DOI 10.1109/ICPC.2017.9

Comprehending Studies on Program Comprehension

***Abstract*—Program comprehension is an important aspect of developing and maintaining software, as programmers spend most of their time comprehending source code. Thus, it is the focus of many studies and experiments to evaluate approaches and techniques that aim to improve program comprehension. As the amount of corresponding work increases, the question arises how researchers address program comprehension. To answer this question, we conducted a literature review of papers published at the International Conference on Program Comprehension, the major venue for research on program comprehension. In this article, we *i)* present preliminary results of the literature review and *ii*) derive further research directions. The results indicate the necessity for a more detailed analysis of program comprehension and empirical research.**

***Keywords*-Systematic Review, Study Comprehension, Empirical Research**

I. Introduction

Understanding how a program works is essential for software engineers. Consequently, numerous approaches to improve or measure program comprehension exist [15, 18]. To evaluate these approaches, empirical studies with human participants are essential. Such studies are becoming standard in software engineering [19], and several guidelines for conducting and reporting exist (e.g., by Kitchenham et al. [12], Jedlitschka and Pfahl [8], or Ko et al. [14]). However, Maalej et al. [15] report a gap between research on program comprehension and its application in industry. The authors found that developers tend to select comprehension strategies depending on the context of their current work. Therefore, it is problematic to compare and replicate studies on program comprehension.

However, an empirical study provides only benefits if researchers can replicate it to confirm or contest the results. Replicating studies is essential to construct and consolidate empirical knowledge in a research area [1, 3, 10, 19]. To this end, comprehensive descriptions are necessary [19], but these often miss important details [7, 20]. For that purpose, Carver [6] proposes guidelines to report replications and discusses several information of an original study that should be provided, for example, participants and design. Still, there are differences regarding which and how authors report their data. This can hamper researchers and practitioners in comprehending, replicating, and comparing studies. Thus, several questions arise, for example:

How do researchers investigate program comprehension?
How did the evaluation of program comprehension evolve?
How did the quality of studies and documentation evolve?

In this paper, we make the first step towards answering these questions and assess the quality of empirical studies on program comprehension at the *International Conference on Program Comprehension* (ICPC), the major venue for research on program comprehension. Hence, the published articles should contain studies on program comprehension. Our goal is to overview the evolution of such studies over time and how clearly authors specify researched topics and evaluations. To this end, we conducted a manual literature review [11] on articles published at ICPC from 2006 until 2016. We analyzed 540 available papers to determine which details their authors report on *context*, *terminology*, and potential *threats to validity*. The findings allow us to gain first insights into past development of research on program comprehension and empirical research in general. Based on the preliminary results, we derive directions for further investigations to assess the aforementioned questions in more detail.

II. Research Method

In this section, we describe our literature review, which we based on the guidelines by Kitchenham and Charters [11].

A. Research Questions

In this paper, we address the following research questions:

RQ-1 **Which part of program comprehension do researchers investigate?**
Program comprehension is related to several aspects of software development, for instance, bug fixing or implementation. We provide an initial set of categories that can be used to categorize corresponding research (i.e., source code, program behavior, testing, API, requirements, documentation, and miscellaneous).

RQ-2 **Which terminology is used to report evaluations?**
By answering this research question, we aim to identify and overview of commonly applied methods to evaluate program comprehension. We found no common terminology to describe evaluations (e.g., empirical study, exploratory study, or user study), which may lead to potential misunderstandings (e.g., exploratory study as opposed to exploratory case study). Hence, it seems necessary to clearly define evaluation approaches and the corresponding terminology.

RQ-3 **Do authors report threats to validity?**
Threats to validity describe potential biases and are essential to understand the quality of empirical studies [19]. Hence, we analyzed how many papers explicitly discussed them, for instance, in an own section or paragraph. This serves as first indicator for the quality of a study, but we plan to refine this in a future phase of the literature review.

By answering these questions, we provide a preliminary overview of research on program comprehension. The main goal of this paper is to illustrate potential problems and discuss directions for future work.

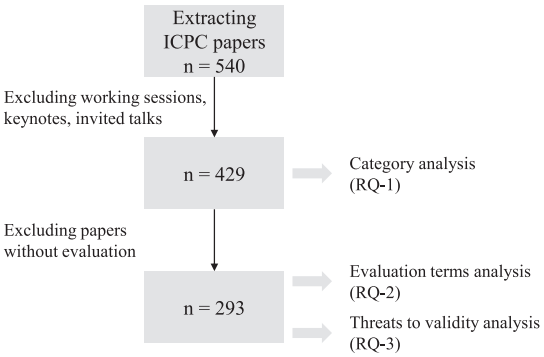

Fig. 1. Steps of identifying relevant articles.

B. Search Process

We selected ICPC as the major venue for research on program comprehension to perform a manual search [22]. From these, we considered only the years since the conference emerged from its corresponding workshop. Conferences are assumed to have higher-quality standards and this switch indicates the increasing importance of the venue, and topics. Hence, our initial sample includes all 11 occasions from 2006 to 2016, for which 540 papers are available. We give an overview of our selection process in Figure 1.

To this end, we used the following criteria sample the papers and collect data:

- To answer our first research question, we determined the investigated part of comprehension. To this end, we excluded articles that do not describe new research, for example, invited talks or working sessions. As we show in Figure 1, 429 papers remained.
- To answer our second research question, we extracted from each paper which terminology authors use to report their evaluations (e.g., empirical study, exploratory study, or experiment).
- To answer our third research question, we identified whether threats to validity are reported.

For the second and third research questions, we excluded articles that do not present an evaluation, which was often omitted in tool demos or short papers. From the 429 articles of the previous step, 293 remained.

III. Results

In this section, we present and discuss the results of our literature review.

RQ-1: Which Part of Program Comprehension do Researchers Investigate?

Results: In Table I, we present and define parts of program comprehension that were the focus of the reviewed articles. We remark that we categorized each article into one category even if it addressed cross-cutting research. We decided to use this categorization to provide an initial overview. In future research, we intend to refine these categories and investigate them in more detail. Under miscellaneous, we summarize all approaches we found not to fit into a category and which were rarely reported. For example, Wang et al. [21] investigate fault diagnosis for automated configurations based on expert knowledge.

TABLE I
Categories of comprehension at ICPC.

Category	Definition	# Articles
Source code	Research on comprehending the source code of a program.	229
Program behavior	Research on comprehending the behavior and architecture of a program based on its execution.	104
Testing	Research on identifying, managing, and comprehending bugs to test and maintain a program.	38
API	Research on the usage and comprehensibility of APIs and their interfaces.	21
Requirements	Research on comprehending requirements and their mapping them towards a program.	16
Documentation	Research on documenting and specifying a program.	6
Miscellaneous	Research on other parts of comprehension with few articles.	15

Discussion: The results in Table I illustrate that research at ICPC focuses on source code and program behavior. This is not surprising, as ICPC is the premier venue for program comprehension and both aspects are essential in this regard. Still, there are some categories that occur occasionally, for instance, testing, API, or requirements comprehension. During our analysis, we found it challenging to categorize such articles. For instance, Jiang et al. [9] address documentation, but in the context of APIs. To assign a single category to a paper, we selected the type of comprehension we identified to be the overarching aspect of comprehension under research (i.e., API for Jiang et al. [9]). The lack of clarity on which part of program comprehension is under research may lead to problems while comprehending papers or identifying related work.

To answer our first research question, we find that *source code* and *program behavior* are the mostly addressed parts of program comprehension at ICPC. While other categories have also been the focus of research papers, they never gained the same amount of attention. Still, we see that the articles cover a broad range of topics and all phases of software development. To get a better overview of how different parts of program comprehension are understood, we intend to look into them in more detail and find definitions that are easier to assign.

RQ-2: Which Terminology is Used to Report Evaluations?

Results: We identified numerous different terms that authors used to describe their evaluation. The terms we present in Table II appeared at least 10 times, for instance, *study*. Combinations with their corresponding refinements (prefixes) appeared 5 times at minimum (e.g., *case study*). Sparsely, authors applied additional refinements (e.g., *empirical case study*) to describe their evaluation.

Discussion: Case studies, empirical studies, and experiments are common descriptions. However, we found several

TABLE II
EXAMPLES OF USED TERMS TO DESCRIBE THE EVALUATION APPROACH.

Additional Prefix	Prefix	#	Term	#
Empirical, exploratory	Case	80	Study	171
-	Empirical	34		
Qualitative	Exploratory	15		
Exploratory, quantitative	User	12		
-	Controlled	18	Experiment	83
-	Within-subject	5		
-	Empirical	6	Evaluation	19
-	-	-	Survey	10
-	-	-	Analysis	10

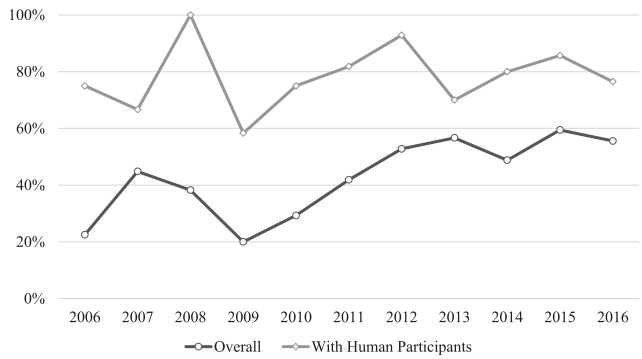

Fig. 2. Ratio of papers that include threats to validity for empirical studies.

papers that name rather unique evaluation methods, such as *in-depth qualitative observation* [5]. The authors conducted a survey with a quantitative study on variable declarations in Java projects and manually analyzed a subset of their results. Additional prefixes are used to refine the used terms. These prefixes become rather similar and the authors do not clarify if there are defined differences or are synonymous. For instance, we argue that it is problematic to separate between *exploratory case studies* [17], *exploratory user studies* [2], and *exploratory studies* [16], because in all these examples, authors observed the behavior of users during different tasks (i.e., program comprehension [17], maintenace [2], and programming [16]). While the tasks and approaches of these papers differ, the general evaluation method is the same, indicating synonymous and ambiguous use of the terminology. This may mislead researchers to identify which type of evaluation is actually applied (i.e. a user study or case study), as they may expect a different one and need to analyze details.

To answer our second research question, we find that researchers use a diverse and potentially misleading terminology to report their evaluation. This might be a threat to the comprehensibility of empirical studies and also reduces the possibility for comparable replications, because it can be unclear which evaluation was actually applied. By consolidating and extending existing catalogs of evaluation methods [8, 12], researchers have guidance on terminology, which can avoid misunderstandings.

RQ-3: Do Authors Report Threats to Validity?

Results: We counted how often threats to validity were reported for all evaluations and also separately for studies with human participants. The results, which we illustrate in Figure 2, indicate a paradigm change at ICPC for the year 2009, in which the highest number of articles was accepted (55), but approximately 44% (24) of them did not provide an evaluation. In addition, only approximately 60% of the studies with participants and 20% overall reported threats to validity.

Discussion: Since 2009, the situation changed considerably. Especially in recent years, most accepted papers reported an empirical evaluation and threats to validity. For example, in 2016, 25 (out of 45) papers report both, an evaluation and threats to validity, while only 10 papers consider neither. While we found no clear trend for studies with participants, the overall ratio of articles discussing threats to their studies increased considerably from approximately 20% in 2009 to 57% in 2016. This indicates an increasing quality of papers at ICPC and growing awareness of the community to report limitations of their studies, which are results similar to those of Siegmund et al. [19] for other venues. While this development indicates an increasing quality, further in-depth analyses are necessary to assess details of the papers.

To answer our third research question, the number of empirical studies for which the authors report threats to validity increases in recent years. We see strong points that these become more important at ICPC. Hence, it seems that researchers consider the corresponding discussion to be important.

IV. THREATS TO VALIDITY

Before discussing threats to validity, we emphasize that the literature review and results presented in this work are preliminary. Hence, we described only parts of our review, currently preventing replication. We aim to extend our analysis and provide a detailed report in future work.

A threat to internal validity is that we deviated from the guidelines of Kitchenham and Charters [11], as only the first author manually searched a single venue [4, 13]. To mitigate this threat we especially focused on ICPC as the premier venue for program comprehension. In addition, we considered papers since 2006, when the conference emerged from its corresponding workshop, assuming a higher quality. Hence, the number of articles was limited and other authors double checked the results, limiting potential biases on our preliminary analysis.

A threat to construct validity is that we found it challenging to correctly interpret all articles mainly for two reasons: First, among different but also within the same papers, a different terminology is used to describe evaluations. As we emphasize, this hampers the understanding of which approach is actually used. Second, cross-cutting research made it problematic to unambiguously assign a paper to a category. Due to these points, other researchers may decide differently during their analysis. However, our findings provide an initial point for further research and highlight the problems of comprehending studies.

V. Conclusion and Future Work

In this paper, we present initial results of a literature review at the International Conference on Program Comprehension to understand how program comprehension is addressed and evaluated in papers. Overall, the results show that:

- Corresponding to ICPC's focus, most papers address the topics *source code* and *program behavior*.
- Ambiguous terminology is used to report studies, which makes it difficult to compare studies.
- Based on our initial assessment the quality of papers increased in recent years. However, we need to extend the definition of quality.

In future work, we aim to deepen the initial analysis presented in this paper. To this end, we will extend our scope, for example, by including further venues and assessing the papers in more detail. We also aim to evaluate connections between the different topics of program comprehension, such as how documentation is used to comprehend API usage [9]. In a more long-term perspective, we aim at a catalog for empirical studies, refining existing ones [8, 12] with concrete guidelines, for example, on identifying and reporting threats to validity.

References

[1] V. R. Basili, F. Shull, and F. Lanubile. Building Knowledge Through Families of Experiments. *IEEE Trans. Software Eng.*, 25(4):456–473, 1999.

[2] F. Beck, O. Moseler, S. Diehl, and G. D. Rey. In Situ Understanding of Performance Bottlenecks through Visually Augmented Code. In *ICPC*, pages 63–72. IEEE, 2013.

[3] R. M. M. Bezerra, F. Q. B. da Silva, A. M. Santana, C. V. C. Magalhaes, and R. E. S. Santos. Replication of Empirical Studies in Software Engineering: An Update of a Systematic Mapping Study. In *ESEM*, pages 1–4. IEEE, 2015.

[4] O. P. Brereton, B. A. Kitchenham, D. Budgen, M. Turner, and M. Khalil. Lessons from Applying the Systematic Literature Review Process within the Software Engineering Domain. *J. Syst. Software*, 80(4):571–583, 2007.

[5] S. Butler, M. Wermelinger, and Y. Yu. A Survey of the Forms of Java Reference Names. In *ICPC*, pages 196–206. IEEE, 2015.

[6] J. C. Carver. Towards Reporting Guidelines for Experimental Replications: A Proposal. In *RESE*. 2010.

[7] T. Dybå, V. B. Kampenes, and D. I. K. Sjøberg. A Systematic Review of Statistical Power in Software Engineering Experiments. *Inform. Software Tech.*, 48(8):745–755, 2006.

[8] A. Jedlitschka and D. Pfahl. Reporting Guidelines for Controlled Experiments in Software Engineering. In *ISESE*, pages 95–104. IEEE, 2005.

[9] J. Jiang, J. Koskinen, A. Ruokonen, and T. Systa. Constructing Usage Scenarios for API Redocumentation. In *ICPC*, pages 259–264. IEEE, 2007.

[10] N. Juristo and S. Vegas. Using Differences Among Replications of Software Engineering Experiments to Gain Knowledge. In *ESEM*, pages 356–366. IEEE, 2009.

[11] B. A. Kitchenham and S. Charters. Guidelines for Performing Systematic Literature Reviews in Software Engineering. Technical Report EBSE-2007-01, Keele University and University of Durham, 2007.

[12] B. A. Kitchenham, S. L. Pfleeger, L. M. Pickard, P. W. Jones, D. C. Hoaglin, K. El Emam, and J. Rosenberg. Preliminary Guidelines for Empirical Research in Software Engineering. *IEEE Trans. Software Eng.*, 28(8):721–734, 2002.

[13] B. A. Kitchenham, O. P. Brereton, D. Budgen, M. Turner, J. Bailey, and S. Linkman. Systematic Literature Reviews in Software Engineering – A Systematic Literature Review. *Inform. Software Tech.*, 51(1):7–15, 2009.

[14] A. J. Ko, T. D. Latoza, and M. M. Burnett. A Practical Guide to Controlled Experiments of Software Engineering Tools with Human Participants. *Empir. Software Eng.*, 20(1):110–141, 2015.

[15] W. Maalej, R. Tiarks, T. Roehm, and R. Koschke. On the Comprehension of Program Comprehension. *ACM Trans. Software Eng. Methodol.*, 23(4):31:1–31:37, 2014.

[16] P. Petersen, S. Hanenberg, and R. Robbes. An Empirical Comparison of Static and Dynamic Type Systems on API Usage in the Presence of an IDE: Java vs. Groovy with Eclipse. In *ICPC*, pages 212–222. ACM, 2014.

[17] T. Roehm. Two User Perspectives in Program Comprehension: End Users and Developer Users. In *ICPC '15*, pages 129–139. IEEE, 2015.

[18] J. Siegmund. Program Comprehension: Past, Present, and Future. In *SANER*, pages 13–20. IEEE, 2016.

[19] J. Siegmund, N. Siegmund, and S. Apel. Views on Internal and External Validity in Empirical Software Engineering. In *ICSE*, pages 9–19. IEEE, 2015.

[20] D. I. K. Sjøberg, B. Anda, E. Arisholm, T. Dyba, M. Jørgensen, A. Karahasanovic, E. F. Koren, and M. Vokác. Conducting Realistic Experiments in Software Engineering. In *ISESE*, pages 17–26. IEEE, 2002.

[21] M. Wang, X. Shi, and K. Wong. Capturing Expert Knowledge for Automated Configuration Fault Diagnosis. In *ICPC*, pages 205–208. IEEE, 2011.

[22] H. Zhang, M. A. Babar, and P. Tell. Identifying Relevant Studies in Software Engineering. *Inform. Software Tech.*, 53(6):625–637, 2011.