Automated Tool to Assess Pair Programming Program Quality

Mazni Omar[a], Rohaida Romli[b], Azham Hussain[c]

*[a,b,c]Graduate Department of Computer Science,
College of Arts and Sciences,
Universiti Utara Malaysia,
06010 UUM Sintok, Kedah,
Malaysia*

[a]E-mail : mazni@uum.edu.my

[b]E-mail aida@uum.edu.my

[c]E-mail : azham.h@uum.edu.my

ABSTRACT

This paper aims to present an automated tool that has been developed to assess pair programming program quality. The tool known as Java Quality Measurement Tool or JaQMeT is used to assess specifically Java program quality. There are two program quality factors that can be assessed which are correctness and complexity. Pair programming program will be graded using JaQMeT. Then the results will be used to evaluate the effectiveness of pair programming. JaQMeT is only at its initial stage. It is an initial effort to facilitate the lecturers to reduce workload on grading programming assignment and specifically to assess pair programming program quality. Although JaQMeT has its several limitation but it is hoped that JaQMeT can be extended by using web-based technology and capable to check others program quality.

Keywords

Pair programming, program quality, automated tool

1.0 INTRODUCTION

Pair programming is a kind of collaborative programming where two people work side-by-side on design, implementation, and testing with one computer (Beck, 2000). William, Kessler, Cunningham & Jeffries (2000), describes pair programming process as follows:

"In pair-programming, two programmers jointly produce one artifact (design, algorithm, code, etc.).One partner is the "driver" and has control of the pencil/mouse/keyboard and is writing the design or code. The other partner continuously and actively observes the work of the driver – watching for defects, thinking of alternatives, looking up resources, and considering strategic implications of the work at hand. The roles of driver and observer are deliberately switched between the pair periodically."

Pair programming is one of the core practices in Extreme Programming. It is claimed that pair programming can promote team work and thus produce high quality software (McDowell, Werner, Bullock & Fernald, 2002; DeClue, 2003; William & Kessler, 2003; Hanks, McDowell, Draper & Krnjajic, 2004; Xu & Rajlich, 2005).

In order to assess program quality, there is a need to develop a tool. The tool can assist the programmer specifically to automate the process of evaluating program quality. On the other hand, the tool can also reduce workload of teaching staff or lecturer in grading the programming assignment.

Pressman (2001) defined program quality as conformance to explicitly state functional and performance requirement, explicitly documented development standard, and implicit characteristic that are expected of all developed program. There are various quality factors in order to assess program quality such as correctness, complexity, maintainability, portability, and others. In this study two main quality factors will be used which are correctness and complexity. Correctness is a degree to which the program performs its required function (Pressman, 2001). Most correctness assessment is empirical and based on checking a program's output from inputs. If the program meet its requirement, formally a program is correct. Correctness is a straightforward criterion to formulate the program quality. Meanwhile, software complexity metrics is developed to identify parts of program that are likely to be difficult to test, understand, or error-prone.
Most studies on pair programming investigate an impact of pair programming on program quality and then provide empirical evidence on the effectiveness of the practices. However the main aim of this study is to develop an automated tool to assess pair programming program quality. The tool will measure two main quality factors which are correctness and complexity of the program.

2.0 RELATED WORKS ON PAIR PROGRAMMING QUALITY TOOL

In this section, related works of the study which includes pair programming quality tool will be described. A summary of pair programming literature which focuses on program quality is given in Table 1: Summary of Literature. Most of the literatures were obtained from the ACM and IEEE databases.

Table 1: Summary of Literature

Author(s)	Quality Metrics	Method to measure program quality	Using Automated Tool?
Williams et al. (2000)	Functionality	Using automated testing tool that executed by impartial teaching assistant	Yes
Ciolkowski & Schlemmer (2002)	Complexity ▪ Lines of Codes (LOC) ▪ Comment Ratio ▪ Couplin g Factor	Using evaluation tool to compute worst-case complexity of system	Yes
McDowell, Werner, Bullock & Fernland (2003)	Functionality Readability	Using student's score on graded programming assignment	No
DeClue (2003)	Functionality	Using student's score on graded programming assignment and qualitative survey	No
Gehringer (2003)	Functionality	Using student's score on graded programming assignment	No
McDowell, Hanks & Werner (2003)	Functionality Readability	Using student's score on graded programming assignment and qualitative assessment which include functionality, style and holistic	No
Hanks et al. (2004)	Complexity	Using JavaNCSS to calculate source code metric of program	Yes
Xu & Rajlich (2005)	Lines of Code (LOC) Readability	Using qualitative assessment	No
Arisholm, Gallis, Dyba & Sjoberg (2007)	Correctness	Reviewed by two independent senior consultants by using the following tool; Correctness analysis tool, Each solution tested using automated test scripts and final grading using web-based grading tool	Yes

There are various methods to measure program quality. Some studies that focus on functionality quality metrics have used student's score on graded programming assignment to assess pair programming program quality (McDowell, et al., 2003; DeClue, 2003; Gehringer, 2003). In contrast, Williams et al. (2000) have used automated tool to assess functionality quality metric in their study.

Ciolkowski & Schlemmer (2002) and Hanks et al. (2004) have performed pair programming experiments by using an automated tool to compute complexity of the student's pair programming program. In addition, Arisholm et al. (2007) have assessed correctness of quality program using correctness analysis tool, automated test scripts and automated web-based grading tool. Besides, another method to measure pair programming quality in terms of readability and lines of codes is by using qualitative assessment (McDowell, Hanks & Werner, 2003; Xu & Rajlich, 2005).

3.0 DESIGN OF JaQMeT

An automated tool which known as Java Quality Measurement Tool (JaQMeT) has been developed to assist teaching staff or lecturer to assess Java program quality. JaQMeT is able to assess two program quality factors which are correctness and complexity. It is designed based on the integration of two studies proposed by Rohaida, Fazilah & Mazni (2004) and Mawarny & Rohaida (2005). Proper enhancements and modifications had been made to fit with current requirements.

Two program quality factors were chosen to assess the quality of student's program. These two types of program quality were adequate to assess program quality that applied basic Java programming concepts. Apart from that JaQMet is used as quality measurement tool to assess program quality produced by student that applied pair programming practices. Figure 1 depicts a process of quality assessment in JaQMeT.

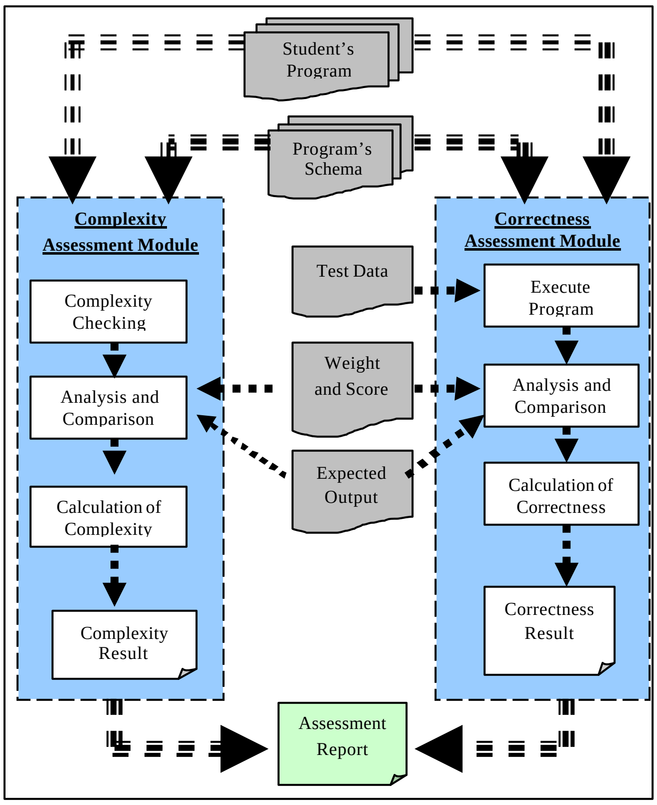


Figure 1: Quality Assessment Process of JaQMeT

As shown in Figure 1, the process of measuring program complexity is done by implementing static analysis to the student's program and program schema. Then, both programs are analyzed by comparing their complexity values to see their similarity results. Finally, the weight value and score are assigned to each selected metric and calculation of complexity mark is executed to obtain the mark given to the student's program. There are two software metrics adopted (Abounader & Lamb, 1997; Xenos, Starrinoudis, Zikouli & Christtodoulakis, 2000) for complexity checking in JaQMeT:

- Cyclomatic Complexity metric - measures the amount of decision logic in a single software module. Cyclomatic complexity is defined to be **e – n + 2**, where e and n are the number of edges and nodes in control flow graph, respectively. This cyclomatic complexity is measured for each method in class.
- Operation Complexity of a class metric - defined $\sum$ O(*i*), where O(i) is operation i's complex value. Summing up the O(i) in for each operation i in the class gives their metric value.

Meanwhile, the correctness of student's program is assessed by using the equivalence partitioning technique. Equivalence partitioning technique is one of the black-box testing techniques and adopted to design the test cases used to assess the correctness of the program. Black-box testing examines the functional operation of the system. This means that the program is executed with given test data, and the output of student's program will be compared with the program schema to determine whether the output is correctly produced (Chu et al., 1997). Specific score and weight were assigned to each test case as a measurement of the program correctness. The assessment result contains marks of student's program.

There are eight main functions consist in JaQMeT;

- Assess Program Correctness
- Upload File Schema
- Set Weight Value
- Assess Program Complexity
- Set Test Data
- Upload Question
- View Assessment Result
- Upload Student Program

Figure 2 illustrates a use case diagram that maps respectively to all functions with the actors of the system. Each use case shows how an actor interacts with system and what the system does. The use case has a set of sequence actions and performs observable result to a particular actor, who interacts with the system.

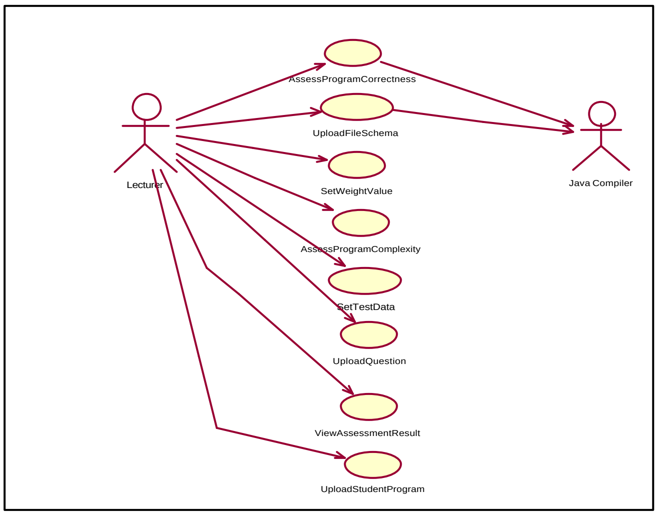
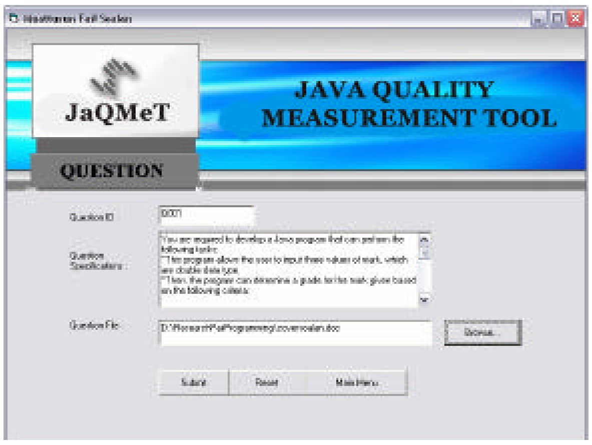


Figure 2: JaQMeT Use Case Diagram

As shown in Figure 2, there are two actors involved in managing the functionalities of the JaQMeT, namely lecturer and Java compiler. Lecturer is a person who evaluates student's program and plays an important role in preparing and managing source needed in processing Java program assessment, managing program schema, and managing weight value for measuring correctness and complexity of program. Java Compiler is an external entity or independent software used to compile and interpret a Java program to be assessed.

All interfaces of JaQMeT system were developed based on number of use cases that have been defined. Figure 3 to 8 depict sample interfaces that map the predefined use cases.

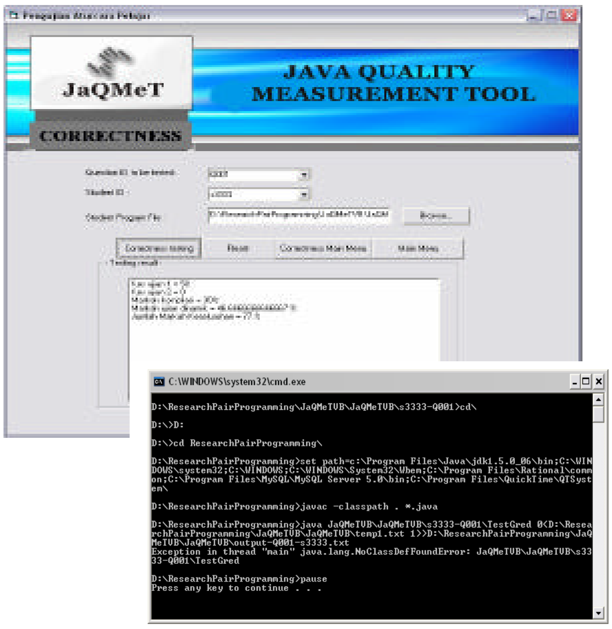


Question ID: Q001

Question Specifications: You are required to develop a Java program that can perform the following tasks: *This program allows the user to input three values of mark, which are double data type. *Then, the program can determine a grade for the mark given based on the following criteria:

Question File: D:\ResearchPairProgramming\...soalan.doc

Figure 3: Upload Question



Question ID to be tested: Q001

Student ID: s3333

```
D:\ResearchPairProgramming\JaQMeTVB\JaQMeTVB\s3333-Q001>cd\

D:\>D:

D:\>cd ResearchPairProgramming\

D:\ResearchPairProgramming>set path=c:\Program Files\Java\jdk1.5.0_06\bin;C:\WIN
DOWS\system32;C:\WINDOWS;C:\WINDOWS\System32\Wbem;C:\Program Files\Rational\comm
on;C:\Program Files\MySQL\MySQL Server 5.0\bin;C:\Program Files\QuickTime\QTSyst
em\

D:\ResearchPairProgramming>javac -classpath . *.java

D:\ResearchPairProgramming>java JaQMeTVB\JaQMeTVB\s3333-Q001\TestGred 0<D:\Resea
rchPairProgramming\JaQMeTVB\JaQMeTVB\temp1.txt 1>>D:\ResearchPairProgramming\JaQ
MeTVB\JaQMeTVB\output-Q001-s3333.txt
Exception in thread "main" java.lang.NoClassDefFoundError: JaQMeTVB\JaQMeTVB\s33
33-Q001\TestGred

D:\ResearchPairProgramming>pause
Press any key to continue . . .
```

Figure 6: Assess Program Correctness

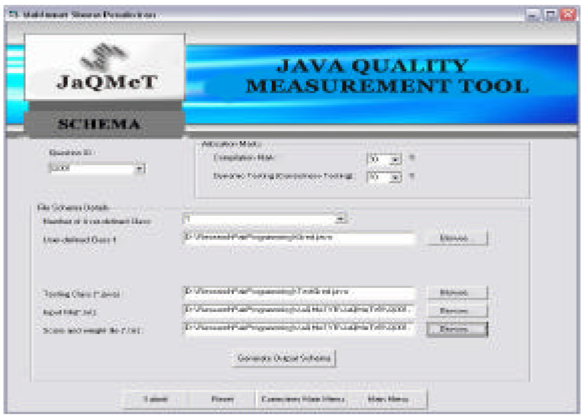
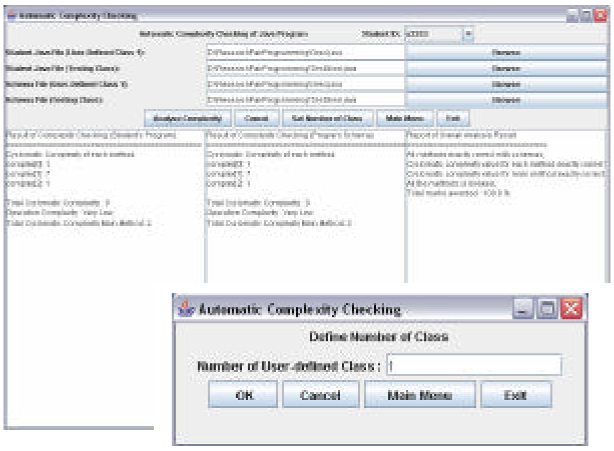


Figure 4: Upload File Schema

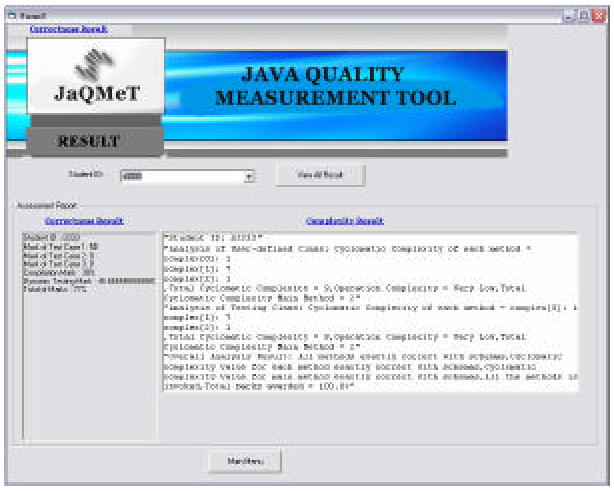

Automatic Complexity Checking
Automatic Complexity Checking of Java Program
Student ID
Browse
Analyse Complexity
Cancel
Set Number of Class
Main Menu
Exit

Automatic Complexity Checking

Define Number of Class

Number of User-defined Class : 1

Figure 7: Assess Program Complexity



"Student ID: s3333"
"Analysis of User-defined Class: Cyclomatic Complexity of each method : complex[0]: 1
complex[1]: 1
complex[2]: 1
,Total Cyclomatic Complexity = 3,Operation Complexity = Very Low,Total Cyclomatic Complexity Main Method = 2"
"Analysis of Testing Class: Cyclomatic Complexity of each method = complex[0]: 1
complex[1]: 1
complex[2]: 1
,Total Cyclomatic Complexity = 3,Operation Complexity = Very Low,Total Cyclomatic Complexity Main Method = 2"
"Overall Analysis Result: All methods exactly correct with schemas,Cyclomatic complexity value for each method exactly correct with schemas,Cyclomatic complexity value for main method exactly correct with schemas.All the methods is invoked.Total marks awarded = 100.0%"

Figure 8: View Assessment Result

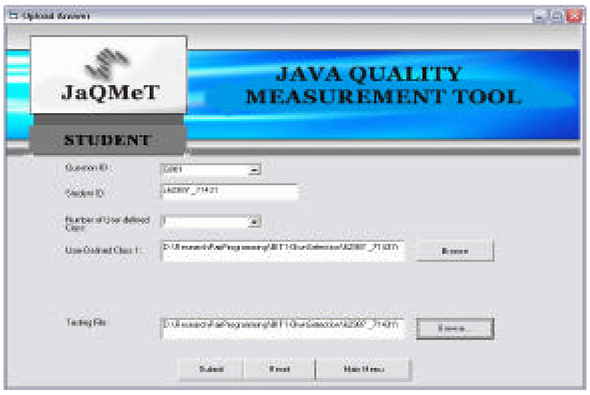


Figure 5: Upload Student Program

The following briefly explained Figure 3 to 8:

- Upload Question interface is used to upload questions to be solved by students.
- Upload File Schema interface is used to generate schema output for a given question.
- Upload Student Program interface is used to upload a student's program submitted to be assessed.
- Assess Program Correctness interface is used to assess the correctness of student's program.
- Assess Program Complexity interface is used to assess complexity of student's program.
- View Assessment Result interface is used to view student's correctness and complexity quality results

A functional testing was conducted based on the strategy of black-box technique to ensure that JaQMeT has met expected requirements. Each predefined use case had been thoroughly checked its functionality. Use cases "Assess Program Correctness" and "Assess Program complexity" were tested by running it with predefined samples of program and samples of students' program that were collected from the pair programming experiment conducted in this college.

4.0 RESULTS AND CONCLUSION

JaQMeT can help the lecturers specifically to reduce workload on grading programming assignment. Therefore it helps to reduce time to assess pair programming program quality. Student's pair programming program quality which focuses on correctness and complexity can be graded by using JaQMeT. This result can be used to evaluate the effectiveness of pair programming practices. However JaQMeT also constrained by a few limitations such as it is a stand alone system and only can assess two quality factors; correctness and complexity for Java programming program. Therefore it is hoped that JaQMeT can be extended by using web-based technology as a web-based system might allow multiple accesses to the system. In addition a new tool that capable to check others program quality for other programming languages can be developed and hence integrate it with the tool developed in this study.

REFERENCES

Abounader, J. R. & Lamb, D.A. (1997). *A Data Model for Object-Oriented Design Metrics*. Retrieved May 26, 2005, from http://citeseer.ist.psu.edu/abounader97data.html.

Arisholm, E., Gallis, H., Dyba, T. & Sjoberg, D.I.K. (2007). Evaluating Pair Programming with Respect to System Complexity and Programmer Expertise. *IEEE Software*, 33(2), February 2007, 65-86.

Beck, K. (2000). *Extreme programming explained*. Massachusetts: Addison-Wesley.

Chu, H. D., Dobson, J. E. & Liu, I.C. (1997). *FAST-A Framework for Automating Statistic-based Testing*. Retrieved April 28, 2002, from: http://citeseer.nj.com/73306.html

Ciolkowski, M. & Schlemmer, M. (2002). *Experiences with a case study on Pair Programmming*. Workshop on Empirical Studies in Software Engineering, Rovaniemi, Finland.

CourseMarker's Research Page (2002). Retrieved March 31, 2004, from: http://www.cs.nott.ac.uk/CourseMarker/

DeClue, T. H. (2003). Pair programming and pair trading: effects on learning and motivation in a CS2 course. *Journal of Computing Sciences in Colleges, 18(5)*, 49-56.

Foxley, E., Higgins C. & Gibbon C. (1996). *The Ceilidh System A General View 1996 (on-line)*. Retrieved March 31, 2004, from: http://www.cs.nott.ac.uk/CourseMarker/more_info/html/Overview96.htm

Foxley, E., Higgins C., Tsintsifas A. & Symeonidis P. (1999). *The Ceilidh-CourseMaster System An Introduction 1999*. Retrieved March 31, 2004, from: http://www.cs.nott.ac.uk/CourseMarker/more_info/html/CMIntro.htm

Gehringer, E. F. (2003). *A pair-programming experiment in a non programming course*. Companion of the 18th annual ACM SIGPLAN conference on Object-oriented programming, systems, languages, and applications, October 2003.

Hanks, B., McDowell C., Draper, D. & Krnjajic, M. (2004) *Program quality with pair programming in CS1*. Paper presented at the Software Engineering, June 2004. Proceedings of the 9th annual SIGCSE conference on Innovation and technology in computer science education.

Matt, U. V. (1994). Kassandra : The Automatic Grading System. *Technical Report UMIACS-TR-94-59, CS-TR-3275*, University of Maryland. Retrieved April, 6, 2004, from: http://citeseer.nj.nec.com/matt94kassandra.html

Mawarny, M. R. and Rohaida, R. (2005). *Automating the Process of Measuring Complexity of Java Programming Assignment*. Final Report Research Faculty Grant. Universiti Utara Malaysia.

McDowell, C., Werner, L., Bullock, H. & Fernald, J. (2002). The effects of pair-programming on performance in an introductory programming course. Paper presented at the Proceedings of the 33rd SIGCSE technical symposium on Computer science education, February 2002, *ACM SIGCSE Bulletin*, 34(1).

McDowell, C., Werner, L., Bullock, H. F. & Fernald, J. (2003). *The impact of pair programming on student performance, perception and persistence*. Paper presented at the Software Engineering, June 2004. Proceedings 25th International Conference on Software Engineering, 3-10 May 2003, 602 – 607.

McDowell, C., Hanks, B. & Werner, L. (2003). Experimenting with pair programming in the classroom. Paper presented at Proceedings of the 8th annual conference on Innovation and technology in computer science education, June 2003, *ACM SIGCSE Bulletin*, 35(3), 60-64.

Pressman, R. S. (2001). *Software Engineering: A Practitioner's Approach* (5th ed.): McGraw-Hill.

Rohaida, R., Cik Fazilah, H. & Mazni, O. (2004). *Correctness Assessment Of Java Programming Assignment.* Final Report Research Faculty Grant. Universiti Utara Malaysia.

Williams, L., Kessler, R.R., Cunningham, W. & Jeffries, R. (2000). Strengthening the case for pair programming. *IEEE Software*, 17(4), July-Aug. 2000, 19-25.

Williams L. and Kessler R. (2003), *Pair Programming Illuminated:* Addison-Wesley.

Xenos, M., Starrinoudis D., Zikouli K. & Christtodoulakis D. (2000). *Object-Oriented Metrics – A Survey.* Retrieved May 10, 2005, from http://citeseer.ist.psu.edu/528212.html

Xu, S. & Rajlich, V. (2005). *Pair Programming in Graduate Software Engineering Course Projects.* Paper presented at the Proceedings of the 35th ASEE/IEEE Frontiers in Education Conference, 19-22 October.