MasterMind©: A Predictor of Computer Programming Aptitude

Torben Lorenzen and **Hang-Ling Chang**
Department of Mathematics and Computer Science
Bridgewater State College
Bridgewater, Massachusetts 02325 USA
<Lorenzen@bridgew.edu, HChang@bridgew.edu>

Abstract
For two semesters, the authors have tested CS1 introductory students on their ability to play the MasterMind© game at the beginning of the semester and compared those scores with in class programming test scores. The resulting correlations suggest that this game can be used as part of a computer programming aptitude test. This aptitude test could be used to advise potential students or employees about their probable success as programmers. Our survey of the literature yielded many programming aptitude tests with correlations lower than ours of 0.6; we are unique in using a game as an aptitude test.

Keywords: aptitude, predict, ability, game, mastermind

1. Introduction

A good computer programmer has many of the skills of a detective. Both professionals deal with a host of details and need to discover the overall pattern among those details that somehow form a related whole. It is often in a flash of intuitive insight that the detective solves the case and the programmer discovers how to code a program. Patience and perseverance are also required. Not all people have these attributes in abundance so some people learn how to program a computer better than others. It is easy to say that those who learn quickly and become successful programmers possess high computer aptitude. The hard task is to take a group of people who have no computer background and to predict which of those people have inherent computer aptitude and can become successful programmers.

All introductory computer science courses will admit students with no computer experience; some graduate schools will provisionally accept candidates with no CS courses or programming experience into a Master's program in CS; some industries will hire people without computer experience (because they are much cheaper to hire than degreed professionals).

In these three areas, if people can just enter without demonstrating computer science aptitude there will be a significant failure rate resulting in many students dropping an introductory course and many job applicants being let go. One could minimize this cost to businesses and educational institutions and loss in student self esteem by attempting to predict computer science aptitude in the applicants.

2. Traditional Aptitude Tests

Three computer science aptitude tests are used to screen programming applicants in industry. The Aptitude Assessment Battery Programming (AABP) test uses "simple calculations, complicated instructions and specifications, statements written succinctly without further explanation, and reasoning with symbols to predict aptitude." [2] Denelsky and McKee found a correlation of .4 between the AABP and student ranking. [2] "The Wolfe Programming Aptitude Test consists of only three problems. The second problem, moreover, depends on the first and the third problem depends on the two earlier ones. Each of the problems is more intricate than the preceding one." [3] The Wolfe-Spence Programming Aptitude Test "consists of eight work exercises which require the candidate to demonstrate logical ability, skill in interpretation of specifications and documentation clarity." [4] IBM no longer endorses the PAT (Programmer Aptitude Test) that it created. These tests cost several hundred dollars to administer per student and require three or more hours to complete.

Academicians have used high school performance on the ACT test and overall math proficiency [1] to predict student-programming success and have produced correlations less than 0.6.

3. A Game as an Aptitude Test

The authors' approach to predicting student programming success is to use the Hasbro MasterMind© board game that involves two players. The challenging player selects a pattern of four colored pegs. Since there are six possible colors, there are 1,296 different combinations possible. After each of the solver player's ten "guesses", the challenger gives a white clue for each peg that is the correct color and in the correct position and a black clue for each peg that is the right color but in the wrong position. In order to deduce the selected pattern, the player must use her analytic skills and logical deductive powers to create "guesses" that provide useful clues as to the nature of the solution. The authors hypothesized that creating useful guesses and correctly interpreting what the resulting clues meant about the nature of the solution was akin to debugging a computer program. To the degree that this is true, MasterMind© can be used as part of a programming aptitude test.

4. A Sample MasterMind© Game Session

The colored guess pegs can be Red, Green, Blue, Orange, Yellow and Cranberry.

Guess	Clues	Conclusions
RRGG	ww	RR?? xor ??GG xor R??G xor R?G? xor ?RG? xor ?R?G
BBOO	b	Solution has 1 Blue xor 1 Orange and it is not where the player guessed it was.
CCCC		Solution has (0 Cranberry) and (1 Yellow xor total Red + total Green = 3)
CYCB	wb	One Yellow and one Blue are in solution. One is in correct position; the other is not Orange is not in the solution. Candidates are YRGB, RYBG or RRYB. Guess one of the three.
YCCB	ww	Wasteful guess. Guess YRGB instead.
YRGB	wwww	The correct solution.

5. Methodology

During the first day of the CS1 class, the class is shown the actual game and the instructions from the game box are handed out to the class. The students then break into two teams – the challenging team creates a hidden solution and then loudly teaches itself to provide the correct clues to each of the other group's guesses. The other team teaches itself to come up with useful guesses. The professor stays out of the process unless incorrect clues are given. The teams then switch roles and the bedlam continues.

At the end of the class period the class is introduced to a computerized version of MasterMind© in which the computer assumes the role of the challenging player. The computer randomly picks a pattern and provides the black and white clues after each of the student player's "guesses". The game is available at http://www.imagiware.com/masterweb/

During the semester the students completed four programming tests that required the writing of a small module of code in class. The tests were zeroxed and the originals returned to the students. The student grade was determined by how many additions/changes the student needed to make in order for the program to run correctly.

6. Results

The first semester the students were told they would be tested on playing the game in the next class and if they did well, they would receive extra credit in the course. This first test showed the start of several games with three or four guesses and resulting clues for each game. The students listed all that they deduced about each game solution. The answers were somewhat akin to an essay and hard to grade and resulted in a correlation of .4 between the test and the average of the semester's in class programming exams. The MasterMind© test was administered and graded by the professor who did not teach the course. The results were not available to the teacher or students until after the semester ended.

The next semester the authors made two changes. The new test again provided the start of several games but enough clues were given for each game that it was then possible to specify the solution and the students attempted to do so. The test was given two weeks after class started which allowed motivated students to spend time mastering MasterMind©. The changes resulted in a correlation of .6 and that is as good or better than the correlation that traditional aptitude tests produce. The only problem was that the test was judged to be too easy since many students received perfect scores and most finished before class ended.

The next semester the authors applied for a local college grant to write a more difficult version of the second test and to administer it. Unfortunately the grant was funded late and the test was not administered until the middle of the semester at which point most of the poorer students had already dropped the course. The terms of the grant required the test to be taken on a voluntary basis (the earlier tests were required of all students) and many of the best students (not needing extra credit to achieve an A) opted to not spend the time to learn the game or to take the optional test. With such a small sample, the results were not conclusive. If any reader is interested in pursuing a similar study, the authors will make sample tests available to them.

References

[1] Bateman, C.R. Predicting performance in a basic computer course. In Proceedings of the 5th Annual Meeting of the American Institute for Decision Sciences AIDS Press, Atlanta, GA, 1973.

[2] Denelsky, G.Y., and McKee, M.G. Prediction of computer programmer training and job performance using the AABP test. Personal Psychology (1974), 129-137.
[3] Wolfe, J.M. An interim validation report on the Wolfe Programming Aptitude Test. Computer Personnel, 1977, 6(1-2), 1-2.
[4] WWW.Waldentesting.com

Endnote
This study was supported through a CART Faculty Librarian Research Grant from Bridgewater State College