Experimental Evaluation of a Program Visualisation Tool for Use in Computer Science Education

Kathryn Kasmarik, Joe Thurbon

School of Information Technologies
The University of Sydney 2006
Australia

kkasmari@it.usyd.edu.au, joet@it.usyd.edu.au

Abstract

This paper presents an experimental evaluation of a program visualisation tool. Computer science students in an introductory object oriented programming course in Java were asked to respond to a series of questions regarding concepts common to the writing and debugging of code at a novice level. Statistical analysis of data collected from this experiment revealed that a diagrammatic representation can significantly improve the novice understanding of program code.

Keywords: program visualisation, experimental evaluation.

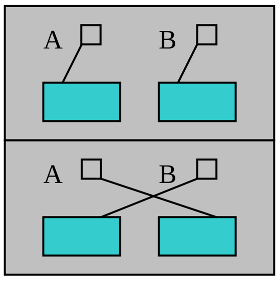
1 Introduction

Representing the state of a computer program diagrammatically is intuitively appealing. Such a representation allows novices to 'see' what is happening as their program executes. The appeal of such a system is reflected in the range of low level program visualisation prototypes that have been developed (Baskerville 1985, Sangwan 1998, Norvell 2000, Sampson 2000, Evangelidis 2001).

(Baskerville 1985, Sangwan 1998 and Norvell 2000) are visualisation systems for C/C++. (Baskervill 1985 and Norvell 2000) are fully automated, (Sangwan 1998) requires some preprocessing of code. (Sampson 2000) presents a visualisation system for Java while (Evangelidis 2001) presents a system for a simple untyped teaching language called X.

A range of other techniques for representing program structure also exists. These include control structure diagrams (Hendrix 2000 and Cross 1999), flow charts and Nassie-Shneiderman diagrams (Nassi 1973). Studies such as (Cross 1998 and Hendrix 2000) evaluate the usefulness of these structures as an aid to improving understanding of program code.

Experience reports exist which describe the usefulness of some of the program visualisation tools described above in qualitative language. However, none of the examples discussed so far has made an empirical study of the value of a program visualisation tool as an aid to improving program comprehensibility. This paper addresses that gap by developing a program visualisation tool and conducting a usability study of the tool.



1.1 The Program Visualisation Tool

(Thurbon 2000) presents an implementation of a diagrammatic programming system. The diagrams used to represent a code state have the characteristics of the classic 'box and arrow' diagrams that might be drawn informally to trace a code segment. An example is shown in Figure 1.

Figure 1. Diagrammatic representation of a 'swap' method.

A reverse implementation of this diagrammatic programming system was performed for the purpose of this project. The reverse implementation takes pure Java code as input. It produces a visualisation of the execution of the code as output. An example code fragment and corresponding visualisation is shown in Figures 2 and 3. This system can represent constants, declarations, assignment, classes, arrays, constructor calls and method calls.

```
public class Swap
{
        public static void main(String[] args)
        {
                Integer a = new Integer(1);
                Integer b = new Integer(2);
                Integer tmp = a;
                a = b;
                b = tmp;
        }
}
```

Figure 2. A Java program to swap the contents of two variables.

The development of a working program visualisation tool for the purpose of this study was considered important for a number of reasons. Firstly, a working tool is more realistic than a usability study of hand drawn trace diagrams. Elements of a hand drawn trace diagram are generally arranged to present the most readable and aesthetically pleasing visualisation of code state. Layout algorithms used in program visualisation tools are not able to produce such an optimal visualisation in all cases.

Development of a working program visualisation tool also provides the potential for its use as a teaching aid should experimental results prove favourable. A usability study of hand drawn diagrams would not conclusively identify a specific tool as a beneficial teaching aid.

The tool developed for this study was also used to investigate the application of a new layout algorithm (Kasmarik 2001) (out of the scope of this paper) to the program visualisation domain.

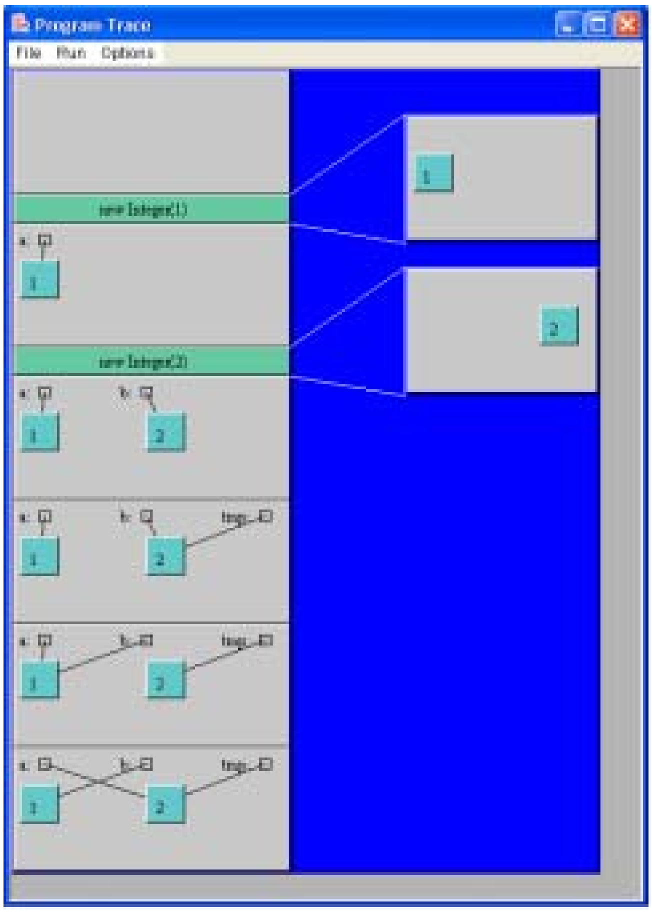


Figure 3. Trace of Swap.java

1.2 Experimental Goals

This study attempts to validate the intuitive appeal of a software visualisation tool as an aid to understanding program code. Specifically, it does this by answering the following question about our diagrammatic program representation:

- Does the diagrammatic representation improve understanding of program functionality?

The operational hypothesis for this experiment is as follows:

- The diagrams will have a positive effect on program comprehensibility.

The null hypothesis is:

- The diagrams will not have a positive effect on program comprehensibility.

2 Experimental Method

2.1 Selection of Participants

The participants in this study are computer science students enrolled in an introductory object oriented programming course in Java. They have limited or no programming background, other than what they have learnt in the course so far. Any previous programming experience is usually in a different language such as Visual Basic.

138 students participated in the study. The experiment required that these students be divided into two groups. To ensure the validity of a comparison between the results produced by these groups, it was necessary that they be approximately equal in terms of academic performance. To facilitate this, the division was made based on marks attained in the first assignment for the course. This division is shown in Figure 4. While circumstances did not permit the division of each mark bracket to be exactly equal, the average mark obtained by students in different groups differed by only 0.1. A t-test showed that this difference is not significant at the 0.05 level.

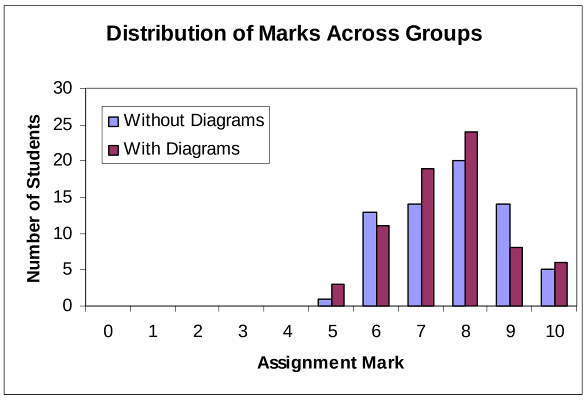


Figure 4. Performance Balance of Groups.

2.2 Questions

The study consisted of nine questions. Each question referred to a fragment of Java code. In order to make the experiment realistic, the code fragments used were similar to those used in lecture and tutorial material. The questions addressed tasks common to the reading, writing, debugging and testing of computer programs at a novice level.

The questions addressed the following concepts:

- Generic code tracing (Q1)
- Object equality (Q2, Q3)
- Run-time errors (Q5)
- Data structures (Q4, Q6)
- Pass-by-reference (Q7, Q8)
- Recursion (Q9)

A majority of the questions covered concepts specifically taught in the course the participants were completing. This subset will be referred to as T for the remainder of this paper. The questions relating to data structures and recursion (Q4, Q6, Q9) were included to investigate the impact of diagrams on the understanding of new concepts. This subset will be referred to as T'.

The questions were designed to have a single, unambiguous answer. In most cases they requested a one-word answer or a true/false choice. Correctness was decided on a 'right or wrong' basis. That is, a question was either deemed correct or incorrect. A serious response to the study was defined as one that attempted to answer all nine questions. Responses that did not attempt to answer all questions were disregarded.

2.3 Presentation of Questions

To facilitate accurate measurement of response times, and to resemble as closely as possible the process of debugging source code, the questions were presented automatically by a computer program. Two systems were used to present results. Each system consisted of a series of screens containing a question, a code fragment, a text-field or set of radio buttons and a submit button. Screens in one of these systems also contained a diagrammatic representation of the code fragment. An example is shown in Figure 5. The corresponding screen in the other system was identical other than the omission of the diagram on the right of Figure 5. Response time was measured as the time from when the question was displayed to when the response was submitted.

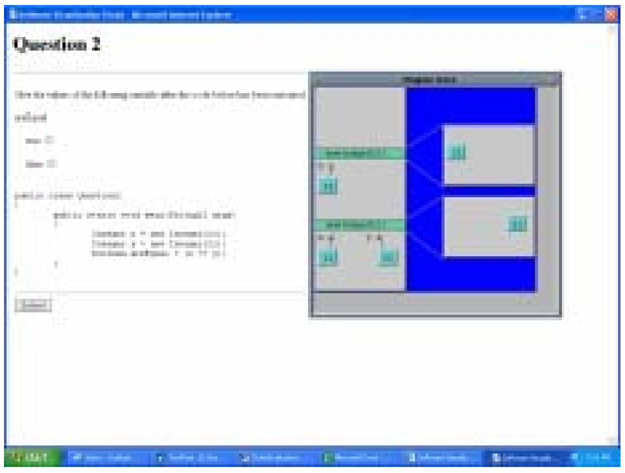


Figure 5. Presentation of Questions.

2.4 Testing Procedure

The questions were presented to students as a tutorial exercise. Collaboration was discouraged and distractions and interruptions minimised. Due to the nature of a tutorial it may not have been prevented entirely in all cases. However it is not envisaged that these cases would impact significantly on a statistical analysis of the results across a group of this size.

Participants were divided into two equal groups according to the performance balancing information in Figure 4. Both groups were presented with source code and asked to respond to a series of questions about its content as shown in Figure 5. One group (the control) received only the code, the other group received the code and corresponding diagrams produced by the system. Before answering the questions, both groups were given identical instructions concerning the completion of the experimental tasks. The group using the diagrams was given extra instructions to introduce the basic symbols used by the system.

The independent variable for this experiment was whether or not participants have access to the diagrams. Other variables such as gender and English language ability were not considered. It was assumed that the distribution of these variables was even across both groups.

The task presented to the students was to answer each question correctly in the shortest possible time. Response time and correctness were thus the two dependent variables. It was assumed that any effects of the diagrams on program comprehensibility would be captured by these statistics.

3 Results

There were 112 serious responses to the study. 53 were in the group using the diagrammatic aid, henceforth referred to as D. 59 were in the group without the aid of diagrams, henceforth referred to as D'.

Analysis of the difference in performance of students in D and D' covered a number of areas. These include time, correctness and efficiency. Analysis was done across all questions and for the subset T defined in Section 2.2. Analysis of the subset T' has been omitted as this subset does not contain enough questions to draw significant conclusions.

3.1 Time to Respond

Two time statistics were considered. These are average total response time without regard for correctness and average time for a correct response. These statistics have been calculated across all questions and for the subset of questions T.

Figure 6 graphs the average response time without regard for correctness. The graph shows the mean response time and the 95% confidence interval for D and D' for each question.

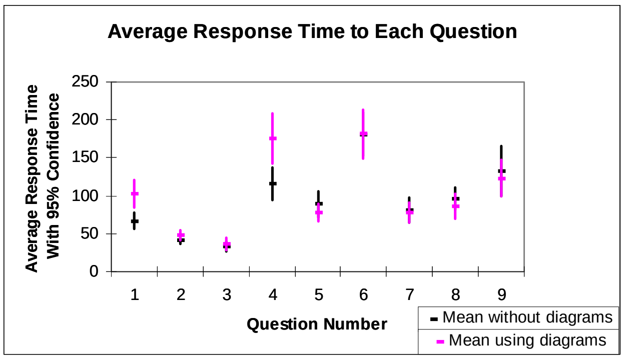


Figure 6. Average response time for each question.

The graph shows that the diagrams increased response time for 5 of the 9 questions. However this increase was only significant for Q1 and Q4 in which the confidence intervals do not overlap. In the case of Q1, this may be explained by the fact that this was the first question and thus the first experience the students had with the diagrams. As a result they studied it longer in order to become familiar with the conventions used.

The large time difference in Q4 may be explained by the fact that this is the first question from T' and that the diagram was significantly more complex than those in the preceding questions.

The average total response time was increased by 9.6%. However a t-test reveals that this increase is not statistically significant at the 0.05 level. When the subset T is considered alone, the average total response time is increased by just 5.6% which is also insignificant at the 0.05 level.

Figure 7 graphs the average response time for correct responses. Only Q1 and Q9 display significant time increases as evidenced by the disjoint confidence intervals. The difference in Q1 was explained above. The difference in response times for Q9 should be considered in light of the fact that it was the most complex question in the study and there were no more than 10 correct responses by either D or D'.

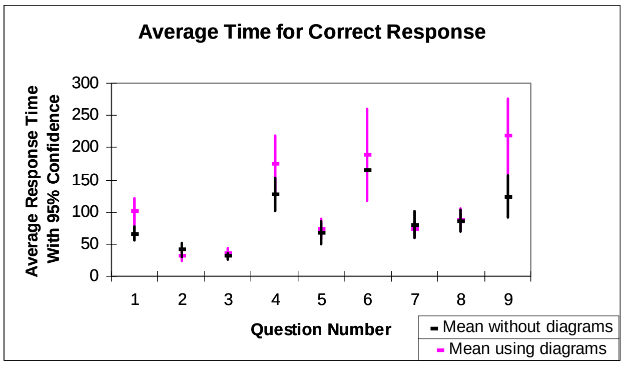


Figure 7. Average response time for correct answer to each question.

The previous Figures show no significant differences in the times taken by D and D'. Thus we cannot reject the null hypothesis based upon the analysis so far. The next section continues our analysis with respect to correctness.

3.2 Correctness of Responses

This section compares the percentage of correct answers produced by students in D and D'.

Figure 8 graphs the percentage of each sample group to achieve each mark out of 9. D is significantly higher than D' for Q1, the general code tracing question as evidenced by the disjoint confidence intervals. Q7 and Q8, the two questions covering the pass-by-reference concept show the next most significant performance improvement as their confidence intervals overlap only partially.

Finding the correct answer to these 3 questions involves understanding the final state of the program after the code fragment has executed. This is a task to which our visualisation tool is particularly suited as it involves a simple inspection of the final diagram from which variable values may be obtained.

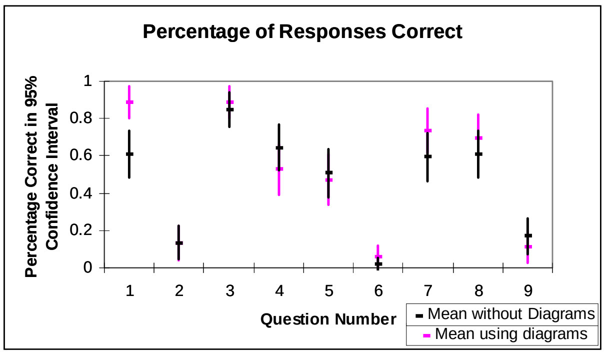


Figure 8. Percentage of students with correct response for each question.

It was expected, for similar reasons, that D be higher than D' in Q5, which asked students to identify which variable was causing a Java `NullPointerException`. However this was not the case. This is most likely due to the distinction that the diagrams make between a null value, represented by a coloured box, and an 'unknown' value, represented by an empty box. The coloured box may have lead students to believe that the null variable actually had some value as this was not covered in the initial explanatory session. This result, although undesirable is still an interesting indication of the power of a diagrammatic representation.

Q2, Q6 and Q9 had very low correct answer rates for both D and D'. Q2 is in T while Q6 and Q9 are in T'. This suggests that, regardless of whether a concept has been taught or not, our diagrams could not improve performance unless there was some initial understanding of that concept.

Figure 9 shows the percentage mark distribution for D and D'. There was a 9.5% improvement in the average mark of students using the diagrammatic aid, however this is not statistically significant at the 0.05 level.

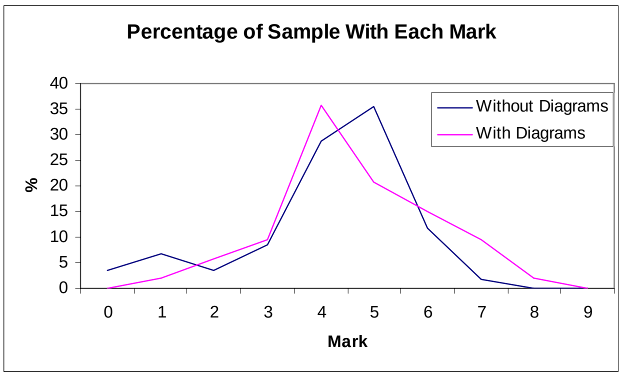


Figure 9. Percentage of students with each mark.

Figure 10 summarises the information presented above. It shows a difference plot of the percentage of correct responses in D and D'.

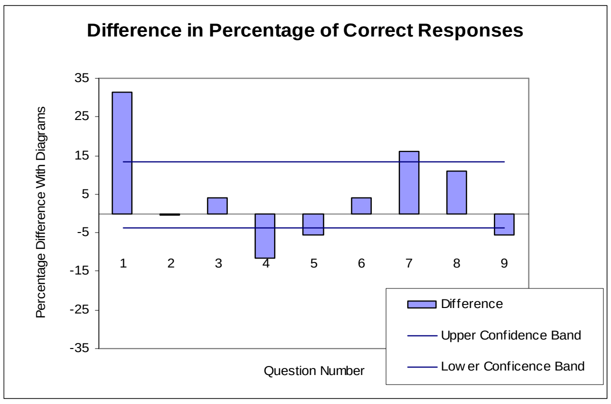


Figure 10. Difference plot for correctness.

Figure 10 emphasises that Q1, Q7 and Q8 display the most significant increase in correctness. Q4 displays the most significant decrease. The 95% confidence interval has been included on this plot. Because this confidence interval spans the x-axis or null-hypothesis line we conclude that the evidence presented so far has not rejected the null hypothesis. We now move our analysis to the subset T.

Figure 11 graphs the mark distribution over the 6 questions on taught material. There was an improvement of 18.2% in the average mark of students using the diagrammatic aid. This improvement is significant at the 0.05 level. It continues to be significant down to the 0.01 level. This result conclusively rejects the null hypothesis.

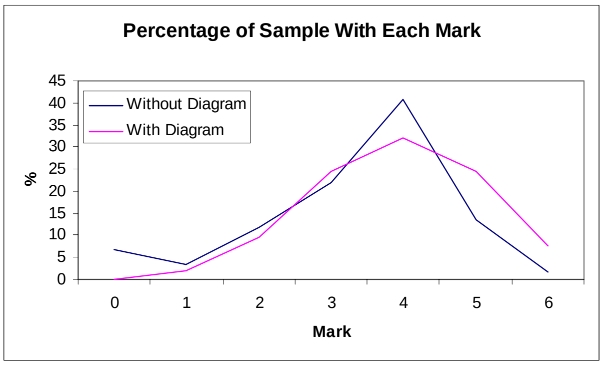


Figure 11. Percentage of students with each mark for questions on taught material.

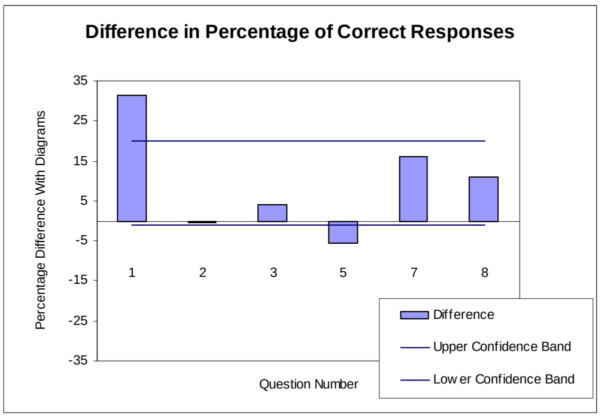


Figure 12. Difference plot for correctness over T.

Figure 12 supports our rejection of the null hypothesis in graphical form. It shows the relevant data points from Figure 10 and recalculates the 95% confidence interval which now barely spans the x-axis. The intersection that remains is caused by the values for Q5. However the decrease in correctness for this question is most likely due to a lack of understanding of diagrammatic conventions described earlier and may thus be ignored.

This section has rejected the null hypothesis by showing a significant increase in correctness of responses over the subset of questions pertaining to taught material. The next section discusses this further with reference to the efficiency with which students produce correct responses.

3.3 Participant Efficiency

We define efficiency as the number of correct answers produced per minute for each question. Figure 13 compares the efficiency of students in D and D’ for each question. It shows increased efficiency for Q5, Q7 and Q8, all questions to which we believe our diagrams are particularly well suited.

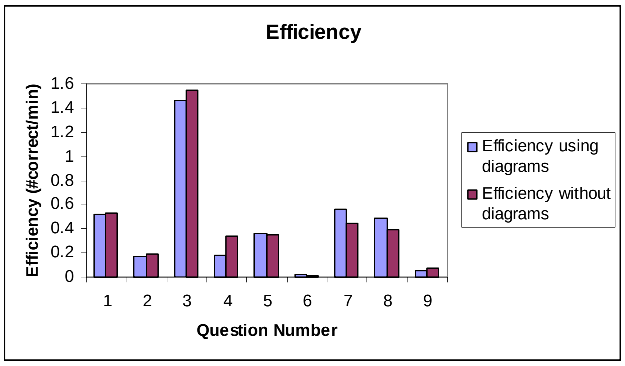


Figure 13. Efficiency.

Figure 14 summarises the efficiency results presented above as a difference plot. The 95% confidence interval spans the null hypothesis. This is the case over all questions and for the subset T shown in Figure 15. As a result we conclude that our diagrams have not significantly altered efficiency. We consider this in light of the fact that the number of correct answers produced by students was 9.8% higher among those using diagrams and 18.2% higher among those using diagrams in T. This means that the use of our diagrammatic aid has produced a significant increase in correctness without affecting efficiency.

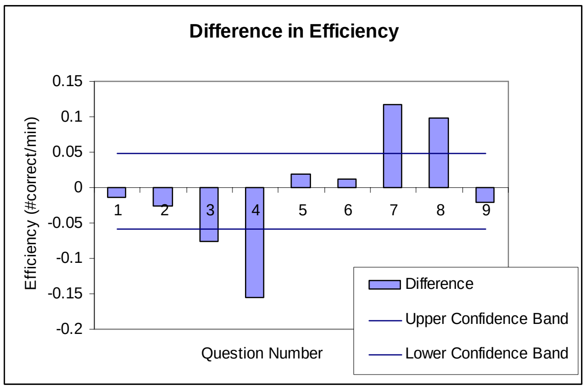


Figure 14. Efficiency difference.

It is also important to note that efficiency is dependent upon time. As students become more familiar with the diagrammatic conventions they will complete questions more quickly. This will improve their efficiency levels.

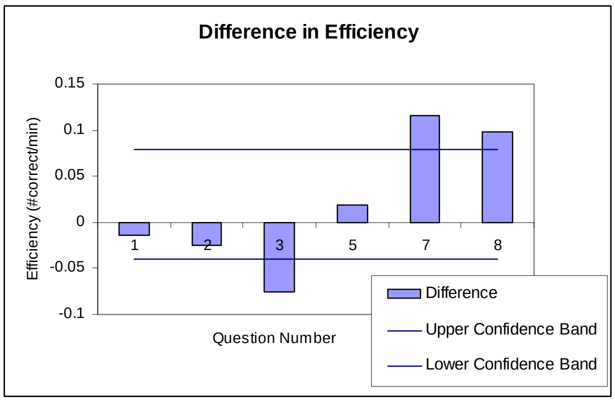


Figure 15. Efficiency difference over T.

4 Conclusion

A number of conclusions follow from the results presented above. With regard to taught material, we have shown that use of a diagrammatic aid increases correctness by 18.2% without impacting significantly on the time required to respond. In addition, we can expect that the time required to answer a question would decrease as students become more familiar with diagrammatic conventions. These facts conclusively reject the null hypothesis and accept the operational hypothesis that our diagrams improve code comprehensibility.

With regard to new material, we accept the null hypothesis that our diagrams do not improve code comprehensibility as there was no significant change in either the time taken to respond or the correctness of the responses.

In summary, our diagrams improved student understanding of concepts specifically taught in the course participants were completing. However initial instruction is still required to provide a knowledge base for the additional benefits of the diagrammatic representation.

5 Future Work

The data collected from this study suggests several avenues for future work. Firstly, we have suggested that greater familiarity with the diagrammatic conventions would decrease the time required to use them. It would be an interesting exercise to run a similar study with the addition of a more substantial tutorial session for the groups of students using the diagrammatic aid. This could include formal instruction and a set of exercises that allow student to experiment with the program visualisation tool.

Results have also suggested that our diagrams improve understanding of material on some concepts more than others. For example, the diagrams appeared particularly effective when used as an aid to general code tracing questions and questions relating to the pass-by-reference concept. However, this study does not contain enough questions on any topic to prove or disprove this claim. A longer study, containing more questions of each type is required for this.

Finally, the program visualisation tool developed for this study is designed to function as a 'graphical debugger'. A user can load their program into the system and step through the code line by line. Each step adds a new diagram to the sequence of code states produced by the program execution. In the interest of simplifying the study, this functionality was not provided to participants. This functionality could be included in future studies to explore the additional benefits of such a tool.

6 References

BASKERVILLE, C. (1985): *Graphic Presentation of data structures in the dbx debugger*, Technical Report UCB/CSD 86/260 UCBerkeley.

CROSS, J., MAGHSOODLOO, S. and HENDRIX, T. (1998): *The Control Structure Diagram: An Initial Evaluation*, Empirical Software Engineering **3**(2):131-156.

EVANGELIDIS, G., DAGDILELIS, V., SATRATZEMI, M. and EFOPOULOS, V., (2001): *X-Compiler: Yet another integrated novice programming environment, advanced learning technologies, 2001.* Proceedings, IEEE International Conference on, 2001, 166-169.

HENDRIX, D., CROSS, J. and MAGHSOODLOO, S. (2000): *An Experimental Validation of Control Structure Diagrams*, Proceedings of 11th Working Confernece on Reverse Engineering, 2000, 224-240.

KASMARIK, K. and THURBON, J. (2002): *Foresighted Layout in Program Visualisation.* OzViz 2002.

NASSI, I. and SHNEIDERMAN, B. (1973): *Flowcharting Techniques for Structured Programming*, SIGPLAN Notices, **8**(8)12-26.

NORVELL, T. and BRUCE-LOCKHART, M. (2000): *Lifting the hood of the computer: Program animation teaching machine, Electrical and computer engineering, 2000* Canadian conference on, **2**:831-835.

SAMPSON, S. (2000): Software Visualisation Tools for Java, MSc Thesis, Acadia University Canada, 2000.

SANGWAN, R. KORSH, J. and LAFOLLETTE, P. (1998): *A System for Program Visualisation in the Classroom*, ACM SIGCSE Bulletin, Proceedings of the 29th SIGCSE technical symposium on Computer Science Education March 1998, **20**(1).

THURBON, J. (2000): Programming with Pictures and Homomorphic Planning, PhD Thesis, University of New South Wales, 2000.