

# Source Code Plagiarism and the Honor Court

*J. Patrick Van Metre<sup>1</sup>, Stephen H. Edwards<sup>2</sup>*

**Abstract** – Unless there is a dramatic shift in the attitudes of society in the next seven years, plagiarism will continue to be a problem in 2016, as it is today, and as it has been. At institutions where an honor court enforces plagiarism rules, when students are accused of plagiarism, a group of the students' peers serve as a jury to render a verdict on the charges. The members of the honor court are usually sourced from across the institution, which can present difficulties when they are asked to hear cases of alleged source code plagiarism, as many may have little to no computer programming experience. A survey of honor court members, the results of which will be discussed in this paper, indicates that they tend to rely on evidence presented by automated plagiarism detection systems over their own ability to assess whether or not plagiarism has taken place.

*Keywords:* Source code plagiarism, Honor Court.

## INTRODUCTION

The plagiarism of computer source code has long been a problem at academic institutions that offer courses in computer programming. Some students will, for a variety of reasons [Roberts, 6], take source code that has been plagiarized from their classmates or other sources, actively disguise their plagiarism through a number of means [Parker, 4], and submit the code as their own. For over thirty years [Ottenstein, 3] instructors have had access to computer-based tools to automatically compare source code submissions to detect similarities that may be the result of plagiarism. These automated tools, including the popular modern tools MOSS [Aiken, 1] and JPlag [JPlag, 2], have been very helpful in catching many cases of source code plagiarism and providing evidence to support the prosecution of alleged cheaters.

What happens once students are accused of plagiarism varies from institution to institution, but many use an honor court or other form of trial to allow the accused to be judged by their peers. At most institutions, particularly larger ones with a diverse offering of courses, many students will never do any computer programming. This presents difficulties when a panel composed of such students must try a case of alleged source code plagiarism. In cases of alleged natural language plagiarism, most any student is able to understand what is or is not a case of plagiarism, since every student speaks the same natural language. However, it is a small subset of the student population that is a “speaker” of computer programming languages. It can be difficult for those with little or no programming experience to understand the source code evidence presented and how similarities in the source code may or may not be a result of plagiarism.

This paper presents some of the results of a survey of honor court panelists and their attitudes towards computer programming, automated plagiarism detection systems, the accused students and the reporters of alleged cases of plagiarism. We also suggest future work to address the specific needs of honor court panelists who hear source code plagiarism cases.

---

<sup>1</sup> Virginia Polytechnic Institute and State University, Department of Computer Science, 114 McBryde Hall (0106), Blacksburg VA 24061, vanmetre@vt.edu

<sup>2</sup> Virginia Polytechnic Institute and State University, Department of Computer Science, 114 McBryde Hall (0106), Blacksburg VA 24061, edwards@cs.vt.edu

## THE HONOR COURT

Many institutions have established an **honor code** that contains a set of ethical guidelines and principles of conduct by which all members of the institution agree to abide. Those academic institutions which have adopted an honor code often use it as the primary formal mechanism for establishing rules against unethical academic activities in all forms, including cheating or plagiarism. In order to enforce these rules, institutions often use an **honor court**, which is a form of panel or tribunal used to try cases of alleged honor code violations. The formal process of bringing forward and trying cases varies at each institution, but they all have a similar basic form.

The process begins when a professor, teaching assistant, or other student (the **reporter**) reports to the honor court a suspected violation of the honor code by one or more students (the **accused**). After a period of review, a **panel** is convened to hear the case. The panel usually consists primarily of members of the accused's peer group (**panelists**) – other undergraduate or graduate students, depending on the academic level of the accused. The panel will hear evidence presented by the reporter, the accused, and other witnesses, and will render a verdict.

### Composition of Honor Court Panels

The student members of the review panels are drawn from throughout the institution, usually with a distribution of majors, academic levels, and experiences that are representative of the entire student population. At large, diverse institutions, where there are many majors, there may be relatively few panel members who have a background in computer programming. Thus, there may be relatively few on any one panel that have the background to understand evidence submitted in the prosecution of an alleged case of source code plagiarism.

The availability and ease-of-use of source code plagiarism detection systems makes it easier for reporters to assess potential cases of plagiarism. As a result, at some institutions, the number of alleged cases of source code plagiarism brought before the honor court is disproportionate to the rest of alleged honor code violations, given the distribution of majors at the institution. With a relatively high number of source code plagiarism cases, and with a relatively low percentage of panelists with strong programming backgrounds, there may be few panelists in each of these cases who readily understand the evidence being presented. Without the background to understand easily the evidence submitted, the remainder of the panelists may have a difficult time deliberating. To better understand the needs of these panelists and their attitudes towards automated source code plagiarism detection systems, we conducted a survey of several dozen honor court panelists.

## HONOR COURT PANELIST SURVEY

A survey was administered to honor court panelists to complete at their option. The survey assessed the panelists' academic level; department; computer use; programming experience; use of automated plagiarism detection tools; trust in the tools, reporters, and accused students; and experiences in hearing cases that involve alleged source code plagiarism. Some survey questions were multiple-choice, such as academic level; some were open-ended; and some were 7-point Likert questions.

### Backgrounds of the Panelists

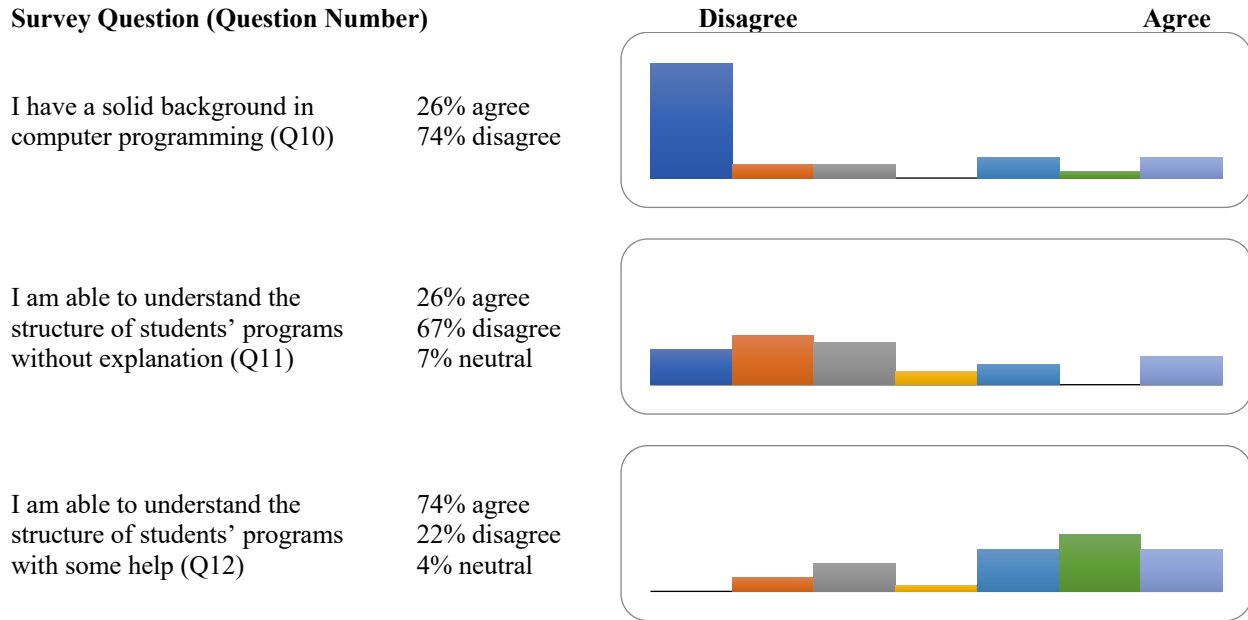
The 32 panelists who responded to the survey were members of a variety of departments from across the university, and were of all undergraduate academic levels, as well as some members of the faculty. Panelists claimed that, on average, they participated in over 15 cases per semester, approximately 58% of which involved alleged source code plagiarism.

Most panelists were from departments that do not involve heavy, if any, computer programming. As seen in Table 1, most panelists (74%) did not feel that they had a strong programming background. Most (67%) felt that they were unable to understand students' programs without some help; but once they receive some explanation, most (74%) were able to understand the students' work.

Some panelists felt that their lack of computer programming ability put them at a disadvantage. Said one panelist (see Table 4 and Table 5 for key quotes from the panelists), "Simply my own ignorance in regards to programming interferes with my ability to judge the case from time to time." Another stated that "Being a non-computer person,

often times cases about programming can get confusing.” A third asserted that “The students that aren’t involved in programming courses don’t really have a clue about how they work.”

**Table 1. Computer programming background of panelists.**



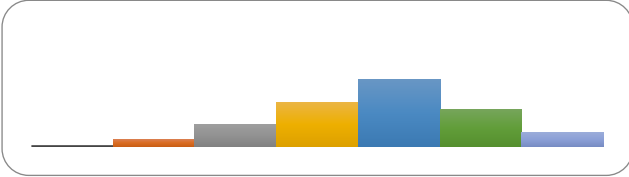
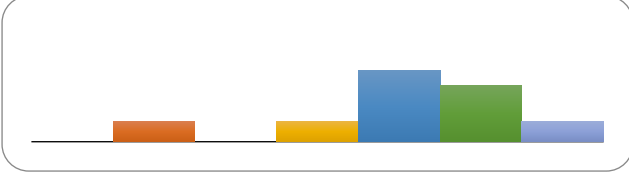
**Attitudes Towards Plagiarism Detection Systems, Reporters, and Students**

All that an automated plagiarism detection system is able to do is to report similarities between students’ assignment submissions; it is left to human beings to determine whether or not similarities are due to plagiarism. When the reporter believes that similarities are due to plagiarism, then he/she brings a case to the honor court for trial. The panelists must understand that while an automated plagiarism detection system may be efficient at finding similarities, a report of similarities by the tool is not a certification that plagiarism took place. Automated tools may find similarities that are not the result of plagiarism; they may be wrong, as may the reporters who bring the cases forward to the honor court.

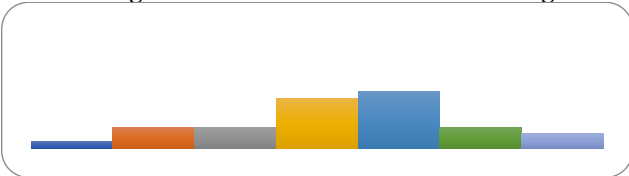
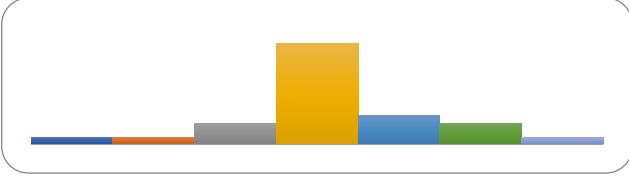
To understand how panelists perceive the evidence provided by the tools, reporters, and the accused, panelists were questioned on how believable they felt each to be. As Table 2 shows, most respondents agree that computers (automated plagiarism detection systems) and reporters may be wrong about the claims they bring forward. Interestingly, panelists tend to believe that there is a greater chance that reporters may be wrong than the computer-based tools; 78% think that reporters may be wrong, compared to 62% for computers. By comparing the responses to questions 23 and 24 for each individual, we can see if each panelist felt that computers or reporters were more likely to be wrong: 41% agree more strongly that reporters are likely to be wrong, 22% agree more strongly that computers are likely to be wrong, and 37% think that they have an equal chance of being wrong.

In terms of believability, a majority (52%) of panelists find all human beings – the accused and the reporters – to be equally believable (Table 3, Q22). The remainder tend somewhat to favor the reporters over the students. However, panelists tend to find computers a bit more believable than the accused students (Table 3, Q21). If we compare each panelist’s responses to Q21 and Q22, we find that 22% of the panelists find students to be less believable than both reporters and computers, while 11% tend to believe students most of all, perhaps giving them the benefit of the doubt.

**Table 2. Panelist beliefs about possibility of error.**

Survey Question (Question Number)		Disagree	Agree
There is a possibility that computers are wrong about students cheating (Q23)	62% agree 15% disagree 23% neutral		
There is a possibility that reporters are wrong about students cheating (Q24)	78% agree 11% disagree 11% neutral		
Comparing individuals' answers to questions 23 and 24:	22% agree more strongly that computers may be wrong 41% agree more strongly that reporters may be wrong 37% agree equally strongly that computers and reporters may be wrong		

**Table 3. Panelist beliefs in students, reporters, and computers.**

Survey Question (Question Number)		Disagree	Agree
If a computer indicates that two students may have cheated, I will believe the computer over the students' claims (Q21)	48% agree 26% disagree 26% neutral		
I would believe the claims of a reporter over the claims of an accused student (Q22)	30% agree 18% disagree 52% neutral		
Comparing individuals' answers to questions 21 and 22:	22% believe a student least of all (less both than reporters + computers) 11% believe students above both reporters + computers		

## ANALYSIS

Honor court panelists put a great deal of trust in the evidence presented by automated plagiarism detection systems – sometimes even more than the trust they place in the reporters and the accused. Most panelists claim not to have the background in computer programming to be able to adequately assess the evidence in source code plagiarism cases on their own, which may be the reason why so much trust is placed in the automated systems. As stated by a panelist from the Computer Science department, “Frequently, panel members without computer training will miss huge similarities. Even after explanation, they miss the significance.” Panelists usually are, however, able to understand the evidence with some help, which comes from the reporters and plagiarism detection systems.

According to one panelist, the “teacher will say how similar programs are but with my limited programming knowledge, I can’t visual[ly] detect/understand these differences without a good deal of help.”

Current plagiarism detection systems provide evidence such as an overall percentage of code that is similar within a pair of assignments, a side-by-side display of regions of similar code within each pair, and the clusters of submissions that are all similar to each other. These metrics are of great value to a reporter, as they identify the groups of assignment submissions that warrant further scrutiny and possible prosecution. However, within each pair of similar submissions, aside from useful feature of highlighting similar code, they often do little to explain how submissions are similar structurally and how those structural similarities may be the result of plagiarism. The onus is currently on reporters to describe how source code submissions are similar and how plagiarism may have been disguised.

Parker and Hamblen [Parker, 4] identify several common means of disguising source code plagiarism, including changing variable and procedure names, rearranging their declarations, and other changes to program logic. These changes are intended to confuse human readers of the code, such as the honor court panelists, but often do not confuse automated systems. Currently, reporters must scrutinize the similar submissions to identify when these changes have taken place, and illustrate those changes to the honor court. If automated tools were able to identify instances of these types of changes, then they could be presented by the tools to the honor court, saving reporters effort and leveraging the trust already placed by panelists in the tools. Several panelists asked for these types of capabilities, as quoted in Table 4.

**Table 4. Do any difficulties arise when hearing cases involving charges of cheating/plagiarism in programming courses?**

“The teacher will say how similar programs are but with my limited programming knowledge, I can’t visually detect/understand these differences without a good deal of help.”

“Simply my own ignorance in regards to programming interferes with my ability to judge the case from time to time.”

“Being a non-computer person, often times cases about programming can get confusing.”

“Frequently, panel members without computer training will miss huge similarities. Even after explanation, they miss the significance.”

“The students that aren’t involved in programming courses don’t really have a clue about how they work.”

**Table 5. Would anything make it easier to hear cases involving charges of cheating/plagiarism in programming courses?**

“Additional examples as well as a break-down of possible program structure to see chances for similarities.”

“Knowing that the automated graders were 100% accurate in finding similarities. From my understanding, if two programs are remotely similar the grader will accuse the person of cheating.”

“A line-by-line comparison of the programs in question with a highlighted section and an explanation [of] why the automated program thought it was cheating.”

“To me, a statistical guess as to the probability of cheating is much less useful than concrete examples of why the program was flagged.”

## CONCLUSIONS AND FUTURE WORK

Current automated plagiarism detection tools provide a valuable capability to instructors enabling them to detect possible cases of source code plagiarism in large classroom settings. These tools are able to identify groups of similar assignment submissions and indicate which regions of source code in each submission are similar to each other. It is currently the job of the reporters to help honor court panelists understand how the evidence shows that similarities are due to plagiarism. Honor court panelists, who generally do not have a strong background in computer programming, have a high level of trust in the output from plagiarism detection systems – even more than they have in the reporters. If automated plagiarism detection systems were able to show in more detail how similarities are a result of plagiarism, and how that plagiarism may have been disguised, a stronger, clear case could be made to the honor court panelists. Such evidence could include how identifiers were renamed, how statements were reordered, or fine-grained similarity metrics for individual program elements such as methods/functions.

The authors believe that this is an area for future work, and are constructing an automated plagiarism detection system to address this need. Current systems generally use string-based comparison algorithms [Schleimer, 7; Prechelt, 5], which treat assignment submissions as strings of tokens. The authors' system compares abstract syntax trees, which are generated by fully parsing the assignment submissions. Thus, more detail on the structure of assignments is available for analysis and presentation to panelists. It is anticipated that this approach will yield detailed evidence that can clarify for non-programmers cases of source code plagiarism heard in an honor court setting.

## REFERENCES

- [1] Aiken, Alex, "MOSS: A System for Detecting Software Plagiarism," <http://theory.stanford.edu/~aiken/moss/>, accessed 19 November 2009.
- [2] JPlag Home Page, "JPlag: Detecting Software Plagiarism," <https://www.ipd.uni-karlsruhe.de/jplag/>, accessed 19 November 2009.
- [3] Ottenstein, K., "An Algorithmic Approach to the Detection and Prevention of Plagiarism," *SIGCSE Bulletin*, 1976, Volume 8, Number 4.
- [4] Parker, Alan and James O. Hamblen, "Computer Algorithms for Plagiarism Detection," *IEEE Transactions on Education*, 1989, Volume 32, Number 2, pages 94-99.
- [5] Prechelt, Lutz, Guido Malpohl and Michael Philippsen, "Finding Plagiarisms Among a Set of Programs with JPlag," *Journal of Universal Computer Science*, 2002, Volume 8, Number 11, pages 1016-1038.
- [6] Roberts, Eric, "Strategies for Promoting Academic Integrity in CS Courses," 32<sup>nd</sup> ASEE/IEEE Frontiers in Education Conference, 2002, Boston, MA, pages F3G-14-19.
- [7] Schleimer, Saul, Daniel S. Wilkerson and Alex Aiken, "Winnowing: Local Algorithms for Document Fingerprinting," *2003 ACM SIGMOD International Conference on Management of Data*, 2003, pages 76-85.

### J. Patrick Van Metre

J. Patrick Van Metre received the BS degree in Computer Science and the MS degree in Computer Science and Applications from Virginia Tech, and is currently working towards the PhD degree. He is currently a Software Engineer with The MITRE Corporation. His research interests include software plagiarism detection, software engineering, modeling and simulation, and programming languages.

### Stephen H. Edwards

Stephen H. Edwards received the BS degree in electrical engineering from the California Institute of Technology, and the MS and PhD degrees in computer and information science from the Ohio State University. He is currently an associate professor in the Department of Computer Science at Virginia Tech. His research interests include software engineering, reuse, component-based development, automated testing, formal methods, and programming languages.