Effectiveness of Teaching Metamorphic Testing

Kunal Swaroop Mishra
Department of Computer Science
Columbia University
New York, NY, 10027

Gail Kaiser
Department of Computer Science
Columbia University
New York, NY, 10027

***Abstract*— This paper is an attempt to understand the effectiveness of teaching metamorphic properties in a senior/graduate software engineering course classroom environment through gauging the success achieved by students in identifying these properties on the basis of the lectures and materials provided in class. The main findings were: (1) most of the students either misunderstood what metamorphic properties are or fell short of identifying all the metamorphic properties in their respective projects, (2) most of the students that were successful in finding all the metamorphic properties in their respective projects had incorporated certain arithmetic rules into their project logic, and (3) most of the properties identified were numerical metamorphic properties. A possible reason for this could be that the two relevant lectures given in class cited examples of metamorphic properties that were based on numerical properties. Based on the findings of the case study, pertinent suggestions were made in order to improve the impact of lectures provided for Metamorphic Testing.**

Keywords-Metamorphic Testing, Software Engineering Education, Unit Testing

I. INTRODUCTION

Although testing cannot assure the absence of errors in many situations, it still remains the most proven technique to improve credibility of a program. Many a times, majority of the test cases come out to be successful. These successful test cases are often thought to be useless and kept aside for regression testing or discarded. More often, test suite acts as the only form of formal specification since most specifications written in prose are as use cases or usage scenarios. Considering the fact that testing is a costly and labor intensive procedure, it is wise to make the best use of each test case. Since successful test cases are ignored most of the times, meaningful spec-related information contained in them remain untapped [1].

Metamorphic Testing can be used to extract useful information from the successful test-cases. Metamorphic properties of the successful test cases can be utilized to build follow up test cases, enlarging the test suite [1]. Execution of the metamorphic test cases may reveal bugs that the original test suite did not and provide sufficient confidence in the program, upon successful bug fixes, which conventional testing techniques like equivalence partitioning and boundary analysis might not have. We now present the basic concept of Metamorphic Testing (MT).

Concept of Metamorphic Testing

MT is a technique for generating follow up test cases from successful test cases. Consider a program p that implements a function f. Let the test suite for testing the program p be T={t1,t2,t3,…,tn}. If there is a test oracle present, then the output of p(t1), p(t2), …., p(tn) can be verified against the result of f(t1), f(t2), …. , f(tn). If the test results are successful, MT can be applied [1]. MT can be applied even when there is no test oracle and thus, we cannot know whether the original test results passed or not [5][7]. But using MT in the absence of a test oracle was not covered by the course, and thus is outside the scope of this paper.

MT generates follow up test cases T'= { t1', t2',…, tn'} based on the metamorphic property(s) contained in the function implemented in the program. A simple example of a function to which metamorphic testing could be applied would be one that calculates the standard deviation of a set of numbers. Certain transformations of the set would be expected to produce the same result. For instance, permuting the order of the elements should not affect the calculation; nor would multiplying each value by -1, since the deviation from the mean would still be the same (think about the numbers being "flipped" around the zero on the number line) [2].

Furthermore, other transformations will alter the output, but in a predictable way. For instance, if each value in the set were multiplied by 2, then the standard deviation should be twice as much as that of the original set, since the values on the number line are just "stretched out" and their deviation from the mean becomes twice as great. Thus, given one set of numbers, we can create three more sets (one with the elements permuted, one with each multiplied by -1, and another with each multiplied by 2), and get a total of four test cases; moreover, given the output of only the first test, we can predict what the other three should be [2].

The above examples can be thought of as numerical since they are derived from numerical properties directly. An example of a not so directly numerical MT would be as follows; if you search the corpus "Romeo and Juliet" for the word Romeo as first test case, you will find certain lines. If you then search with "Romeo and foo" where "foo" is not present in the corpus, you will find zero lines. Further, if you now search with "Romeo or foo" you will find exactly the same lines as for the original case. This particular example does not have an evident mathematical/numerical/set property in it yet a metamorphic property could be derived. It could be argued that the property derived is still mathematical but the problem statement did not have a direct connotation for it.

As part of this experiment, we did not compare the performance of students in conventional testing techniques with their performance in metamorphic testing. The paper only critiques on how well the students understood metamorphic testing.

In the next section, we will discuss the rationale for this experiment. After that we will discuss the setup for the student project in ***Case Study Group*** section. Then we will take a look into the various students' projects in ***Projects' Description & Metamorphic properties*** section. After that we will discuss our findings in ***Observations*** section and then put forward our suggestions in ***Suggestions*** section. We will then discuss about our future work in ***Future Work*** section and finally conclude with ***Conclusion***.

II. Experiment Rationale

The rationale of this review, of Metamorphic Testing conducted in various student projects, is to understand the effectiveness of teaching Metamorphic Testing Concept in a senior/graduate software engineering course class room setting and to suggest future changes to the mode of the lectures based on the central findings of the case study. MT can be can be used both at the unit (subroutine) level and at the full system level. In this experiment, students were asked to apply MT at the unit level. The following research questions are the central aspects of the case study:

1) To what extent were the students successful in finding metamorphic properties for testing in their respective projects? The measure of students' success depended on whether we could find metamorphic properties in the project that the students could not find.

2) What were the reasons that contributed to the performance of the students on finding MT properties? The reasons that could contribute to the students' performance could be to do with the quality of the lectures and materials provided in class. It could also be how important the students perceived the concept to be. We delve deeper into this in the Observations section.

3) What were the types of MT properties that the students found in their projects? Mostly the students found the conventional numerical metamorphic properties that were derivatives of the examples delivered in class or contained in the provided materials. In this experiment, first author was an impartial observer and was in no way involved with the course offering by the second author, and it was the first author who looked for metamorphic properties in the students' projects and evaluated whether or not the students had found them. As an impartial observer, the author could find various unconventional metamorphic properties that dealt with numbers and sequences or rankings. We shall discuss this further in the Observations section.

4) What were the main causes for the students to find the properties that they did? As discussed in (2) above, the materials provided to the students did not contain broader set of examples showing metamorphic properties. This may be a reason why the students could only find simple, basic numerical metamorphic properties which were similar to the ones contained in the material provided to the class. Perhaps, these examples were so basic that the students could not dig deep into the concept of Metamorphic Testing and could not find varied metamorphic properties. Further, some of the students did not understand the concept and reported properties that are not metamorphic.

It is worth mentioning that there were no such projects encountered where students could define some or all of the metamorphic properties but were not able to come up with corresponding test cases, i.e., the derivations of the new test cases.

III. Case Study Group

The projects under consideration in this case study are the projects done by senior/graduate students as part of the curriculum of Advanced Software Engineering course that was imparted at CS department, Columbia University to both Undergraduate and Graduate students in Fall,2011. The course is designed to impart knowledge about software lifecycle from the viewpoint of designing and implementing N-tier applications with special emphasis on quality assurance [6]. The projects were carried out by group of four students. There were a total of seventeen projects and all the seventeen projects were considered for this case study. In the following section, these projects are discussed at a greater depth.

In order to equip the students with an insight of MT, there were two lectures provided in class. The first lecture [3] was an introductory note about MT. The subsequent lecture [4] delved deeper into the fundamentals of Metamorphic Testing with citations of examples demonstrating metamorphic properties. The examples were of numerical type as, in an introductory class like this, such examples would make it simpler for the students to understand metamorphic properties. Some of the examples discussed in those lectures were:

Suppose we have a test case that finds the sum of N numbers and let's assume that sum to be S. The test case exhibits the following metamorphic properties:

1) Permute the N numbers and the sum would still be S.

2) Add 2 to each number and then the sum would be S + (N*2).

3) Multiply 2 to each number and then the sum would be 2*S.

4) Include another number A in the set (making the set N +1) and then the sum would be S + A.

5) Exclude a number F from the set, making the total elements of the set N-1. The sum now would be S – F.

Further, the students were assigned a paper [5] written by Christian Murphy, Kuang Shen and Gail Kaiser for a follow up reading on Metamorphic Testing which contained examples of metamorphic properties similar to the ones mentioned above.

IV. PROJECTS' DESCRIPTION & METAMORPHIC PROPERTIES

All the seventeen student projects, submitted as part of the course, are now discussed with importance being given to the key idea of the project along with the metamorphic properties discovered by the students. First author's comments about each of the projects are also mentioned so as to give the readers an idea about how did the projects perform with respect to finding and testing metamorphic properties.

Project 1

The project intended to build a social site that would discuss and help in discovering restaurants based on the ethnicity of the cuisine served in the restaurants. The site would be seeded with a knowledge base about various ethnic cuisines, so that reviews of restaurants, in addition to being rated by users, could be rated in a slightly more objective way. The students also planned to acquire data from MenuPages, or some similar site, to include information about the restaurants' dishes and the authenticity of those dishes along with the reviews and any other information.

Metamorphic properties:

1) If a given restaurantId and page number 1 returns a certain view, then the same restaurantId with any page number <= 0 should return the same view.
2) If a given restaurantId and a given page number returns a valid view, then the same restaurantId with any page number should return a valid view.
3) For a given restaurantId, if page number X and page number X+1 return the same view, and both views are valid, then for any integer Y > X + 1, the same valid view should be returned.
4) If a given restaurantId and page number 1 returns a failure message, then that restaurantId and any page number (positive, negative, or zero) should return the same failure message.
5) For a given restaurantId, if the page number 1 returns a valid view, then the empty page argument should return the same valid view.
6) If a given restaurantId returns an invalid restaurantId error page redirect for a given Review, then it should return the same error page for all Reviews.
7) If a given restaurantId and a given Review object with weight attribute X returns a given valid redirect page, then the same valid redirect page should be returned for any weight attribute Y.
8) If a given restaurantId and a given Review object with non-Null text attribute X returns a given valid redirect page, then the same valid redirect page should be returned for any non-Null text attribute Y. If a pair of input arguments create a new Review r with a given timestamp X, then the same pair of input arguments one minute later should create the same Review r with the timestamp X + 1 minute.

Comments:

The metamorphic properties found by the students, as part of the project, are in accordance with what the author thinks. Besides, the author could not find any extra metamorphic property. The properties listed also seemed very interesting as they were not exactly numerical properties. However, it could be argued that they are derivatives of numerical metamorphic properties.

Project 2

The students intended to build a hotel management system that would help users to search for hotels, tariffs, etc. and book rooms as per their stay. The users of the site could search for hotels based on tariff/room/availability. The user could book available rooms as per his/her criteria.

Metamorphic properties:

1) Book a room for duration 11th - 15th. Increment the start and end date by 2 days and book the room for duration 13th - 17th makes a booking of 5 days in both the cases.
2) Book a room for duration 11th - 15th. Now book the same room for date 16th. Check that the date 16th is included in the reservation.
3) Book a room for duration 11th - 15th. Now cancel the booking for one day i.e. 15th. Check that the date 15th is excluded from the reservation.
4) Book a room for 3 durations in any random order i.e. 21st-25th, 10th-15th, 16th-20th. The room is booked for duration 10th to 25th.

Comments:

The author could not think of any additional metamorphic properties. The students could figure out these metamorphic properties because of the fact that the project's core idea was based on numerical properties which are easy to associate with Metamorphic Testing.

Project 3

The students designed a multi-player interactive game. Users could log on to the system to initiate the game. Once logged on, a player would be given an identity randomly - killer, medic, or lay person. Each player would only know his/her own identity when the game starts. The game is essentially a battle; the killer against the lay people and medics. The battle is fought while identities are not revealed and ended when identities are exposed. The lay people would win when the identities of all the killers have been revealed and the killers would win when the only people left in the game are killers. The goal of the medics in the game is to heal lay people that have been killed by the killers.

Metamorphic properties:

None Identified.

Comments:

The students perhaps could not identify any metamorphic property. Hence, they did not mention any. One metamorphic property, evident from the project description is the way assignment of role is handled. It is basically *request _number mod 3*. Suppose first person logs in; she will be assigned killer (since 1 mod 3 = 1). This could have been pointed out by the students. Either, it was hard for them to understand the property or they did not understand MT.

Project 4

The project was an implementation of a board game. The users would attend this small game by putting pieces of different shapes on the square chessboard. In this game, the users were required to use their own strategies to put their own pieces.

Metamorphic properties:

1) Encryption of passwords using encoding algorithm.

Comments:

The property listed is not really a metamorphic property. But position of a piece on a board is a metamorphic property, i.e., position of a piece on the board now (after some game time has elapsed) as compared to position of that piece on the board during the start of the game is a metamorphic property. Further, the position of all the pieces on a board could be identified as the state of the board and the current state of the board (after some game time has elapsed) as compared to the state of the board during start of the game could be a metamorphic property as well. Students failed to point out these properties. Again, it may be the case that the students did not understand metamorphic properties well.

Project 5

The students implemented a gang game. In the game there are several gangs in the world. Each gang has its own base and members, while the final mission is to conquer other gang's base and be the greatest gang in the City. Every player in the world belongs to one gang, and has his/her own profile, weapons and money. Players could cooperate with their friends in the same gang to rob banks, collect protection money or join other exciting missions. After successfully finishing these missions, the system would provide user with money as awards, which would enable a user to upgrade weapons and base. When one gang is powerful enough, it could try to conquer other gang's base and expand its territory.

Metamorphic properties:

1) Player with weapon whose power is 20 can attack the guard whose health is 100 and make his health 80.
2) Player whose credits are 200, when buys a 100 credits weapon his credits become 100.

Comments:

What has been listed as a metamorphic property is not exactly a metamorphic property. Rather the property should have been that when the guard with health 100 is attacked by a weapon of 20 and a weapon of 30 his health is 50 and when the same guard with health 100 is first attacked by weapon of 30 and then by weapon of 20, his health is 50 again. The students could have also considered where both test cases in the pair start from the same state, i.e., both test cases have initial health of H. So, at health H when attacked with weapon of W if remaining health is X, then it should be that at the same health H when attacked with weapon of W+10 remaining health should be X-10. Further, if at health H when attacked with weapon W results in remaining health as X, then at health H-10 when attacked with weapon W remaining health should be X-10 (provided that original H > 10 since we cannot go below 0).

Project 6

The project students implemented a social networking site that aimed at automatically providing users with customized social activities and like-minded friends. When web-users would be browsing this site, joyful mingles and like-minded friends would be immediately recommended based on the knowledge of user's personal information, which would be collected from user's profile or pop-up questionnaires.

Metamorphic properties:

1) The metamorphic property of the test unit was that you could do permutation and combination of the original test cases and generate new test cases.

Comments:

The author does not think that the property listed is a metamorphic property. What could be a metamorphic property is, the recommendations done for users with similarities. The order should not matter but the recommendations should be same for all similar users.

Project 7

The students implemented a fantasy football league where there were team owners who managed the teams and there was a commissioner who controlled the creation of the league. Before the start of the professional football season, a league Commissioner would log into the fantasy football application and, using an administrative console, would create the fantasy football League in which a group of Team Owners would play. Naturally, the league Commissioner would also be responsible for creating the Team Owner accounts that each person playing in the league would use to log into the application. Through a lottery, each Team Owner would be assigned a slot order number in which they would draft real National Football League (NFL) Players. Once the draft is completed and Team Owners would finalize their starting rosters, after each week of the NFL season scoring for each fantasy team would be calculated whereby Players would earn their team points based on their performance in their weekly games.

Metamorphic properties:

1) The metamorphic property was the team's counter per position player, i.e., if a player is benched or started, the team counter should change accordingly.

Comments:

The property listed is correct as per the author. But the students could have also identified another metamorphic property; on randomly adding/removing players from certain positions, do the credits add/reduce correctly? Since the project has a strong numerical underlying, the mentioned property could have been identified. For instance, a player p1 at position A has a value of X and a player p2 at position B has value of Y. If p1 is removed followed by removal of p2, the value of credits should T-X-Y (where T is the team's total credit). Alternatively, if p2 is removed followed by p1's removal, the total credit should still be T-X-Y. Further, if total credit of the team were T-10, removal of p1 followed by removal of p2 should have resulted in total credit of T-X-Y-10 as compared to T-X-Y when the team's initial credit was T.

Project 8

The project implemented is a shooting based game where users are challenged to shoot as many targets as possible in the allotted time. Different targets are worth different points. The duck which is the most challenging to shoot is worth the most at 20 points. The army men are worth the second most at 10 points. Lastly the plane is worth the least amount at 5 points. As users select games, they are assigned to the next available game room. Each game room would close when it has reached the maximum amount of players and another room would be opened.

Metamorphic properties:

1) Changing a valid user ID to another valid user ID should yield correct result.
2) Changing invalid user ID to another invalid user ID should throw exception in both cases.
3) Changing an invalid user ID to a valid user ID should go from yielding an error to correctly working.
4) Changing a valid user ID to an invalid user ID should go from correctly storing information to yielding an error.

Comments:

The students identified most of the properties that the author could. But the author could identify three metamorphic properties which are not listed by the students. They are 1) the list of top scores won't change if a game is played and the score generated in the game is not greater than the tenth score on the list. 2) Similarly, for list of top users as well afore mentioned metamorphic property could be used. 3) Since this is a shooting game with points involved, the students could have used the following metamorphic property; Assume a shooter starts with zero points and first shoots a duck to get 20 points and then shoots an army man to get 10 points bringing her total point tally to 30. Imagine another shooter playing in another game room starts off at zero point and first shoots an army man worth 1 point and then shoots a duck worth 10 points amassing a total of 30 points as well, i.e., irrespective of the order of targets, if two players have shot down the same set and number of targets, they should have same total points. Similarly, if a player had 5 points then shooting a duck and then an army man in that particular order would have taken his points tally to 35 as compared to 30 points when the player started with zero points.

Project 9

The students implemented a stock paper trading tool with integrated social networking functionality. The project combined concepts of a regular online brokerage service with collaborative aspects of social networks, which included friends, posting comments on the profiles of other users, private messaging and rankings of individual traders.

Metamorphic properties:

1) Buying 10X shares would be 10 times costlier than buying x shares.

Comments:

The property mentioned is a straightforward metamorphic property that the students would have found in the materials provided in the class. In fact, it is not really stated properly; a metamorphic property always involves a pair of test cases. The stated property is only one test case. What could have been an identifiable metamorphic property is a sequence of transactions that lead to a same amount. For instance, let's assume a buyer has amount A. When the buyer buys shares worth a total of amount X, then the buyer is left with A-X(and A-X > 0). Now when the buyer buys shares worth a total of amount Z, then the buyer is left with A-X-Z (again we assume A-X-Z > 0). If another buyer with total amount A buys shares worth Z first and then shares worth X, he should have leftover amount of A-X-Z as well. Further, since there is a social status sharing wall, it could be tested that suppose the wall holds "n" messages and if there are "n" new updates then the user's wall would hold the new "n" messages and the old "n" messages would vanish. The ranking page can also exhibit metamorphic properties similar to the ones mentioned in Project 8's comments.

Project 10

The students implemented a multi-player economics game where a player had either a seller or a buyer role. The goal of the game is to obtain as much money as possible. As a user accumulates money, he would be awarded badges or upgrades or some form of flair. The game is continuous as people log on they would be placed in whichever role has fewest people. The application would keep track of each user's current role, their accumulated cash, and their current buy/sell price.

Metamorphic properties:

1) The change in the amount of money that the buyer has should be exactly equal to A - P. As this is linear in A, any metamorphic property of linear functions applies to the change in buyer money. For instance, if we increase the allowance by a certain amount D, we will expect the resulting change in money to also increase by D. If we multiply both A and P by the same constant K > 0, the resulting change in money should also change by the same factor K.
2) The change in the amount of money that the seller has should be exactly equal to P - C in every case, except for the case when P <= 0. Therefore, the same metamorphic properties apply as in the previous unit test. For instance, increasing the asking price by D should result in an increase to the change in money of the same D. For P <= 0, all values should result in the same error and therefore we cannot extract any metamorphic properties.

Comments:
The metamorphic properties identified by the students are mostly in accordance with what the author thinks. However, the students could have identified more properties. Another metamorphic property could be if a buyer or seller did back to back transaction, the left over amount should be same irrespective of the order of the transactions. Similar metamorphic properties have been described in the comment section of Project 9, 8, 7 and 5.

Project 11
Students implemented a game board with several spaces along the margin, forming a loop. Spaces indicated different properties, such as a building, a hospital, a chance room, etc. Two to three players would start one game with a specific amount of money for each at initialization of the game. All players would start from the low-right corner check of the board, and go clockwise in turns. In each turn, each player would throw the dice and advance his/her piece around the board to the corresponding number of squares.
Metamorphic properties:
1) A player with \$1000 wants to upgrade a property cost \$500. He/she has \$500 left.
2) Player A with \$1000 robs player B with \$1000 of \$500; after robbing, A has \$1500 left, and B has \$500 left.
3) A player with \$1000 wants to buy a property cost \$500. He/she has \$500 left.
4) A player with \$1000 should pay the rent \$500. He/she has \$500 left.

Comments:
The property mentioned is a straightforward metamorphic property that the students would have found in the materials provided in the class. But again, these aren't really stated properly; a metamorphic property always involves a pair of test cases. The stated properties come across as single test cases. An identifiable metamorphic property could be one where two players with same starting amount did N similar transactions but in different order and ended up at equivalent leftover amount after those N transactions. Similar metamorphic properties have been described in the comments section of Project 9, 8, 7 and 5. Further, there could be more metamorphic properties like rolling a dice twice and seeing where a player lands could be one. If the dice throws out the same numbers but in different order, the player's position should not change, given the player started from the same starting position. The position of all the pieces on a board could be identified as the state of the board and the current state of the board (after some game time has elapsed) as compared to the state of the board during start of the game could be a metamorphic property as well (as discussed in comments section of Project 4).

Project 12
The students created a card game called hearts. New user would run the client application to connect to the server, and would register a new ID in the login page and then log into the game lobby. After entering the lobby, the user would choose which table to join. The user would join one table and the lobby page would transfer to a READY/EXIT page, as soon as 4 people are inside. The user would click READY along with other users to begin the game. After the game, the user would get his rank and score.
Metamorphic properties:
Not clearly mentioned.
Comments:
The students did not mention any metamorphic property clearly. The testing of ranking and scoring functionalities would have metamorphic properties. Similarly, top ranks list would also have metamorphic property as discussed earlier.

Project 13
The students created an online version of the popular casino game BlackJack.
Metamorphic properties:
1) Changing the order or arrangement of the cards in your hand does not change its value.
2) Many hands are semantically equivalent. Hands are measured by their value, and many hands can have cards that end up adding up to the same value as other hands. For example, an 8 of hearts and 7 of spades is "semantically equivalent" to a 9 of spades and a 6 of hearts, whose value is also 15.
3) Adding a card to hand adds the value of that card to your hand. Ex. If you have a hand worth 10, let's say a 5 of hearts and another 5 of clubs, and you hit to gain a 10 of hearts, you now have a hand worth 20.

Comments:
The students could find all the metamorphic properties that the author could find. A possible reason for this could be that the project was based on numerical properties.

Project 14
The students tried to create a social network based man page. The project aimed to provide a meaningful user interactive man page that would allow users to view the comments, useful tips and practical examples of the commands.
Metamorphic properties:
None Identified. They assumed that metamorphic properties are applicable only to numerical input/output.
Comments:
The students here thought that metamorphic properties are only for numbers and could not be extended beyond that. The metamorphic property existing in their project is related to their functionality that searches for top searched command. Now if a user searches for a command that is less searched than the least top searched command, the list displayed for top searched commands should not be altered. Since the students did not understand Metamorphic Testing correctly, they could not identify this property.

Project 15
This was a question game: users would compete with each other to solve series of questions in a short amount of time. When a game would start, people would answer n questions in a row and they would be rated on the number of good answers. Everything would be done under a strong time constraint and should look as smooth as possible. So, as soon as a user would answer he would see the result and then would be able to go and join a new game.
Metamorphic properties:
1) If we shuffle the submission order, the score gained from right answers shouldn't change (though the bonus point will change) and the result screen ranking should be the same if who gets the bonus point can't change the ranking.
2) If we change the answers but keep the submission time the same, the bonus point should award to the same user.
3) If everyone submits n second earlier, the bonus point should award to the same user.
4) If everyone answers one more question right (everyone's score++), the ranking should remain the same.

Comments:
The students found all the properties that the author could find. The students also found some interesting properties related to time.

Project 16
The students implemented a basic competitive typing game. This typing game would enable two players to play head to head in a game of "who-can-type-this-arbitrary-string-the-fastest". The implementation would be played remotely by multiple users over the internet. The game would be implemented as a client/server application; the server would maintain a database of each user's login, their overall statistics and the possible strings that could be used in a round of the game. There would also be an admin maintaining the application.
Metamorphic properties:
1) On addition of new Words per Minute the user’s old Words per Minute should become the weighted average of the old WPM and the new one.
2) If you type some multiple times of some number of words in a given time then that should multiply the resultant words per minute by said multiple.

Comments:
The students identified most of the properties that the author identified. But there were more properties that the students failed to identify such as if you click on “view game history” after playing an extra game, it should display n+ 1 games, where n is the number of games being displayed before playing that extra game. The top ten players’ list can also have a metamorphic property that has already been discussed earlier.

Project 17
The students designed an online poker game.
Metamorphic properties:
1) Multiplying the small blind by any positive integer should not change its equivalence class.
2) Multiplying the small blind by any negative integer or zero should change its equivalence class (except when small Blind = 0)
3) Multiplying i by any positive integer should not change its equivalence class.
4) Multiplying i by any negative integer should always change its equivalence class (except when i = 0)
5) Multiplying i by any multiple of 4 should not change the method output.
6) abs(seatId) % 4 is a valid input for the id of a seat.

Comments:
The students identified all of the properties that the author identified as well. This could be possible because of the fact that the project had a strong numerical underlying. The students defined an equivalence relation for the game from which they derived the equivalence class for the game.

V. Observations

Effectiveness of the Metamorphic Properties Lecture
In order to appreciate the inference drawn out from the findings, the projects were divided into three classes; the projects that found all the properties found by the first author were placed in Class A. The projects that could determine some of the metamorphic properties but not all were put in Class B. The projects that could not understand what metamorphic properties are and reported some false metamorphic properties that did not exist in the project or did not report any metamorphic property at all were placed in

Class C. TABLE I shows the total number of projects in each class

TABLE I.

Class	**Number of Projects**	**Percentage of Projects**	**Project IDs**
Class A: found all Metamorphic Properties	5	29.41	1, 2, 13, 15, 17
Class B: found few Metamorphic Properties	7	41.18	5, 7, 8, 9, 10, 11, 16
Class C: didn't understand/reported false Metamorphic Properties	5	29.41	3, 4, 6, 12, 14

From Table1 it can be inferred that the most of the students in the class either did not understand the concept of metamorphic testing completely or understood metamorphic testing partially. Total number of projects in Class B and Class C are 12 which are about 70% of the total projects. In light of this observation, it can be inferred that most of the students taking the class could not grasp the notion of metamorphic properties in testing completely. Although unlikely, it could also be possible that the students understood the concept of metamorphic testing but did not understand how to apply the concept to their own projects, and if asked to apply to some simple program rather than their own project, might have been able to do so (but the course did not address this).

There can be several contributors for the above mentioned shortcoming. (1) *The lectures provided probably could not reach the students in the form that the instructor expected*. One possible factor for this could be that the materials provided in class could be inadequate. The examples given in the class might have been so simple that the students could not use it as a base so as to extrapolate advanced/different metamorphic properties. It can also be argued that the examples provided in those lectures were deliberately simplified in order to aid the students' understanding of the topic. Another reason could be the inability of majority of students to appreciate the concept of MT when taught in the class. Since MT is relatively a lesser known topic in unit testing, it might be the case that the majority of the students could not relate to MT as they did not have prior familiarization with MT (2) *Overall acceptance of MT by the students*. Since MT was included for the first time in the curriculum, it could have been the case that the students would have associated less importance with MT than other conventional testing approaches. Further, the students might not have taken a liking for the concept as compared to other testing concepts taught as MT is not as straight forward as the other concepts are. And as already mentioned, since MT is not taught in other lower level classes, students might not relate to it as readily as compared to other topics with which they have some familiarity.

Types of Metamorphic properties identified

Most of the metamorphic properties identified in the projects were of numerical type. The properties identified were either similar to the examples given in the class or were derivatives of the same. Most of the students were able to identify these example related properties. But they failed to identify the other metamorphic properties in their projects. For example in Project 8, the students found various metamorphic properties related to the user ID as discussed in the above section. But they did not notice the metamorphic property in the functionality that would list the top scores and top players. The metamorphic property that lies here is that if player plays a game when the top ten scores is already populated then after playing the game if the score of the player is less than the least score in the top scores' list, the player's score will not be accommodated in the list and hence, the list won't change. Similar property lies in the top players' list as well.

In Project 5, the students identified a metamorphic property as "Player with weapon whose power is 20 can attack the guard whose health is 100 and make his health 80". Though the students manifest that they have some notion of metamorphic properties, they do not put across the usage of the property clearly. Metamorphic Testing is used to exploit information contained in test cases. The above property is basically a specification that the student should have a tested. It is not a metamorphic property in its current form. The metamorphic property would be that when a guard of health 100 is attacked by a weapon of power 20 and then is again attacked by a weapon of power 30, the guard's health is 50. The same guard with health 100 is when attacked by a weapon of power 30 and then is again attacked by a weapon of power 20, his health is 50, i.e., in both the cases, the health of the guard is the same. Another way of testing this metamorphic property would be to see that when a guard of health 100 is attacked by a weapon of power 20 and when the same guard of health 100 is attacked by a weapon of power 2 * 20 = 40, the difference in remaining health of the guard after execution of both the cases should be equal to the difference of the power of the two weapons used in both the cases.

Some of the students probably could not comprehend the concept and mentioned properties that were not metamorphic. For example in Project 3, the students mentioned in the report that they were testing metamorphic properties but the properties are not mentioned in the report. One possible metamorphic property in their plan could have the way players are assigned roles. Since the program assigned roles by determining the number of players modulo four, it could be tested that a player's role does not change if the total number of players are p (where his role would be p modulo 4) and when the total number of players are p+4 (where his role would be (p+4)modulo 4).

The students of Project 14 assumed that metamorphic properties are only applicable to numerical input/output which is a gross misinterpretation of the concept. However, it should be noted that this assumption is understandable given the lecture materials shared in the class. They had a metamorphic property regarding the functionality top searches similar to the functionality of top scorers in Project 8 which is discussed above.

In Project 6, the students assumed that permutation class of metamorphic properties can be applied to test cases (not the input) and devised a metamorphic property around this. This is again an instance of incorrect understanding of metamorphic testing. This could be contributed to the problem of poor English of the students, i.e., what the students meant were the inputs of the test cases.

The students that could identify all the known metamorphic properties in their projects could do that because the projects had an arithmetic background. Thus, they could relate to the examples of MT provided in class in order to derive the properties that they did. For example in Project 2, the students found out all the project related metamorphic properties which are discussed in the above section. Similarly, students of Project 13 and Project 15 could find all the metamorphic properties because their projects had a numerical/arithmetic base.

However, there was only one team (Project 1) whose students found very unique metamorphic properties. But this could be termed as an aberration in the metamorphic property finding trend of the class. For instance, "*if a given restaurantId and page number 1 returns a certain view, then the same restaurantId with any page number <= 0 should return the same view*" was a metamorphic property identified by the students doing Project 1. This property is trivial to understand. But the property does not come across as an obvious metamorphic property that can be identified by students who do not have prior familiarization with Metamorphic Testing. Similarly, the metamorphic property "*If a pair of input arguments create a new Review r with a given timestamp X, then the same pair of input arguments one minute later should create the same Review r with the timestamp X + 1 minute*" is another unique property which shows the impact of time on the project. The property manifests that a Review created with certain arguments will not change when created at a later time if the arguments are repeated. This property also corroborates the fact that these students could find properties that were different from the examples provided in the class as there was no such example given in the lectures or the materials provided.

After analyzing the nature of the properties identified during this exercise, it can be concluded that almost all the properties identified were of numerical type. This could be because of the MT examples given in class. Since the lectures delved into explaining MT through numerical examples, students could relate to numerical metamorphic properties better. Further, the students who could identify all the metamorphic properties that their project contained could do so because the project had an arithmetic base.

There were also a few students who did not understand the concept of MT completely or misinterpreted MT and ended up in identifying certain properties that were not metamorphic. This could be because either the students did not concentrate on the lectures regarding MT or the overall lectures fell short in explaining MT to the students. For example, students of Project 3 did mention the term Metamorphic Testing in their testing plan but did not mention any metamorphic properties. One possible metamorphic property in their plan could have the way players are assigned roles. Since the program assigned roles by determining the number of players modulo four, it could be tested that a player's role does not change if the total number of players are p (where his role would be p modulo 4) and when the total number of players are p+4 (where his role would be (p+4)modulo 4). Alternatively, students of Project 14 assumed that metamorphic properties are only applicable to numbers (probably because of the examples given in class) and did not find other metamorphic properties that existed in their program. For example, they have a functionality that searches for top searched commands. If there are N top searched commands (let the commands be named 1,2,….N where 1 is top top-most searched and N is least top-most searched), then if a command M is searched less number of times than N, the list of top most searched commands should not be altered. It is worth noting that this is still numerical, in a sense, even though the inputs are not actual numbers.

VI. Suggestions

It can be clearly seen that the lectures fell short of its intent as almost 70% of the students either failed to understand the concept of Metamorphic testing or misinterpreted it entirely. Hence, it is safe to suggest that the method of teaching or lectures delivery attempted needs to change. Metamorphic Testing lectures may be made more examples oriented than what it is in its current form. This would definitely give the students a better understanding of MT. The examples cited in the observation section can be looked upon to derive such examples. Further, if the course aims at making students understand the concept of metamorphic testing with greater priority, assignments can be handed out to the students where they would be asked to find basic numerical metamorphic properties as well as unusual metamorphic properties like page contents' limit testing wherein if a page can hold N items, then N+1 items would need another page. Alternatively, any number of items less than N would not affect the total number of pages. Similarly, top scorers functionality's (discussed in above section) metamorphic property where in showing top 10 scores would show same scores in same order if new user did not score more than least score on the top 10 list can also be included as an example. Even if the students cannot identify all the properties, discussion of the solutions of these assignments would inculcate a better understanding of metamorphic testing among the students.

Further, it was observed that the majority of the properties identified by the students were of numerical type. This is also because of the fact that the students, during the lectures, were given examples of such basic numerical metamorphic properties to aid their understanding. However, this did not give the students a wholesome perspective about metamorphic properties and their testing. It would be very beneficial if different sets of metamorphic properties' examples were included in the training set as a wider precedent base would help students identify different metamorphic properties. The approach described in the above paragraph would also suffice to address this problem.

VII. Future Work

Based on our findings, we have two targets for the future. The short term goal is to overcome the short coming of the materials provided in the class on Metamorphic Testing so that the students are well equipped to find these properties in their own projects.

In order to achieve this, the example base provided in the class' slides need to be changed. The base examples need to have a wider variety of metamorphic property types so that the students get a large precedent to follow and learn from. Another approach (as discussed in ***Suggestions***) could be to assign homework/assignments with respect to only metamorphic property. This would have two advantages; 1) Students would give more priority to Metamorphic Testing. 2) Students would understand the concept better. How well the students understand Metamorphic Testing depends on the assignments provided. Hence, it is important to come with an assignment structure that is robust enough to cater to the curiosities of the students. Hence, the short term goal is to enhance the material provided in class and come up with full-bodied assignments for the students and we aim to achieve this over the summer of 2012.

The long term goal is to implement the enhanced material and assignments in a class room setting over the fall of 2012 (when the course would be offered again) and gauge the success achieved by the implementation of the revamped approach for teaching Metamorphic Testing in a class room setting. We also intend to come up with introductory level materials for teaching Metamorphic Testing that would aid instructors at other schools as well.

Future instructors who intend to work on this subject can look at the results obtained from our future tests and try to improve the process if it is deemed feasible or necessary. Further, we believe that there are other classes of metamorphic properties yet to be identified. That can be an interesting route to foray into. It would also be an interesting challenge to build an educational game that focuses on teaching concepts of metamorphic testing to students.

VIII. Conclusion

This article has introduced the results of teaching the concept of MT, primarily as a concept to increase the test suite and thus, increase the chances of catching a bug, in a class room environment. The results show that concept of MT was not fully understood by the majority of the class. Further, the article analyzes the results obtained as a result of the case-study of various student projects. The observations show that the material provided in class was inadequate. Further, the students did not give considerable importance towards understanding the concept of MT.

Besides the findings, we also suggest reforms in order to achieve greater success in teaching MT to the class with the help of enhanced material and assignments. We aim at making the changes to the material by summer 2012 so that we can apply these changes to test again when the course would be again offered in fall 2012.

IX. Acknowledgment (Heading 5)

The Programming Systems Laboratory is funded in part by NSF CCF-1161079, NSF CNS-0905246, and NIH 2 U54 CA121852-06.

X. References

[1] T. Y. Chen, S. C. Cheung, and S. Yiu. Metamorphic testing: a new approach for generating next test cases. Technical Report HKUST-CS98-01, Dept. of Computer Science, Hong Kong Univ. of Science and Technology, 1998.

[2] C. Murphy and G. Kaiser. Improving thedependability of machine learning applications.Technical Report CUCS-49-08, Dept. of ComputerScience, Columbia University, 2008.

[3] http://ase.cs.columbia.edu/confluence/download/attachments/786582/20Oct11.ppt?version=1&modificationDate=1318453043000

[4] http://ase.cs.columbia.edu/confluence/download/attachments/786582/MetamorphicTesting-Columbia-17Nov2011.ppt?version=1&modificationDate=1321497303000

[5] Christian Murphy, Kuang Shen and Gail Kaiser, Automatic System Testing of Programs without Test Oracles. International Symposium on Software Testing and Analysis, July 2009.

[6] http://ase.cs.columbia.edu

[7] Christian Murphy, Kuang Shen and Gail Kaiser, Using JML Runtime Assertion Checking to Perform Metamorphic Testing in Applications without Test Oracles. 2nd IEEE International Conference on Software Testing, Verification and Validation, April 2009