Claims and Beliefs about Code Clones: Do We Agree as a Community?

A Survey

Debarshi Chatterji, Jeffrey C. Carver, Nicholas A. Kraft
Department of Computer Science
University of Alabama
Tuscaloosa, AL, USA
dchatterji@ua.edu; {carver, nkraft}@cs.ua.edu

***Abstract*—Research on code clones and their impact on software development has been increasing in recent years. There are a number of potentially competing claims among members of the community. There is currently not enough empirical evidence to provide concrete information about these claims. This paper presents the results of a survey of members of the code clone community. The goal of the survey was to determine the level of agreement of community members regarding some key topics. While the results showed a good bit of agreement, there was no universal consensus on all topics. Survey respondents were not in complete agreement about the definitions of Type III and Type IV clones. The survey respondents were more uncertain about how developers behave when working with clones. From the survey it is clear that there are areas where more empirical research is needed to better understand how to effectively work with clones.**

***Keywords*-survey; code clones; clone evolution; clone management; software maintenance; developer behavior.**

I. INTRODUCTION

The community of code clone researchers has become well-established as evidenced by the success of International Workshop on Software Clones (IWSC) for example. As a community becomes more successful and diverse, there is the potential for divergence among members of the community with regards to their opinions about the current state of knowledge, the most important research goals, and the roadmap for future research. We believe that the code clone community is quickly approaching this point, if it is not already there. Our goal in this paper was to gather information from the code clone community to identify and document areas of agreement and disagreement, with any eye towards initiating a discussion about future research directions. To provide a bit of background, the remainder of this section illustrates some of the areas in which there is a need to gather information from across the community.

Before describing the research motivation in detail, we provide a brief description of our research methodology. The best way to quickly gather the opinions of a large group of distributed experts is via survey [8]. In our case, to begin to understand and document the level of agreement, we created and distributed an online survey to gather information from members of the code clone community regarding some of the most important topics as described in Section II. The details of the survey design are provided in Sections III and IV.

The rapid progression of code clone research has resulted in numerous techniques and tools for clone detection, management, and visualization. In addition, researchers have conducted a number of empirical studies designed to validate the various properties and characteristics of those tools and techniques. However, in many of these cases, researchers have made different assumptions and used different definitions when designing and reporting their studies. As a result, it is not always obvious how to compare results across studies.

The remainder of the paper is organized as follows. Section II provides some background on code clone research which leads to the research hypothesis. Section III describes the design of the survey. Section IV describes the pilot study of the survey. Section V discussion the demographics of the survey respondents. Section VI analyzes and discusses the results of the survey. Finally Sections VII, VIII and IX suggest future work, identify the threats to validity of the study and draw conclusions respectively.

II. RESEARCH GOALS

In our own survey of the literature and impressions from interacting with the community, we identified three areas in which we believed that a survey could make a contribution to the code clone community. The following three subsections describe each of these research thrusts in more detail along with some background literature that motivated their inclusion in the survey.

A. Research Thrust 1: General Clone Usage Information

In this thrust, we focus on the definitions of clone types and high-level uses of clone information. In order to move a community forward, there is a need for a set of concrete and agreed upon definitions of key terms. While it is a widely accepted belief that there are different types of clones that impact software differently, it is not clear whether there are consistent, agreed upon definitions. Some researchers defined different types of code clones depending upon the level of similarity of the code fragments [1, 2, 6]. However, the similarity and size thresholds for determining that code fragments are clones is not clearly defined. A common classification of clone types is as follows [10]:

- "Type I: Code fragments are identical except for variations in whitespace, layout, and comments.
- Type II: Code fragments are structurally and syntactically identical except for variations in identifiers, literals, types, layout and comments.
- Type III: Code fragments are copies with further modifications. Statements can be changed, added or removed in addition to variations in identifiers, literals, types, layout and comments.
- Type IV: Two or more code fragments perform the same computation but are implemented through different syntactic variants."

In our survey, we wanted to understand the general acceptance of these definitions.

A second general topic was the ratio of cloned code to non-cloned code. The current belief is that code clones are not necessarily harmful. However, developers do need to track them. Therefore, we can hypothesize that code clones are important for system quality. To better understand this point, we asked respondents about the effect of the clone ratio on code quality.

B. Research Thrust 2: Clones and Developer Behavior

It is a widely accepted fact that code clones impact software maintenance. While there have been some studies focused on understanding developer behavior during maintenance tasks [4, 7], there is not enough evidence to draw any general conclusions about management of clones or use of clone-aware tools.

Our expectation from this survey was to determine whether there was any disagreement or confusion within the community regarding claims and beliefs about developer behavior. In practice, developers have some expectations from their tools, but there is currently not a body of empirical evidence to suggest exactly what these expectations should be. Such evidence would require a number of empirical studies. The goal of this survey was not to prove any claims, but rather to lay a foundation for constructive discussion at the workshop and suggest directions for future work.

C. Research Thrust 3: Clone Evolution

A newer research direction within the clone community focuses on how cloned code evolves over time. As this code changes, it exhibits various patterns and characteristics. An analysis of clone evolution can reveal, for example, which clones are change-prone and which clones are long-lived [9]. Developers can use this information to better manage clones. In the survey, we wanted to understand the current beliefs about clones and their evolution.

III. Survey Design

Based on the three research thrusts described in the previous section, we designed a 30-question survey with three distinct sections, including some demographic questions. The survey questions are provided in Sections V and VI along with the analysis of the results from each question. To reduce the time burden on the survey respondents, we included as many multiple choice questions as possible. The survey contained 14 multiple choice questions along with an optional field to explain the given answer. In a survey like this one, where we are trying to gather the current beliefs in the community, it is not possible to make all questions multiple choice. So, the survey contained 8 short answer qualitative questions. In addition to those, there were 8 objective questions, which required selection from choices or a one word answer.

Out of the 30 questions we had to exclude six questions because there were not enough responses to provide any useful insight. The remainder of this paper discusses the results of the remaining 24 questions. (Note that we renumbered the questions in the paper for simplicity.)

Using the list of papers posted on Dr. Robert Tairas' website[1], we generated a list of experts in the code clone community to serve as the audience for our survey. We identified 71 people to include on the distribution list for the survey. We sent the initial email out in the third week of November, 2011. After two reminders, we closed the survey in the second week of January, 2012.

IV. Pilot

Before distributing the survey to the mailing list, we conducted a pilot study to debug and improve it. Three local researchers with knowledge of code clone research participated in the pilot. We used a two-phase pilot study. First, two of our pilot participants took an initial version of the survey and suggested improvements. The comments primarily concerned the wording of some questions and the length of the survey. After making these changes, we had a third pilot participant take the survey. The only change suggested by this third pilot participant was to add a 'not familiar' option to the multiple choice questions. After this change, we considered the survey complete.

V. Respondent Demographics

This section discusses the demographics for the 22 survey responses we received. To get a sense of the interests of the survey respondents, we asked them to select their primary and secondary research areas related to clones. Figure 1 shows the distribution of the primary and secondary research interests among the five choices provided. We came up with these choices based on our understanding of the code clone area. One interesting observation from Figure 1 is that the code clone research seems to be heavily focused on clone detection research, with little focus on clone visualization.

To get a more detailed understanding of the specific topics that survey respondents were familiar with, we asked them to indicate which of the following topics they were familiar with:

A. Causes and effects of clones;
B. Effect of clones on system complexity and quality;

[1] http://students.cis.uab.edu/tairasr/clones/literature/

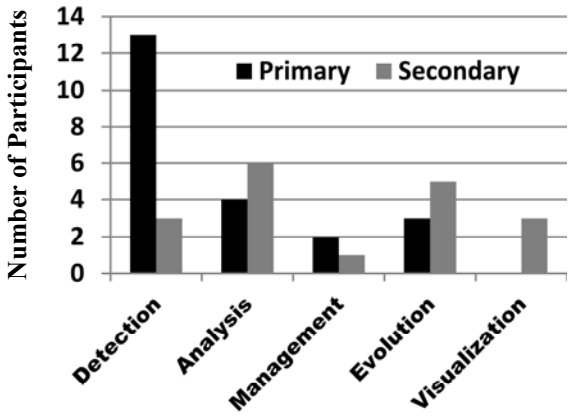


Figure 1: Primary and Secondary Areas of Research

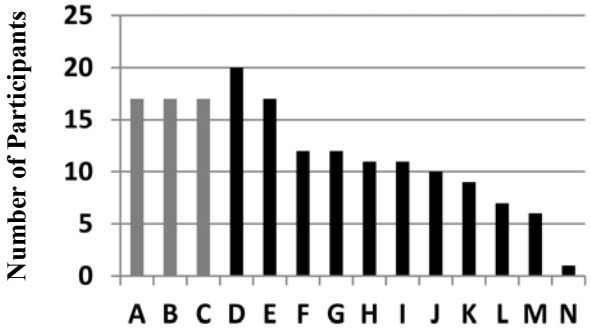


Figure 2: Familiarity distribution

C. Applications of clone analysis;
D. Tools and systems for detecting and analyzing software clones;
E. Techniques and algorithms for clone detection, analysis, and management;
F. Clone and clone pattern visualization;
G. Clone evolution and variation;
H. Evaluation and benchmarking of clone detection methods;
I. Role of clones in software system evolution;
J. Clone management;
K. Clone analysis in families of similar systems;
L. Refactoring through clone analysis;
M. Clone-aware software design and development;
N. Others.

These categories were obtained from the call for papers of IWSC 2011. We allowed each respondent to select as many topics as they wanted to. Figure 2 shows the results. One thing we noticed about the list of topics was that, the first three topics were more general than the other topics. Therefore, we expected all respondents to select one or more of the first three items. The results showed that all but one respondent selected at least one of those three, with just over 50% selecting all three. Consistent with the response to the first question, 80% of the respondents selected both D and E which are related to code clone detection tools and techniques.

The remaining demographic questions (shown in Table 1) help to characterize the respondents. This characterization provides some context for the analysis discussed in Section VI. From D1, 86% of the respondents work at Universities, 9% work at Research labs while 5% work in the Industry. For Question D2, we can argue that the responses could be ordered as follows (in decreasing order of credibility): *Professor* > *Researcher* > *Post Doc* > *Graduate Student* > *Undergraduate Student*. Figure 3(a) shows that our respondent population is skewed towards the higher credibility end of the spectrum. Similarly, for question D4, more years of experience should translate into more credibility in general. Figure 3(c) again shows the response skewed towards higher credibility. Figure 3(c) shows the responses from question D4. Finally, the data from question D5, shows that the vast majority of respondents, 86%, are currently involved in clone research and therefore should be more credible than those who are not currently involved in clone research. The answers to Questions D2, D4 and D5 indicate that the respondent pool had high credibility and therefore likely provided answers that are trustworthy.

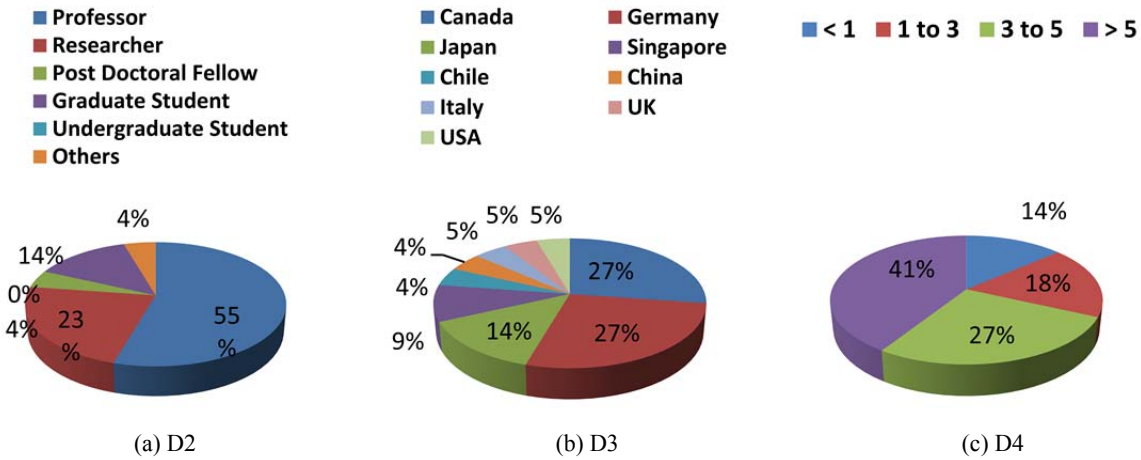
VI. Data Analysis

Before describing the detailed results from the survey, we first describe the analysis process. Because our survey was exploratory, it contained a number of open-ended, qualitative questions. To analyze these responses, we adopted a systematic qualitative data analysis approach. Two of the authors went through the responses to each question to develop a bottom-up coding scheme (i.e., directly from the data rather than from an *a priori* list of codes) that grouped responses into a small number of categories for further analysis.

agreed upon a combined coding scheme and used this coding scheme to classify the survey responses. We compared our results to identify any discrepancies. There were only a small number of responses which were coded differently. The two researches discussed these items and agreed upon a final code for each one. We used the same analysis process for all of the open-ended questions. The following subsections provide and analysis for each of the 3 distinct research thrusts defined in Section II.

Table 1: Demographic Section

DEMOGRAPHIC QUESTIONS
D1. What type of institution is your primary employer?
D2. At the institution indicated in Question D1, what is your primary role?
D3. In which country are you currently working?
D4. How long have you conducted research in the area of code clones?
D5. Are you currently conducting research in the area of code clones?



(a) D2 (b) D3 (c) D4

Figure 3: Demographics

A. RT1: General Clone Usage Information

Table 2 shows the general survey questions. The respondents were fairly evenly divided with regards to whether *clone ratio* is a measure of system quality (45% - no vs. 55% - yes). Further study is required to determine whether clone ratio is a useful measure of quality. The remainder of this section gauged whether members of the community tended to agree upon the definitions clone types described in Section II. Figure 4 shows the distribution of the answers for all four clone types.

While all respondents agreed upon the definition for Type I clones, one preferred the term "identical clones" to the term "Type 1 clones". Regarding the definition of Type II clones, the three respondents who disagreed with the definition generally expressed concern with the ambiguity of this definition as compared to the definitions of Type I and Type III clones. Similarly, the respondents who disagreed with the definition of a Type III clone had issues with the ambiguity of the boundaries between Type II, Type III and Type IV clones. One thought that Type II and III clones should be merged into one group, two said that Type II clones are more like Type IV clones, and the rest of the negative respondents were concerned about the definition of terms like 'further modifications'. Finally, regarding Type IV clones, of the six respondents who disagreed, one suggested that Type IV clones are the only ones that should be called "code clones" because the other types were mostly identical code. Another respondent suggested that the definition of Type IV clones was too broad. Others expressed their confusion over the boundary between Type III and Type IV clones.

TABLE 2: GENERAL SECTION

GENERAL QUESTIONS
G1. Do you think that the clone ratio in a software system can be a measure of the quality of the system? Please explain briefly.
G2. Type 1: Code fragments are identical except for variations in whitespace, layout, and comments. Do you agree? If no, give your definition?
G3. Type 2: Code fragments are structurally and syntactically identical except for variations in identifiers, literals, types, layout and comments. Do you agree? If no, give your definition?
G4. Type 3: Code fragments are copies with further modifications. Statements can be changed, added or removed in addition to variations in identifiers, literals, types, layout and comments. Do you agree? If no, give your definition?
G5. Type 4: Two or more code fragments perform the same computation but are implemented through different syntactic variants. Do you agree? If no, give your definition?

B. RT2: Clones and Developer Behavior:

The next set of questions focused on understanding developer behavior related to clones while performing maintenance tasks. This section contained two types of questions: questions about specific developer actions/expectations and questions about literature claims regarding the maintenance of code clones. The former type was intended to gather knowledge on the beliefs of the researchers from the community while the later was intended to gather their opinions about claims.

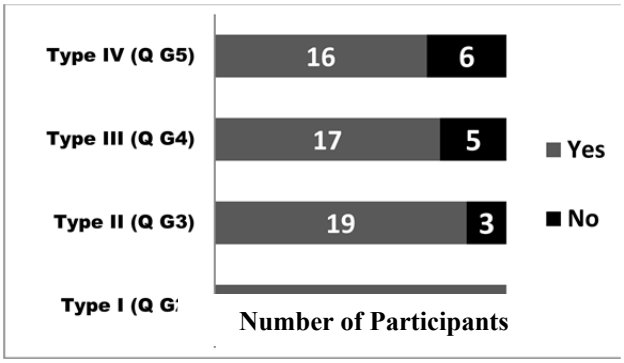


Figure 4: Agreement on Clone Type Definitions

All the observations about developer behavior discussed below are mere hypotheses at this point. Additional empirical studies, focused specifically on these questions can provide the data required to validate or refute these claims.

Due to a lack of qualitative response for three of the questions about 'claims' we were able to analyze only quantitative the responses. The first of these questions asked whether systems with distributed authorship are more prone to inconsistent changes [11]. The responses were: 11 – *Yes*, 2 – *No* and 9 – *Don't Know*. The second question asked whether an experienced developer tends to use a symptom driven approach and go directly to the problem region therefore making a snapshot tool more appropriate for them [5]. The responses were 4 – *Yes*, 1 – *No* and 17 – *Don't know*. The third question asked whether an inexperienced developer uses a typographic debugging strategy (i.e. examining code line by line) making a visualization tool more effective for them [5]. The responses were 4 – *Yes*, 3 – *No* and 15 – *Don't know*.

Question M1 asked the respondents to indicate when developers address clones. Figure 5 shows the answer choices and distribution of responses. Most respondents that answered "other" indicated that it depends upon the task as hand. Interestingly enough, two respondents specifically said that this question needs a study to validate it.

Questions M2 through M8 required open-ended respones and were analyzed using the process described in Section VI. Table 3 shows the results of this analysis. Question M9 relates to a claim from the literature [3]. The first column provides the text of the question. The second column describes the reason why the question was included. The third column presents the elements of the coding scheme that resulted from the anslysis. Finally, the fourth column shows the number responses for each category.

These responses suggest that maintenance tasks which have a broad impact or affect long-term system qualities should be assisted with clone evolution information. Whereas, short term or relatively minor types of maintenance, such as bug fixing or adding modular functionality, do not require expensive information provided by the evolution tools. However, these results are only suggestions; proper studies need to be performed to collect evidence.

A majority of the respondents said that removing clone groups provides long-term benefits to quality. An open question is: What are those long-term benefits? Additionally, the responses indicate that it is better to leave cloned fragments if there is a risk involved that refactoring might render a part of the system or the whole system not to function the way it should.. The respondents thought it okay to independently evolve clone fragments that occur in different contexts. The respondents also thought that developers consistently propagate clones of which they are aware.

C. RT3: Clone Evolution

The last set of questions focused on clone evolution including: late propagation and the impact of system age. Only three questions received enough responses to properly analyze. Because these questions were all open-ended, we followed the analysis process described in Section VI. Table 4 shows the results of the analysis, using the same columns as in Table 3. A majority of the respondents indicated that clone evolution information could be useful for some specific tasks.

VII. Roadmap for Future Work

The primary aim of this survey was to develop a roadmap for empirical research about code clones. This survey identified a number of open questions in need of further empirical study. In this section, we present a list of research question that are derived from the observations earlier in the paper.

First, it is clear that there is a general lack of consensus about the appropriate differentiation among different types of clones. In order to make any significant progress in an area, we must have agreed upon definitions. Therefore, the first open question in need of further research is:

1. How should types of code clones be defined so they provide useful differentiation relative to other important research questions?

Second, in regards to developer behavior during maintenance, there is not widespread agreement on the use of clone information in some key development activities. Currently, many beliefs are based upon anecdotal evidence rather than objective empirical evidence that can result from empirical studies. Such studies would help to eliminate some of the disagreement within the community. Some specific questions in need of further research include:

2. How does *when* a developer addresses clones affect *how* they address clones?
3. How does the type of maintenance (i.e. broad vs. localized) affect the importance of clone information?
4. When is it beneficial to remove a clone group? What benefits can be realized by this action?
5. In what ways can visualization of clone information help developers?

Finally, related to clone evolution, there is a trade-off between the expense of tracking clone evolution information and its importance for version-sensitive maintenance tasks. There is a need to understand how best to use clone evolution information. Specific research questions include:

6. How does the cloning pattern change with system age?
7. Can we identify develop a mechanism for easily identifying cloning patterns which could ease evolution tasks?

TABLE 3: SYSTEMATIC QUALITATIVE ANALYSIS OF MAINTENANCE RELATED SECTION

Question	Intention	Coding	Responses
M2. Describe a maintenance scenario or task when developers track code clones.	Estimate the situations in which developers track code clones so that the tools built to assist the developers can be fine-tuned to certain scenarios.	Refactoring	3
		Fixing bugs	6
		Making a change in multiple locations	3
		Performing a quality assessment	5
		Other answers	2
M3. Describe a maintenance scenario or task where static clone information from the current version of the system is useful.	Estimate the situations in which clone information from the current version is substantial.	Fixing bugs	7
		Refactoring	4
		Performing a quality assessment	4
		Ensuring consistent propagation of clones in a group	2
		Other answers	0
M4. Describe a maintenance scenario or task where clone evolution information over a limited history of the system is useful.	Estimate the situations where more detailed clone information over the multiple versions of the system might be required.	Judge the evolution of the system through version in terms of increase or decrease in the number of clones or other propagation related issues.	3
		Determine history of a ghost fragment that may have diverged out of a clone group in some previous version	2
		Fixing bugs	2
		Tracking the appearance or disappearance of clones	3
		Deciding whether to refactor based on identification of changes that might break the system or affect its quality	4
		Other answers	0
M5. Describe a maintenance scenario or task where you would remove a clone/clone group via refactoring.	To judge scenarios where developers get rid of clones from the system, thus lowering the clone ratio over a period of time.	Clones that can be merged into a parameterized function	3
		When removing clones might help improve the quality of the system giving long term benefits.	7
		Clones which are identical or nearly identical.	4
		Buggy code fragments that get affected in a similar way	4
		Other answers	4
M6. Describe a maintenance scenario or task where you would leave a clone/clone group untouched?	To judge scenarios where developers would not touch or refactor clones in order to change system quality.	A change might harm the system	7
		Cloned parts that never change or are never refactored	3
		In case the clones have separately evolved	4
		Not being sure of what to change	6
		Other answers	3
M7. Describe a maintenance scenario or task where it is ok to make an inconsistent change to clones within a clone group.	To judge the reasons of inconsistent propagation of clones. A situation that might lead to ghost fragments.	Planned independent evolution to a part where the contexts of the cloned parts are different	5
		Planned independent evolution where the purpose varies to change a certain cloned fragment	4
		Planned independent evolution to add a new functionality to a particular clone in a group	3
		Other answers	2
M8. Describe a maintenance scenario or task in which you would make a consistent change to a clone/clone group.	To judge a scenario where a developer would find all the cloned fragments in a clone group to evolve them consistently.	Fixing bugs	9
		Cannot be refactored or changed due to constraints	6
		Identical code designed such that a change one part requires a change in others	6
		Other answers	3
M9. Clones located in different files are more likely to be refactored than the clones in the same file [3]	To estimate if a developer necessarily tries to find the code fragments or is this action causal on the ease of finding the cloned fragment.	Proximity enables easy identification and refactoring	3
		Code refactoring is easier to perform at different places in the same file than in different files.	3
		Depends on the situation and the level of coupling	3
		Other answers	0

TABLE 4: SYSTEMATIC QUALITATIVE ANALYSIS OF EVOLUTION RELATED SECTION

Questions	Intention	Coding	Responses
E1. Is clone evolution information useful to developers? Why or why not?	To estimate if the extra resources spent in capturing clone information over multiple versions of the system is worth the extra effort.	To check what has happened to a clone or a clone group over a period of time.	6
		Locate clones that have inconsistently diverged	4
		The see how the code evolved.	6
		Other answers	3
E2. Do you think the clone evolution pattern can be impacted by how long a system has existed (long-lived systems vs. newly developed systems)?	To estimate the effects of long lived systems.	Old and large systems are more prone to code reuse.	3
		Older systems will have more inconsistencies.	4
		Other answers	1
E3. Developers tend to consistently propagate clone changes immediately where needed [11].	To judge developer behavior regarding consistent propagation.	If they are aware of the clones	3
		Other answers	2

VIII. THREATS TO VALIDITY

This section describes the threats to validity for our survey. Related to external validity, our sample contained 22 respondents. Even though that number is small, it still reflects a 31% response rate (which is quite good). Even so, we cannot be sure that the sample is representative of the entire code clone community.

Related to construct validity, many of the survey questions were based on claims from the literature that have not been proven. It is possible that our selection of questions either excluded important topics or could have been misunderstood or misinterpreted by some of the survey respondents. But, we have no evidence of this problem.

Related to internal validity, we performed a bottom-up qualitative analysis of the survey responses. It is possible that we were biased in our interpretation of the answers. We avoided this threat as much as possible by having two researchers independently evaluate the data. Furthermore, we did our best not to "read in" any information to a response and use only the text that was provided.

IX. SUMMARY AND CONCLUSION

This paper presents the results of a survey conducted within the code clone research community to explore the level of agreement among community members on a number of important topics. Twenty-two members of the community responded to our survey request and provided some useful responses. The results show that in some cases, such as the definition of a Type I clone, there is general agreement, while in other cases, such as the effect of clone ratio on a system, there is strong disagreement. These results indicate that there is a need for empirical work to begin providing insight into some of these questions via data that can be collected from human participants.

ACKNOWLEDGEMENTS

We thank all the respondents for taking some time out of their invaluable time to take the survey. We acknowledge support from NSF grant CCF-0915559.

REFERENCES

[1] Baxter, I. D., Yahin, A., Moura, L., Sant'Anna, M. and Bier, L. "Clone detection using abstract syntax trees." In *Proc.Proceedings of the International Conference on Software Maintenance.* 1998. pp. 368.

[2] Bellon, S., Koschke, R., Antoniol, G., Krinke, J. and Merlo, E., "Comparison and Evaluation of Clone Detection Tools," *IEEE Transactions on Software Engineering,* 33(9): 577-591. 2007.

[3] Cai, D. and Kim, M. "An empirical study of long-lived code clones." In *Proc.Proceedings of the 14th International Conference on Fundamental Approaches to Software Engineering: Part of the Joint European Conferences on Theory and Practice of Software.* 2011. pp. 432-446.

[4] de Wit, M., Zaidman, A. and van Deursen, A. "Managing code clones using dynamic change tracking and resolution." In *Proc.Software Maintenance, 2009. ICSM 2009. IEEE International Conference on.* 2009. pp. 169-178.

[5] Jablonski, P. and Daqing Hou. "Aiding software maintenance with copy-and-paste clone-awareness." In *Proc.Program Comprehension (ICPC), 2010 IEEE 18th International Conference on.* 30 2010-july 2, 2010. pp. 170.

[6] Kamiya, T., Kusumoto, S. and Inoue, K., "CCFinder: a multilinguistic token-based code clone detection system for large scale source code," *IEEE Transactions on Software Engineering,* 28(7): 654-670. 2002.

[7] Kim, M., Bergman, L., Lau, T. and Notkin, D. "An ethnographic study of copy and paste programming practices in OOPL." In *Proc.International Symposium on Empirical Software Engineering (ISESE).* 2004. pp. 83-92.

[8] Nguyen-Hoan, L., Flint, S. and Sankaranarayana, R. "A survey of scientific software development." In *Proc.Proceedings of the 2010 ACM-IEEE International Symposium on Empirical Software Engineering and Measurement.* 2010. pp. 12:1-12:10.

[9] Pate, J. R., Tairas, R. and Kraft, N. A., "Clone evolution: A systematic review," Department of Computer Science, University of Alabama, Tech. Rep. SERG-2010-01, Dec. 2010, 2009.

[10] Roy, C. K., Cordy, J. R. and Koschke, R., "Comparison and evaluation of code clone detection techniques and tools: A qualitative approach," *Science of Computer Programming,* 74(7): 470-495. 2009.

[11] Thummalapenta, S., Cerulo, L., Aversano, L. and Di Penta, M., "An empirical study on the maintenance of source code clones," *Empirical Software Engineering,* 15(1): 1-34. 2010.