model mappings in UML, **Hausmann, J.H. *et al.*** | S'03 | - |
| [S18] Visualizing software for telecommunication services..., **Gansner, E.R. *et al.*** | S'03 | - |
| [S19] Graph visualization for the analysis of the structure an..., **Zhou, C. *et al.*** | S'03 | - |
| [S20] Interactive locality optimization on NUMA architectures, **Mu, T. *et al.*** | S'03 | - |
| [S21] End-user software visualizations for fault ..., **Ruthruff, J. *et al.*** | S'03 | Experiment |
| [S22] Interactive visual debugging with UML, **Jacobs, T. *et al.*** | S'03 | Usage Scenario |
| [S23] Designing effective program visualization too..., **Tudoreanu, M.E.** | S'03 | Experiment |
| [S24] Dancing hamsters and marble statue..., **Huebscher-Younger, T. *et al.*** | S'03 | Experiment |
| [S25] Algorithm visualization in CS education: com..., **Grissom, S. *et al.*** | S'03 | Experiment |
| [S26] A system for graph-based visualization of t..., **Collberg, C. *et al.*** | S'03 | Usage Scenario |
| [S27] Visualization of program-execution data for dep..., **Orso, A. *et al.*** | S'03 | Usage Scenario |
| [S28] Visualizing Java in action, **Reiss, S.P.** | S'03 | Usage Scenario |
| [S29] Plugging-in visualization: experiences integrating a ..., **Lintern, R. *et al.*** | S'03 | - |
| [S30] EVolve: an open extensible software visualizatio..., **Wang, Q. *et al.*** | S'03 | Usage Scenario |
| [S31] 3D representations for software visualization..., **Marcus, A. *et al.*** | S'03 | Usage Scenario |
| [S32] Growing squares: animated visualization of ..., **Elmqvist, N. *et al.*** | S'03 | Experiment |
| [S33] Program animation based on the roles of va..., **Sajaniemi, J. *et al.*** | S'03 | Experiment |
| [S34] Visualizing Feature Interaction in 3-D, **Greevy, O. *et al.*** | V'05 | Usage Scenario |
| [S35] Identifying Structural Features of Java Prog..., **Smith, M.P. *et al.*** | V'05 | Usage Scenario |
| [S36] Support for Static Concept Location with sv3D, **Xie, X. *et al.*** | V'05 | Usage Scenario |
| [S37] Interactive Exploration of Semantic Clusters, **Lungu, M. *et al.*** | V'05 | Usage Scenario |
| [S38] Exploring Relations within Software Systems ..., **Balzer, M. *et al.*** | V'05 | Usage Scenario |
| [S39] The Dominance Tree in Visualizing Software Dep..., **Falke, R. *et al.*** | V'05 | Usage Scenario |
| [S40] User Perspectives on a Visual Aid to Program Com..., **Cox, A. *et al.*** | V'05 | Experiment |
| [S41] Interactive Visual Mechanisms for Exploring So..., **Telea, A. *et al.*** | V'05 | Usage Scenario |
| [S42] Fractal Figures: Visualizing Development Ef..., **D'Ambros, M. *et al.*** | V'05 | Usage Scenario |
| [S43] White Coats: Web-Visualization of Evolving S..., **Mesnage, C. *et al.*** | V'05 | Usage Scenario |
| [S44] Multi-level Method Understanding Using Microprints , **Ducasse, S. *et al.*** | V'05 | - |
| [S45] Visual Realism for the Visualization of Softwa..., **Holten, D. *et al.*** | V'05 | Usage Scenario |
| [S46] Visual Exploration of Combined Architectural and Met..., **Termeer, M. *et al.*** | V'05 | - |
| [S47] Evaluating UML Class Diagram Layout base..., **Andriyevska, O. *et al.*** | V'05 | Experiment |
| [S48] Interactive Exploration of UML Sequence Diagra..., **Sharp, R. *et al.*** | V'05 | Usage Scenario |
| [S49] SAB - The Software Architecture Browser, **Erben, N. *et al.*** | V'05 | - |
| [S50] Towards understanding programs through wear-b..., **DeLine, R. *et al.*** | S'05 | Experiment |
| [S51] Online-configuration of software visualizations with Vi..., **Panas, T. *et al.*** | S'05 | - |
| [S52] Visualization of mobile object environments..., **Frishman, Y. *et al.*** | S'05 | Case Study |
| [S53] Visualizing structural properties of irregular par..., **Blochinger, W. *et al.*** | S'05 | - |
| [S54] Jove: java as it happens, **Reiss, S.P. *et al.*** | S'05 | - |
| [S55] Methodology and architecture of JIVE, **Gestwicki, P. *et al.*** | S'05 | Anecdotal |
| [S56] Visual specification and analysis of use cas..., **Kholkar, D. *et al.*** | S'05 | Case Study |
| [S57] Visualizing multiple evolution metrics, **Pinzger, M. *et al.*** | S'05 | Usage Scenario |
| [S58] The war room command console: shared visual..., **O'Reilly, C. *et al.*** | S'05 | Case Study |
| [S59] CVSscan: visualization of code evolution, **Voinea, L. *et al.*** | S'05 | Case Study |
| [S60] Visual data mining in software archives, **Burch, M. *et al.*** | S'05 | Usage Scenario |
| [S61] Algorithm visualization using concept keyboa..., **Baloian, N. *et al.*** | S'05 | Experiment |
| [S62] Mondrian: an agile information visualization f..., **Meyer, M. *et al.*** | S'06 | Usage Scenario |
| [S63] Multiscale and multivariate visualizations of ..., **Voinea, L. *et al.*** | S'06 | Usage Scenario |
| [S64] Visualization of areas of interest in softwar..., **Byelas, H. *et al.*** | S'06 | Case Study |
| [S65] Visual exploration of function call graphs for feature..., **Bohnet, J. *et al.*** | S'06 | - |
| [S66] Using social agents to visualize software..., **Alspaugh, T.A. *et al.*** | S'06 | Experiment |
| [S67] Transparency, holophrasting, and automatic layout appl..., **Gauvin, S. *et al.*** | S'06 | - |
| [S68] A data-driven graphical toolkit for softwa..., **Demetrescu, C. *et al.*** | S'06 | Usage Scenario |
| [S69] Visualizing live software systems in 3D, **Greevy, O. *et al.*** | S'06 | Usage Scenario |
| [S70] Execution patterns for visualizing web servic..., **de Pauw, W. *et al.*** | S'06 | Anecdotal |
| [S71] Experimental evaluation of animated-verifying o..., **Jain, J. *et al.*** | S'06 | Experiment |
| [S72] Narrative algorithm visualization, **Blumenkrants, M. *et al.*** | S'06 | Experiment |
| [S73] The Clack graphical router: visualizing net..., **Wendlandt, D. *et al.*** | S'06 | Anecdotal |
| [S74] A Visualization for Software Project Awaren..., **Ripley, R.M. *et al.*** | V'07 | Usage Scenario |
| [S75] YARN: Animating Software Evolution, **Hindle, A. *et al.*** | V'07 | Usage Scenario |
| [S76] DiffArchViz: A Tool to Visualize Correspondence ..., **Sawant, A.P.** | V'07 | Usage Scenario |
| [S77] A Bug's Life" Visualizing a Bug Database""A..., **D'Ambros, M. *et al.*** | V'07 | Usage Scenario |
| [S78] Task-specific source code dependency investig..., **Holmes, R. *et al.*** | V'07 | Experiment |
| [S79] Visualizing Software Systems as Cities, **Wettel, R. *et al.*** | V'07 | - |
| [S80] Onion Graphs for Focus+Context Views of UML Cl..., **Kagdi, H. *et al.*** | V'07 | Usage Scenario |
| [S81] CocoViz: Towards Cognitive Software Visuali..., **Boccuzzo, S. *et al.*** | V'07 | Usage Scenario |
| [S82] Distributable Features View: Visualizing the..., **Cosma, D.C. *et al.*** | V'07 | Usage Scenario |
| [S83] Trace Visualization Using Hierarchical Edge B..., **Holten, D. *et al.*** | V'07 | Usage Scenario |
| [S84] Visualization of Dynamic Program Aspects, **Deelen, P. *et al.*** | V'07 | Usage Scenario |
| [S85] Visualizing Dynamic Memory Allocations, **Moreta, S. *et al.*** | V'07 | Usage Scenario |
| [S86] Applying visualisation techniques in software..., **Nestor, D. *et al.*** | S'08 | Usage Scenario |
| [S87] Stacked-widget visualization of scheduling-..., **Bernardin, T. *et al.*** | S'08 | Usage Scenario |
| [S88] Visually localizing design problems with dish..., **Wettel, R. *et al.*** | S'08 | Usage Scenario |
| [S89] Visualizing inter-dependencies between scenarios, **Harel, D. *et al.*** | S'08 | - |
| [S90] Software visualization for end-user pr..., **Subrahmaniyan, N. *et al.*** | S'08 | Case Study |
| [S91] Streamsight: a visualization tool for large-s..., **de Pauw, W. *et al.*** | S'08 | Anecdotal |
| [S92] Improving an interactive visualization of transition ..., **Ploeger, B. *et al.*** | S'08 | - |
| [S93] Automatic layout of UML use case diagrams, **Eichelberger, H.** | S'08 | - |
| [S94] Gef3D: a framework for two-, two-and-a-h..., **von Pilgrim, J. *et al.*** | S'08 | Usage Scenario |
| [S95] A catalogue of lightweight visualizations to ..., **Parnin, C. *et al.*** | S'08 | Usage Scenario |
| [S96] An interactive reverse engineering environment..., **Telea, A. *et al.*** | S'08 | Experiment |
| [S97] Representing unit test data for large scale ..., **Cottam, J.A. *et al.*** | S'08 | Anecdotal |
| [S98] HDPV: interactive, faithful, in-vivo run..., **Sundararaman, J. *et al.*** | S'08 | Usage Scenario |
| [S99] Analyzing the reliability of communication be..., **Zeckzer, D. *et al.*** | S'08 | Usage Scenario |
| [S100] Visualization of exception handling constructs..., **Shah, H. *et al.*** | S'08 | Experiment |
| [S101] Assessing the benefits of synchronization-adorn..., **Xie, S. *et al.*** | S'08 | Experiment |
| [S102] Extraction and visualization of call dependen..., **Telea, A. *et al.*** | V'09 | Usage Scenario |
| [S103] Visualizing the Java heap to detect memory proble..., **Reiss, S.P.** | V'09 | Anecdotal |
| [S104] Case study: Visual analytics in software prod..., **Telea, A. *et al.*** | V'09 | Usage Scenario |
| [S105] Visualizing massively pruned execution trace..., **Bohnet, J. *et al.*** | V'09 | Case Study |
| [S106] Evaluation of software visualization tool..., **Sensalire, M. *et al.*** | V'09 | Experiment |
| [S107] The effect of layout on the comprehension of..., **Sharif, B. *et al.*** | V'09 | Experiment |

Table 2: The papers included in the study [S108-S181].

| Id and Reference | Venue | Evaluation |
|---|---|---|
| [S108] Beyond pretty pictures: Examining the benef..., **Yunrim Park *et al.*** | V'09 | Experiment |
| [S109] Representing development history in s..., **Steinbrueckner, F. *et al.*** | S'10 | Usage Scenario |
| [S110] Visual comparison of software architectures, **Beck, F. *et al.*** | S'10 | Usage Scenario |
| [S111] An automatic layout algorithm for BPEL processes, **Albrecht, B. *et al.*** | S'10 | - |
| [S112] Off-screen visualization techniques for clas..., **Frisch, M. *et al.*** | S'10 | Experiment |
| [S113] Jype - a program visualization and programm..., **Helminen, J. *et al.*** | S'10 | Survey |
| [S114] Zinsight: a visual and analytic environment..., **de Pauw, W. *et al.*** | S'10 | Case Study |
| [S115] Understanding complex multithreaded softwa..., **Truemper, J. *et al.*** | S'10 | Case Study |
| [S116] Visualizing windows system traces, **Wu, Y. *et al.*** | S'10 | Usage Scenario |
| [S117] Embedding spatial software visualization in th..., **Kuhn, A. *et al.*** | S'10 | Experiment |
| [S118] Towards anomaly comprehension: using structural..., **Lin, S. *et al.*** | S'10 | Experiment |
| [S119] Dependence cluster visualization, **Islam, S.S. *et al.*** | S'10 | Usage Scenario |
| [S120] Exploring the inventor's paradox: applying jig..., **Ruan, H. *et al.*** | S'10 | Usage Scenario |
| [S121] Trevis: a context tree visualization & anal..., **Adamoli, A. *et al.*** | S'10 | Usage Scenario |
| [S122] Heapviz: interactive heap visualizati..., **Aftandilian, E.E. *et al.*** | S'10 | Usage Scenario |
| [S123] AllocRay: memory allocation visualizati..., **Robertson, G.G. *et al.*** | S'10 | Experiment |
| [S124] Software evolution storylines, **Ogawa, M. *et al.*** | S'10 | - |
| [S125] User evaluation of polymetric views using a ..., **Anslow, C. *et al.*** | S'10 | Experiment |
| [S126] An interactive ambient visualization fo..., **Murphy-Hill, E. *et al.*** | S'10 | Experiment |
| [S127] Follow that sketch: Lifecycles of diagrams an..., **Walny, J. *et al.*** | V'11 | Experiment |
| [S128] Visual support for porting large code base..., **Broeksema, B. *et al.*** | V'11 | Usage Scenario |
| [S129] A visual analysis and design tool for planning..., **Beck, M. *et al.*** | V'11 | Case Study |
| [S130] Visually exploring multi-dimensional code coup..., **Beck, F. *et al.*** | V'11 | Usage Scenario |
| [S131] Constellation visualization: Augmenting progra..., **Deng, F. *et al.*** | V'11 | Experiment |
| [S132] 3D Hierarchical Edge bundles to visualize relations ..., **Caserta, P. *et al.*** | V'11 | - |
| [S133] Abstract visualization of runtime m..., **Choudhury, A.N.M.I. *et al.*** | V'11 | Usage Scenario |
| [S134] Telling stories about GNOME with Complicity, **Neu, S. *et al.*** | V'11 | Usage Scenario |
| [S135] E-Quality: A graph based object oriented so..., **Erdemir, U. *et al.*** | V'11 | Experiment |
| [S136] Automatic categorization and visualization o..., **Reiss, S.P. *et al.*** | V'13 | Usage Scenario |
| [S137] Using HTML5 visualizations in software faul..., **Gouveia, C. *et al.*** | V'13 | Experiment |
| [S138] Visualizing jobs with shared resources in di..., **de Pauw, W. *et al.*** | V'13 | Usage Scenario |
| [S139] SYNCTRACE: Visual thread-interplay analysis, **Karran, B. *et al.*** | V'13 | Usage Scenario |
| [S140] Finding structures in multi-type code c..., **Abuthawabeh, A. *et al.*** | V'13 | Experiment |
| [S141] SourceVis: Collaborative software visualizat..., **Anslow, C. *et al.*** | V'13 | Experiment |
| [S142] Visualizing software dynamicities with heat..., **Benomar, O. *et al.*** | V'13 | Usage Scenario |
| [S143] Performance evolution blueprint: Underst..., **Sandoval, J.P. *et al.*** | V'13 | Usage Scenario |
| [S144] An empirical study assessing the effect of s..., **Sharif, B. *et al.*** | V'13 | Experiment |
| [S145] Visualizing Developer Interactions, **Minelli, R. *et al.*** | V'14 | Usage Scenario |
| [S146] AniMatrix: A Matrix-Based Visualization of ..., **Rufiange, S. *et al.*** | V'14 | Usage Scenario |
| [S147] Visualizing the Evolution of Systems and The..., **Kula, R.G. *et al.*** | V'14 | Usage Scenario |
| [S148] ChronoTwigger: A Visual Analytics Tool for Unde..., **Ens, B. *et al.*** | V'14 | Experiment |
| [S149] Lightweight Structured Visualization of Asse..., **Toprak, S. *et al.*** | V'14 | Experiment |
| [S150] How Developers Visualize Compiler Messages: A..., **Barik, T. *et al.*** | V'14 | Experiment |
| [S151] Feature Relations Graphs: A Visualisation ..., **Martinez, J. *et al.*** | V'14 | Case Study |
| [S152] Search Space Pruning Constraints Visualizati..., **Haugen, B. *et al.*** | V'14 | Usage Scenario |
| [S153] Integrating Anomaly Diagnosis Techniques int..., **Kulesz, D. *et al.*** | V'14 | Experiment |
| [S154] Combining Tiled and Textual Views of Code, **Homer, M. *et al.*** | V'14 | Experiment |
| [S155] Visualizing Work Processes in Software Engine..., **Burch, M. *et al.*** | V'15 | Usage Scenario |
| [S156] Blended, Not Stirred: Multi-concern Visua..., **Dal Sasso, T. *et al.*** | V'15 | Usage Scenario |
| [S157] CodeSurveyor: Mapping Large-Scale Software to..., **Hawes, N. *et al.*** | V'15 | Experiment |
| [S158] Revealing Runtime Features and Constituent..., **Palepu, V.K. *et al.*** | V'15 | Usage Scenario |
| [S159] A Visual Support for Decomposing Complex Featu..., **Urli, S. *et al.*** | V'15 | Usage Scenario |
| [S160] Visualising Software as a Particle System, **Scarle, S. *et al.*** | V'15 | Usage Scenario |
| [S161] Interactive Tag Cloud Visualization of Sof..., **Greene, G.J. *et al.*** | V'15 | Usage Scenario |
| [S162] Hierarchical Software Landscape Visualizati..., **Fittkau, F. *et al.*** | V'15 | Experiment |
| [S163] Vestige: A Visualization Framework for Eng..., **Schneider, T. *et al.*** | V'15 | Usage Scenario |
| [S164] Visual Analytics of Software Structure and Met..., **Khan, T. *et al.*** | V'15 | Experiment |
| [S165] Stable Voronoi-Based Visualizations for Sof..., **Van Hees, R. *et al.*** | V'15 | Usage Scenario |
| [S166] Visualizing the Evolution of Working Sets, **Minelli, R. *et al.*** | V'16 | Experiment |
| [S167] Walls, Pillars and Beams: A 3D Decompositio..., **Tymchuk, Y. *et al.*** | V'16 | Case Study |
| [S168] CuboidMatrix: Exploring Dynamic Structura..., **Schneider, T. *et al.*** | V'16 | Experiment |
| [S169] A Tool for Visualizing Patterns of Spread..., **Middleton, J. *et al.*** | V'16 | Experiment |
| [S170] Jsvee & Kelmu: Creating and Tailoring Program Ani..., **Sirkiae, T.** | V'16 | Usage Scenario |
| [S171] Visualizing Project Evolution through Abstr..., **Feist, M.D. *et al.*** | V'16 | Usage Scenario |
| [S172] Merge-Tree: Visualizing the Integration of Com..., **Wilde, E. *et al.*** | V'16 | Usage Scenario |
| [S173] A Scalable Visualization for Dynamic Data in ..., **Burch, M. *et al.*** | V'17 | Experiment |
| [S174] An Empirical Study on the Readability of R..., **Hollmann, N. *et al.*** | V'17 | Experiment |
| [S175] Concept-Driven Generation of Intuitive Explana..., **Reza, M. *et al.*** | V'17 | Usage Scenario |
| [S176] Visual Exploration of Memory Traces and Call ..., **Gralka, P. *et al.*** | V'17 | Usage Scenario |
| [S177] Code Park: A New 3D Code Visualization Tool..., **Khaloo, P. *et al.*** | V'17 | Experiment |
| [S178] Using High-Rising Cities to Visualize Perform..., **Ogami, K. *et al.*** | V'17 | Usage Scenario |
| [S179] iTraceVis: Visualizing Eye Movement Data With..., **Clark, B. *et al.*** | V'17 | Experiment |
| [S180] On the Impact of the Medium in the Effective..., **Merino, L. *et al.*** | V'17 | Experiment |
| [S181] Method Execution Reports: Generating Text and ..., **Beck, F. *et al.*** | V'17 | Experiment |

## 5.2. *Evaluation Strategies*

In twenty-four (*i.e.*, 13%) studies we did not find an explicit evaluation that presents evidence for supporting the claim of effectiveness of software visualization approaches. These studies indicate that the evaluation of the proposed visualization is planned as future work. In the remaining studies, we found that several strategies were used to evaluate software visualization approaches. We observed that only two studies (*i.e.*, 1%) used *theoretical* references to support the claim of the effectiveness of software visualizations. One technique paper *[S1]* that proposes aesthetic criteria for class diagrams, considered their proposed criteria effective since it was derived from the UML specification, and one design study paper *[S12]* evaluated the visualization based on previously proposed criteria for visualizing software in virtual reality [47]. Both studies planned

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

| Category | Strategy | Reference | # |
|---|---|---|---|
| Theoretical | | S1, S12 | 2 |
| No Explicit Evaluation | | S4, S7, S15, S16, S17, S18, S19, S20, S29, S44, S46, S49, S51, S53, S54, S65, S67, S79, S89, S92, S93, S111, S124, S132 | 24 |
| Empirical | Survey | S13, S71, S100, S113 | 4 |
| | Anecdotal Evidence | S55, S70, S73, S91, S97, S103 | 6 |
| | Case Study | S52, S56, S58, S59, S64, S90, S105, S114, S115, S129, S151, S167 | 12 |
| | Experiment | S5, S11, S13, S21, S23, S24, S25, S32, S33, S40, S47, S50, S61, S66, S71, S72, S78, S96, S100, S101, S106, S107, S108, S112, S117, S118, S123, S125, S126, S127, S131, S135, S137, S140, S141, S144, S148, S149, S150, S153, S154, S157, S162, S164, S166, S168, S169, S173, S174, S177, S179, S180, S181 | 53 |
| | Example | S57, S60, S62, S63, S68, S69, S74, S75, S76, S77, S80, S81, S82, S83, S84, S85, S86, S87, S88, S94, S95, S98, S99, S102, S104, S109, S110, S116, S119, S120, S121, S122, S128, S130, S133, S134, S136, S138, S139, S142, S143, S145, S146, S147, S152, S155, S156, S158, S159, S160, S161, S163, S165, S170, S171, S172, S175, S176, S178 | 83 |

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

| Method | Reference | # |
|---|---|---|
| Questionnaire | S11, S13, S25, S32, S40, S47, S50, S61, S66, S72, S90, S100, S106, S107, S108, S112, S125, S126, S127, S135, S137, S140, S141, S144, S149, S150, S153, S154, S157, S162, S164, S168, S173, S177, S179, S180, S181 | 37 |
| Think-Aloud | S40, S50, S100, S112, S117, S118, S123, S125, S126, S135, S141, S148, S150, S169, S173, S179, S180 | 17 |
| Interview | S33, S71, S78, S90, S100, S106, S123, S127, S153, S174, S177, S180 | 12 |
| Video Recording | S33, S50, S117, S125, S127, S140, S141, S144, S180 | 9 |
| Sketch Drawing | S117, S127, S180 | 3 |
| Others | Eye Tracking (S144), Log Analysis (S166), Feelings Cards (S180) | 3 |

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 strategiesWe 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”,

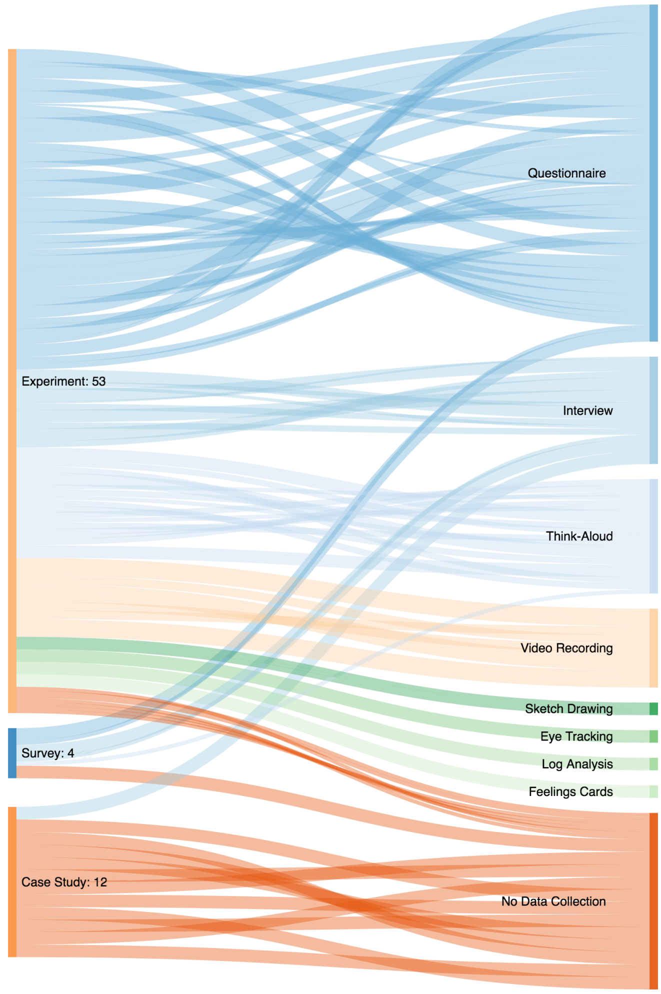



Figure 3: Sankey diagram showing the data collection methods (right) employed in evaluation strategies (left) adopted in empirical evaluations.

"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 ranges, 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

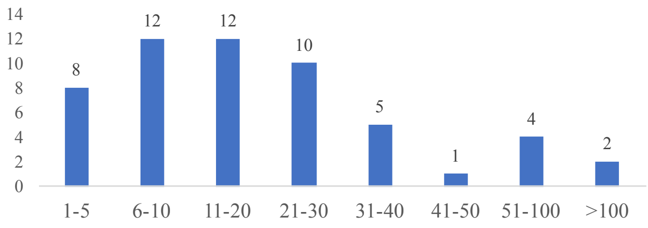



Figure 4: Histogram of the number of participants reported in evaluation.

Table 5: Type of analysis adopted in experiments.

| **Type of Analysis** | **References** | **#** |
|---|---|---|
| Quantitative | S21, S23, S24, S25, S71, S78, S101, S107, S137, S150, S154, S164, S174 | 13 |
| Qualitative | S11, S13, S33, S61, S66, S96, S100, S106, S112, S117, S123, S127, S135, S140, S141, S148, S149, S153, S157, S166, S169, S181 | 22 |
| Quantitative / Qualitative | S5, S32, S40, S47, S50, S72, S108, S118, S125, S126, S131, S144, S162, S168, S173, S177, S179, S180 | 18 |

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:

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

| Query \ **Search** | Identify | Compare | Summarize |
|---|---|---|---|
| Lookup | — | S5, S125 | S108 |
| Locate | S123, S131, S137, S153, S177, S180 | S168 | S21, S71, S100, S112, S126, S149, S179 |
| Explore | S11, S173 | S72 | S13, S23, S24, S25, S32, S33, S40, S50, S61, S78, S96, S106, S117, S118, S127, S135, S140, S144, S148, S150, S154, S157, S162, S166, S169, S174, S181 |
| Browse | S66, S101 | S47 | S107, S141, S164 |

*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]*.

## 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" *[S55]*, "feedback was positive" *[S97]* 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 *[S32]*. 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-

[5]"effective, adj. and n." OED Online. Oxford University Press, June 2017. Accessed October 27, 2017.

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

| Dependent Variable | | References | # |
|---|---|---|---|
| User Performance | Not Explicit | S96, S108 | 2 |
| | Time | S5, S11, S32, S40, S71, S107, S125, S137, S144, S162, S164, S173, S174, S177, S180 | 15 |
| | Correctness | S5, S11, S13, S21, S24, S25, S32, S33, S40, S47, S71, S72, S78, S101, S106, S107, S108, S118, S123, S125, S126, S137, S144, S150, S162, S164, S168, S173, S179, S180 | 29 |
| | Effectiveness | S13, S21, S50, S66, S72, S78, S100, S101, S112, S127, S131, S141, S148, S157, S162, S164, S166 | 17 |
| | Completion | S50,S164 | 2 |
| | Recollection | S150,S180 | 2 |
| | Others | Visual Effort (S144), Scalability (S32), Efficiency (S32) | 3 |
| User Experience | Not Explicit | S96, S126, S49 | 3 |
| | Usability | S11, S13, S32, S40, S61, S117, S137, S140, S49, S153, S164, S169, S177, S181 | 14 |
| | Engagement | S154, S177 | 2 |
| | Understandability | S118, S181 | 2 |
| | Feeling | Enjoyment (S32), Intuitive (S137), Satisfaction (S164), Confidence (S107, S126) | 5 |
| | Others | Acceptability (S164), Learnability (S164), Difficulty (S180) | 3 |

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

| Id. | Test | Reference | # |
|---|---|---|---|
| T1 | ANOVA | S25, S32, S40, S107, S144, S164, S174, S177, S180 | 9 |
| T2 | Pearson | S25, S40, S50, S107, S108, S150 | 6 |
| T3 | Cohen | S107, S150 | 2 |
| T4 | Wilcoxon | S101, S107, S126, S150, S164 | 5 |
| T5 | Student T | S5, S72, S101, S137, S162 | 5 |
| T6 | Shapiro-Wilk | S107, S162, S177, S180 | 4 |
| T7 | Kruskal-Wallis | S25, S108, S180 | 3 |
| T8 | Mann-Whitney | S25, S107, S168 | 3 |
| T9 | Descriptive Statistics and Charts | S24, S78, S118, S125, S131, S141, S154, S173, S179 | 9 |
| T10 | Levene | S162, S180 | 2 |
| T11-T18 | | Tukey (S180), Mauchly (S174), Greenhouse-Geisser (S174), Friedman (S21), Hotelling (S71), Kolmogorov-Smirnov (S72), Spearman (S25), Regression Analysis (S24) | 8 |

perience in terms of (*i*) *engagement* when the target audience is composed of students of a software engineering course; (*ii*) *recollection* when the audience involves developers understanding legacy code [5]; and (*iii*) positive *emotions* in general.

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:

> "Analyze *[Object of study]* executing in a *[Environment]* to support the *[Task]* using a *[Technique]* displayed on a *[Medium]* for the purpose of *[Purpose]* with respect to *[Quality Focus]* from the point of view of *[Perspective]* in the context of *[Context]*."

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 9: The evaluation scope of experiments in software visualizations (left-to-right): reference, object of study, task (check mark ✓ identifies tasks that were found explicit in evaluations), environment, visualization technique, medium (*i.e.*, standard computer screens SCS, immersive 3D environments I3D, and multi touch tables MTT), purpose, quality focus, perspective, context, statistical test (acronyms shown in Table 8) .

| Ref. | Object of Study | Task | Env. | Technique | Med. | Purpose | Quality Focus | Pers. | Context | Statistical Test |
|---|---|---|---|---|---|---|---|---|---|---|
| S5 | *UML diagram notation* | Identify if an UML diagram correspond to a specification | – | UML | SCS | To evaluate whether a specification matches a diagram | Correctness, Time | All | 35 CS students | T5 |
| S11 | Genisom | Search for information held within the self-organizing map. | – | City | SCS | To characterize the capability of users to extract information from a visual | Correctness, Time, Usability | All | 114 CS students | – |
| S13 | Xia | Why a particular file changed | – | Node-link | SCS | To test the initial requirements | Effectiveness, Usability | All | 5 CS students | – |
| S21 | *Spreadsheets* | Localization of faulty cells | – | Aug. source code | SCS | To gain insights on faulty cells in spreadsheets | Effectiveness, Robustness | Novice | 87 CS students | T14 |
| S23 | *Reducing Cognitive Effort* | Tasks related to distributed computations | – | Node-link; Iconic | SCS | To evaluate cognitive economy | Correctness | Novice | 20 CS students (5 female) | – |
| S24 | *DancingHamsters; MarbleSt.* | Tasks related to algorithm analysis | – | Anim. Node-link | SCS | To evaluate the impact of visualization in learning | Correctness | Novice | 12 CS students; 43 CS students | T18 |
| S25 | *Algorithm visualization* | Tasks related to the sorting algorithms | – | Aug. source code | SCS | To evaluate the impact of visualization in learning | Correctness | Novice | 157 CS students | T1,T7,T8,T17 |
| S32 | *GrowingSquares* | Is process $x$ causally related time to process $y$? ✓ | – | Node-link; Hasse | SCS | To evaluate the impact of a technique | Correctness, Efficiency, T., … | All | 12 participants (4 female) | T1 |
| S33 | PlanAni | Tasks related to sorting algorithms | Various | Aug. source code | SCS | To gain insights on supporting teaching programming in CS | Correctness | Novice | 91 CS students | – |
| S40 | *Variable dependency* | Complete an unfinished function | – | UML | SCS | To evaluate the impact of of intra-procedural variable dependencies | Correctness, Time, Usefulness | All | 38 CS students (3 female) | T1,T2 |
| S47 | *UML class diagram layout* | Match the role of a particular class | – | UML | SCS | To evaluate the impact of stereotype-based architectural UML layout | Correctness | All | 20 CS students | – |
| S50 | *Wear-based filtering* | Change the program to obtain an expected behavior ✓ | – | UML | SCS | To evaluate the impact of using wear-based filtering | Completion | All | 7 male developers | T2 |
| S61 | *Algorithm visualization* | Tasks related to algorithm analysis | – | Node-link | SCS | To evaluate the impact of using concept keyboards | Interactivity, Usefulness | Novice | 17 CS students; 18 CS students | – |
| S66 | *Social agents* | What faults did you find, and when did you find each one? | – | Iconic | SCS | To evaluate the impact of the tool | Effectiveness | Novice | 22 CS students | – |
| S71 | jGrasp | Find and correct all the non-syntactical errors | Various | Aug. source code | SCS | To gain insights on supporting teaching programming in CS | Correctness, Time | Novice | - | T15 |
| S72 | *Algorithm visualization* | What is the main difference between Prim and Dijkstra algorithms? ✓ | – | Node-link | SCS | To evaluate the impact of the narrative visualization approach | Correctness | Novice | 34 CS students | T5,T16 |
| S78 | Gilligan | Analyze the impact of changing a program. ✓ | – | Aug. source code | SCS | To evaluate the impact of a tool | Correctness | All | 6 participants | T9 |
| S96 | SolidFX | Tasks related to reverse-engineering open-source code | Windows | HEB; Pixel | SCS | To gain insights on architecture, metrics and dependencies | Performance, User Experience | All | 8 participants (ind. & acad.) | – |
| S100 | Enhance | Find dependencies between structural elements | – | Node-link | SCS | To gain insights on how devs. understand exception-handling constructs | Effectiveness | Novice | 3 CS students | – |
| S101 | saUML | Select the candidate that best describes the depicted behavior | – | UML | SCS | To evaluate the benefits of synchronization-adorned sequence diagrams | Correctness | Novice | 24 CS students | T4,T5 |
| S106 | Various | Tasks related to program comprehension and maintenance | – | Various | SCS | To evaluate the impact of a tool | Correctness | All | 90 participants (ind. & acad. | – |
| S107 | *UML Class diagram* | Identify classes to be changed to add a requirement | – | UML | SCS | To evaluate the impact of the layout | Confidence, Correctness, Time | All | 45 CS students | T1-T4,T6 |
| S108 | Version Tree vs Augur | Which classes will be directly affected by this change? ✓ | – | Node-link | SCS | To gain insights on the benefits of vis. for open source newcomers | Correctness | Novice | 27 CS students (9 females) | T2,T7 |
| S112 | *UML Class diagram* | Count abstract classes to see if proxies are distinguished | – | UML | SCS | To test the initial requirements | Effectiveness | All | 8 CS stud. & staff (2 female) | – |
| S117 | CodeMap | Find the purpose of the given application ✓ | – | Island | SCS | To gain insight on how devs. interact with vis. that are embedded in the IDE | Usability | All | 7 participants (ind. & acad.) | – |
| S118 | ProfVis | How the program can be modified to improve its performance ✓ | Java | Node-link | SCS | To analyze execution traces of Java programs | Correctness, Understanding | All | 4 participants | T9 |
| S123 | AllocRay | Find the location of a memory leak | – | Pixel | SCS | To evaluate a visualization of allocation patterns and memory problems | Correctness | All | 4 developers | – |
| S125 | SystemHotspotsView | How much bigger is the Component class than the Window class? ✓ | – | Polymetric views | SCS | To evaluate visualization rendered on a wall display | Correctness, Time | All | 11 par. (3 fem. ind. & acad.) | T9 |
| S126 | StenchBlossom | Identify code smells | Eclipse | Aug. source code | SCS | To gain insights on supporting software quality based on code smells | Confidence, Correctness | All | 12 participants (ind. & acad. | T4 |
| S127 | *Software dev. lifecycle* | Analyze the context, and roles of involved people in projects | – | Node-link | SCS | To gain insights on how devs. draw sketches and diagrams of soft. lifecycle | Effectiveness | All | 8 par. (CS stud. & resear.) | – |
| S131 | *Constellation* | Identify the location in the code of a fault | – | Node-link | SCS | To evaluate a technique for software understanding and pattern recognition | Effectiveness | All | 30 CS students | T9 |
| S135 | E-Quality | Select the most significant refactoring candidates of a program | – | Node-link; Iconic | SCS | To gain insights on visualization of design flaws and refact. opportunities | Intuitiveness | All | 16 developers | – |
| S137 | Gzoltar | Identify the location in the code of a fault | Java;Ecli. | Icicle; Treemap | SCS | To gain insights on fault localization for debugging Java progs. | Correctness, Intuit., Time, … | All | 40 CS students | T5 |
| S140 | PNLV; IMMV | What interesting visual structures do you find in the vis.? ✓ | – | Node-link | SCS | To gain insights on visualization for understanding an unknown system | Usefulness | All | 8 participants (ind. & acad.) | – |
| S141 | SourceVis | How many interfaces does this class depend on? ✓ | – | Polymetric views | MMT | To gain insights on vis. on multitouch tab. for co-located collab. in unders | Effectiveness | All | 6 par. (CS stud. &resear.) | T9 |
| S144 | SeeIT3D | Identify why the program produce a poor print quality ✓ | Eclipse | City | SCS | To gain insights on visualization for architecture of Java systems | Corr., Time, Visual Effort | All | 97 CS students | T1,T18 |
| S148 | ChronoTwigger | Investigate the software while thinking out loud | – | Node-link | I3D | To evaluate visualization of the development process and testing | Effectiveness | All | 3 developers (1 female) | – |
| S149 | regVIS | Track of the Overall Control Flow ✓ | Windows | Visual language | SCS | To gain insights on supporting assembler control-flow of regular expr. | Usability | All | 10 par. (CS stud. & resear.) | – |
| S150 | *Compiler Messages* | Identify the cause of an error by analyzing highlighted elements ✓ | – | Aug. source code | SCS | To evaluate a technique to aid devs. on comprehending error messages | Correctness, Recollection | Novice | 28 CS students | T2-T4 |
| S153 | SIFEI | Find a failure and specify a test scenario for it ✓ | Excel | Visual language | SCS | To test spreadsheets formulas | Usability, Usefulness | All | 9 participants (ind. & acad.) | – |
| S154 | TiledGrace | Describe the behavior of a program ✓ | Web | Visual language | SCS | To gain insights on supporting programming in the Grace language | Engagement, Usefulness | Novice | 33 CS students | T9 |
| S157 | CodeSurveyor | Rank the code maps that best represent the codebase ✓ | – | Island | SCS | To evaluate the support of code maps in learning and navigating systems | Effectiveness | All | 5 developers (1 female) | – |
| S162 | ExplorViz | What is the purpose of the WWWPRINT application in your opinion? ✓ | Web | City | I3D | To evaluate an architecture based on metric analysis | Correctness, Time | All | 25 CS students | T5,T6,T10 |
| S164 | VIMETRIK | What is the number of compilation units in the Tomcat system? ✓ | – | City; Matrix | SCS | To evaluate metrics and vis. of a software system according to requirements | Correctness, Intuit., Time, … | All | 21 CS students | T1,T4,T9 |
| S166 | *Working Sets* | Analyze the developer activity on entities of the working sets | – | Node-link | SCS | To gain insights on visualization of the evolution of working sets | Effectiveness | All | 14 developers | – |
| S168 | CuboidMatrix | Identify identical interactions, along time, between classes? ✓ | – | Space-time cube | SCS | To evaluate the impact of the tool in software comprehension | Correctness | All | 8 par. (CS stud. & resear.) | T8 |
| S169 | Perquimans | Which sheets contained the most dangerous formula practice ✓ | – | Node-link | SCS | To gain insights of vis. tool's support of exploration, and quantification | Applicability | All | 4 CS students | – |
| S173 | *Indented Hierarchy* | Find the most symmetric subtree in the tree. ✓ | – | Pixel | SCS | To evaluate the impact of a technique compared with node-link diagrams | Correctness, Readability, Time | All | 18 vis. experts (3 female) | T9 |
| S174 | *Regex textual vs graphical* | Is *ABC* in the language defined by a regular expression? ✓ | – | Aug. source code | SCS | To test the impact of a graphical notation on the readability of a regex | Correctness, Readability, Time | Novice | 22 par. (CS stud. and staff) | T1,T12,T13 |
| S177 | Code Park | Identify where in the code add the logic to support a feature ✓ | – | City | I3D | To evaluate the impact of the tool on usability and engagement | Ease-to-use, Engagement, Time | Novice | 28 CS students (6 female) | T1,T6 |
| S179 | iTraceVis | Where did the dev. not look at with respect to the same method? ✓ | Eclipse | Heatmap | SCS | To gain insights on analyzing eye movement data of code reading | Correctness | All | 10 CS students | T9 |
| S180 | CityVR | Locate the best candidate for the god class smell ✓ | Pharo; U. | City | SCS | To gain insights on visualization for architecture based on metrics in OOP | Correctness, Recollection, T., … | All | 21 participants | T1,T6,T7,T10,T11 |
| S181 | MethodExecutionReports | Tasks related to execution report for profiling and debugging ✓ | Java | Charts | SCS | To gain insights on supporting summarization of methods execution | Understandability, Usefulness | All | 11 participants (ind. & acad. | – |

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: (*1*) 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 (*1*) 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

[1] Alves, V., Niu, N., Alves, C., Valença, G., 2010. Requirements engineering for software product lines: A systematic literature review. Information and Software Technology 52 (8), 806–820.

[2] Amar, R., Stasko, J., 2004. A knowledge task-based framework for design and evaluation of information visualizations. In: Proc. of INFOVIS. IEEE, pp. 143–150.

[3] Bassil, S., Keller, R., 2001. Software visualization tools: survey and analysis. In: Proc. of IWPC. pp. 7 –17.

[4] Bertini, E., Tatu, A., Keim, D., 2011. Quality metrics in high-dimensional data visualization: An overview and systematization. Transactions on Visualization and Computer Graphics 17 (12), 2203–2212.

[5] Bloom, B. S., et al., 1956. Taxonomy of educational objectives. vol. 1: Cognitive domain. New York: McKay, 20–24.

[6] Elmqvist, N., Yi, J. S., 2015. Patterns for visualization evaluation. Proc. of INFOVIS 14 (3), 250–269.

[7] Fink, A., 2003. The survey handbook. Vol. 1. Sage.

[8] Forsell, C., 2010. A guide to scientific evaluation in information visualization. In: Proc. of IV. IEEE, pp. 162–169.

[9] Greene, G. J., Esterhuizen, M., Fischer, B., 2017. Visualizing and exploring software version control repositories using interactive tag clouds over formal concept lattices. Information and Software Technology 87, 223–241.

[10] Hornbæk, K., Høegh, R. T., Pedersen, M. B., Stage, J., 2007. Use case evaluation (UCE): A method for early usability evaluation in software development. In: Proc. of IFIP. Springer, pp. 578–591.

[11] Höst, M., Regnell, B., Wohlin, C., 2000. Using students as subjects — a comparative study of students and professionals in lead-time impact assessment. Empirical Software Engineering 5, 201–214.
[12] Isenberg, T., Isenberg, P., Chen, J., Sedlmair, M., Möller, T., 2013. A systematic review on the practice of evaluating visualization. Transactions on Visualization and Computer Graphics 19 (12), 2818–2827.
[13] Kienle, H. M., Müller, H. A., 2010. The tools perspective on software reverse engineering: requirements, construction, and evaluation. In: Advances in Computers. Vol. 79. Elsevier, pp. 189–290.
[14] Kitchenham, B. A., Pfleeger, S. L., Pickard, L. M., Jones, P. W., Hoaglin, D. C., Emam, K. E., Rosenberg, J., 2002. Preliminary guidelines for empirical research in software engineering. IEEE Trans. Softw. Eng. 22 (8), 721–734.
[15] Lam, H., Bertini, E., Isenberg, P., Plaisant, C., Carpendale, S., 2012. Empirical studies in information visualization: Seven scenarios. Transactions on Visualization and Computer Graphics 18 (9), 1520–1536.
[16] Lopez-Herrejon, R. E., Illescas, S., Egyed, A., 2018. A systematic mapping study of information visualization for software product line engineering. Journal of Software: Evolution and Process 30 (2).
[17] Mackinlay, J., 1986. Automating the design of graphical presentations of relational information. Transactions On Graphics 5 (2), 110–141.
[18] Maletic, J. I., Marcus, A., 2003. CFB: A call for benchmarks-for software visualization. In: Proc. of VISSOFT. Citeseer, pp. 113–116.
[19] Mattila, A.-L., Ihantola, P., Kilamo, T., Luoto, A., Nurminen, M., Väätäjä, H., 2016. Software visualization today: systematic literature review. In: Proc. of International Academic Mindtrek Conference. ACM, pp. 262–271.
[20] Merino, L., Bergel, A., Nierstrasz, O., 2018. Overcoming issues of 3D software visualization through immersive augmented reality. In: Proc. of VISSOFT. IEEE, p. in review.
[21] Merino, L., Fuchs, J., Blumenschein, M., Anslow, C., Ghafari, M., Nierstrasz, O., Behrisch, M., Keim, D., 2017. On the impact of the medium in the effectiveness of 3D software visualization. In: Proc. of VISSOFT. IEEE, pp. 11–21.
URL http://scg.unibe.ch/archive/papers/Meri17b.pdf
[22] Merino, L., Ghafari, M., Anslow, C., Nierstrasz, O., 2017. CityVR: Gameful software visualization. In: Proc. of ICSME. IEEE, pp. 633–637.
URL http://scg.unibe.ch/archive/papers/Meri17c.pdf
[23] Merino, L., Ghafari, M., Nierstrasz, O., 2016. Towards actionable visualisation in software development. In: Proc. of VISSOFT. IEEE.
URL http://scg.unibe.ch/archive/papers/Meri16a.pdf
[24] Merino, L., Ghafari, M., Nierstrasz, O., Bergel, A., Kubelka, J., 2016. MetaVis: Exploring actionable visualization. In: Proc. of VISSOFT. IEEE.
URL http://scg.unibe.ch/archive/papers/Meri16c.pdf
[25] Merino, L., Lungu, M., Nierstrasz, O., 2015. Explora: A visualisation tool for metric analysis of software corpora. In: Proc. of VISSOFT. IEEE, pp. 195–199.
URL http://scg.unibe.ch/archive/papers/Meri15b.pdf
[26] Merino, L., Seliner, D., Ghafari, M., Nierstrasz, O., 2016. CommunityExplorer: A framework for visualizing collaboration networks. In: Proc. of IWST. pp. 2:1–2:9.
URL http://scg.unibe.ch/archive/papers/Meri16b.pdf
[27] Müller, R., Kovacs, P., Schilbach, J., Eisenecker, U. W., Zeckzer, D., Scheuermann, G., 2014. A structured approach for conducting a series of controlled experiments in software visualization. In: Proc. of IVAPP. IEEE, pp. 204–209.
[28] Munzner, T., 2008. Process and pitfalls in writing information visualization research papers. In: Information visualization. Springer, pp. 134–153.
[29] Munzner, T., 2014. Visualization analysis and design. CRC press.
[30] Novais, R. L., Torres, A., Mendes, T. S., Mendonça, M., Zazworka, N., 2013. Software evolution visualization: A systematic mapping study. Information and Software Technology 55 (11), 1860–1883.
[31] Panas, T., Epperly, T., Quinlan, D., Sæbjørnsen, A., Vuduc, R., 2016. Comprehending software architecture using a unified single-view visualization. In: Antonakos, J. L. (Ed.), Data Structure and Software Engineering: Challenges and Improvements. CRC Press, Ch. 2, pp. "22–41".
[32] Razali, N. M., Wah, Y. B., et al., 2011. Power comparisons of shapiro-wilk, kolmogorov-smirnov, lilliefors and anderson-darling tests. Journal of Statistical Modeling and Analytics 2 (1), 21–33.
[33] Runeson, P., Höst, M., 2009. Guidelines for conducting and reporting case study research in software engineering. Empirical Software Engineering 14 (2), 131.
[34] Schots, M., Vasconcelos, R., Werner, C., 2014. A quasi-systematic review on software visualization approaches for software reuse. Technical report.
[35] Schots, M., Werner, C., 2014. Using a task-oriented framework to characterize visualization approaches. In: Proc. of VISSOFT. IEEE, pp. 70–74.
[36] Sensalire, M., Ogao, P., Telea, A., 2008. Classifying desirable features of software visualization tools for corrective maintenance. In: Proc. of SOFTVIS. ACM, pp. 87–90.
[37] Sensalire, M., Ogao, P., Telea, A., 2009. Evaluation of software visualization tools: Lessons learned. In: Proc. of VISSOFT. IEEE, pp. 19–26.
[38] Seriai, A., Benomar, O., Cerat, B., Sahraoui, H., Sep. 2014. Validation of software visualization tools: A systematic mapping study. In: Proc. of VISSOFT. pp. 60–69.
[39] Shahin, M., Liang, P., Babar, M. A., 2014. A systematic review of software architecture visualization techniques. Journal of Systems and Software 94, 161–185.
[40] Sjøberg, D. I., Hannay, J. E., Hansen, O., Kampenes, V. B., Karahasanovic, A., Liborg, N.-K., Rekdal, A. C., 2005. A survey of controlled experiments in software engineering. Transactions on Software Engineering 31 (9), 733–753.
[41] Storey, M.-A. D., Čubranić, D., German, D. M., 2005. On the use of visualization to support awareness of human activities in software development: a survey and a framework. In: Proc. of SOFTVIS. ACM Press, pp. 193–202.
URL http://portal.acm.org/citation.cfm?id=1056018.1056045
[42] Van Wijk, J. J., 2006. Views on visualization. Transactions on Visualization and Computer Graphics 12 (4), 421–432.
[43] Wobbrock, J. O., Findlater, L., Gergle, D., Higgins, J. J., 2011. The aligned rank transform for nonparametric factorial analyses using only anova procedures. In: Proc. of SIGCHI. ACM, pp. 143–146.
[44] Wohlin, C., Runeson, P., Höst, M., Ohlsson, M. C., Regnell, B., Wesslén, A., 2000. Experimentation in Software Engineering. Kluwer Academic Publishers.
[45] Wohlin, C., Runeson, P., Höst, M., Ohlsson, M. C., Regnell, B., Wesslén, A., 2012. Experimentation in software engineering. Springer Science & Business Media.
[46] Yin, R. K., 2013. Case study research: Design and methods. Sage publications.
[47] Young, P., Munro, M., 1998. Visualising software in virtual reality. In: Proc. of IWPC. IEEE, pp. 19–26.
[48] Zelkowitz, M. V., Wallace, D. R., 1998. Experimental models for validating technology. Computer 31 (5), 23–31.