In P. Romero, J. Good, E. Acosta Chaparro & S. Bryant (Eds). Proc. PPIG 18 Pages 22 - 37

# Testing Programming Aptitude

Saeed Dehnadi

School of Computing Middlesex University, UK
s.dehnadi@mdx.ac.uk

**Abstract.**

An initial cognitive study of early learning of programming aimed to extract experimental test data to establish novices' understanding process has been carried out by us [1]. This empirical study was inspired by the notion that different people bring different patterns of knowledge in any new learning process, and demonstrated that how each student tackles the problem in a different way based on their mental model. The initial study suggests that success in the first stage of an introductory programming course is predictable, by noting consistency in use of the mental models which students apply to a basic programming problem even before they have had any contact with programming notation, but the consistency/inconsistency measurement was somewhat subjective. In this paper I present an objective marking method which hope will lead us to more precise and more finely-graduated predictions. This method is being trialed in at least one experiment, and we hope that by the time of the conference I will be able to describe the results.

## 1. Introduction

An initial cognitive study of early learning of programming aimed to extract experimental test data to establish novices' understanding process is described in [1]. We believe that different people bring different patterns of knowledge to any new learning process, and learning programming is not exempted from this common fact. Each student tackles the problem in a different way based on their mental model.

The study started with the hypothesis that "we are able to identify small number of groups to represent novice programmers by looking at their problem solving methods and techniques." We were looking for any sub-populations which are likely to achieve success. Our intention was to observe the mental models that students used when thinking about assignment instructions and short sequences of assignments and we hoped to be able to find out what those models are. We administered a test at the very beginning of their course before the students had begun to be taught about assignment and sequence, and then a second time to the same subjects after the topic had been taught. We correlated the results of these two administrations with each other and we found three groups: consistent using a single mental model (44%), inconsistent using

several mental models (39%) and blank not answering (8%), and an apparent correlation between the consistent group and students who successfully passed the test.

The result demonstrated that the success in the first stage of an introductory programming course may be predictable, by examining the way that students approach to a basic programming problem even before they have had any contact with programming notation.

In our previous study the decision that led us to assign students to each particular group was rather subjective. We looked for repeated use of the same or related models, but we did not assign thresholds or describe patterns of relationships. Introducing an objective marking mechanism, one which will perhaps allow mechanical marking of the test and is free of any biases such as markers' prejudice, has become a high priority, particularly when we received offer from an Australian and a Canadian institution to replicate our experiment. I present here an objective marking method which I hope will lead us to more precise and more finely-graduated predictions. This method is being trialed in at least one experiment, and we hope that by the time of the conference we will be able to describe the results. This method also allows us to graduate levels of consistency. In this paper I describe the method, its interpretation algorithms and speculate on its potential advantages. With a bit of luck before September I shall be able to prepare a data analysis report with Australian and/or Canadian data and present it to the conference.

## 2. Related Work

An initial cognitive study of early learning of programming aimed to extract experimental test data to establish novices' understanding process is described in [1]. We believe that different people bring different patterns of knowledge to any new learning process, and learning programming is not exempted from this common fact. Each student tackles the problem in a different way based on their mental model.

The study started with the hypothesis that "we are able to identify small number of groups to represent novice programmers by looking at their problem solving methods and techniques." We were looking for any sub-populations which are likely to achieve success. Our intention was to observe the mental models that students used when thinking about assignment instructions and short sequences of assignments and we hoped to be able to find out what those models are. We administered a test at the very beginning of their course before the students had begun to be taught about assignment and sequence, and then a second time to the same subjects after the topic had been taught. We correlated the results of these two administrations with each other and we found three groups: consistent using a single mental model (44%), inconsistent using several mental models (39%) and blank not answering (8%), and an apparent correlation between the consistent group and students who successfully passed the test.

The result demonstrated that the success in the first stage of an introductory programming course may be predictable, by examining the way that students approach to a basic programming problem even before they have had any contact with programming notation.

In our previous study the decision that led us to assign students to each particular group was rather subjective. We looked for repeated use of the same or related models, but we did not assign thresholds or describe patterns of relationships. Introducing an objective marking mechanism, one which will perhaps allow mechanical marking of the test and is free of any biases such as markers' prejudice, has become a high priority, particularly when we received offer from an Australian and a Canadian institution to replicate our experiment.

I present here an objective marking method which I hope will lead us to more precise and more finely-graduated predictions. This method is being trialed in at least one experiment, and we hope that by the time of the conference we will be able to describe the results. This method also allows us to graduate levels of consistency. In this paper I describe the method, its interpretation algorithms and speculate on its potential advantages. With a bit of luck before September I shall be able to prepare a data analysis report with Australian and/or Canadian data and present it to the conference.

## 3. Study on Learners

An initial cognitive study of early learning of programming aimed to extract experimental test data to establish novices' understanding process is described in [1]. We believe that different people bring different patterns of knowledge to any new learning process, and learning programming is not exempted from this common fact. Each student tackles the problem in a different way based on their mental model.

The study started with the hypothesis that "we are able to identify small number of groups to represent novice programmers by looking at their problem solving methods and techniques." We were looking for any sub-populations which are likely to achieve success. Our intention was to observe the mental models that students used when thinking about assignment instructions and short sequences of assignments and we hoped to be able to find out what those models are. We administered a test at the very beginning of their course before the students had begun to be taught about assignment and sequence, and then a second time to the same subjects after the topic had been taught. We correlated the results of these two administrations with each other and we found three groups: consistent using a single mental model (44%), inconsistent using several mental models (39%) and blank not answering (8%), and an apparent correlation between the consistent group and students who successfully passed the test.

The result demonstrated that the success in the first stage of an introductory programming course may be predictable, by examining the way that students approach to

a basic programming problem even before they have had any contact with programming notation.

In our previous study the decision that led us to assign students to each particular group was rather subjective. We looked for repeated use of the same or related models, but we did not assign thresholds or describe patterns of relationships. Introducing an objective marking mechanism, one which will perhaps allow mechanical marking of the test and is free of any biases such as markers' prejudice, has become a high priority, particularly when we received offer from an Australian and a Canadian institution to replicate our experiment.

I present here an objective marking method which I hope will lead us to more precise and more finely-graduated predictions. This method is being trialed in at least one experiment, and we hope that by the time of the conference we will be able to describe the results. This method also allows us to graduate levels of consistency. In this paper I describe the method, its interpretation algorithms and speculate on its potential advantages. With a bit of luck before September I shall be able to prepare a data analysis report with Australian and/or Canadian data and present it to the conference..

## Predictors of Success

Cross [2] and Mayer & Stalnaker [3] attempted to use occupational aptitude tests to predict successful candidates for software industry employers. In [4] Cross admits that they had not been very successful in predicting work adjustment in the computer programming occupation through personality and interest measures. He had relied rather heavily on aptitude tests, which also have not been entirely satisfactory. Mayer & Stalnaker administered a variety of tests such as IBM's Programmer Aptitude Test (PAT), the Wonderlic Personnel Test, and the test of Primary Mental Abilities (PMA) on 580 U.S firms and 98 Canadian users. Surprisingly, in the follow-up survey very little substantive information was obtained. McCoy, Burton [5] indicated good mathematical ability as a success factor in beginners' programming but this claim has not been supported by any objective validation. In [6] Wilson & Shrock claimed three predictive factors in order of importance: comfort level, math, and attribution to luck for success/failure (based on students' beliefs about their reasons for success or failure). The result was weak, hasn't been validated for an individual factor and was biased to a number of subjectivities. Beise et al examine correlations among age, race and sex as predictors of success in a first programming course, particularly for computer science and information systems. Statistical analysis of their data indicates that neither sex nor age is a good predictor of success in the first programming class [7]. Nathan Rountree et al claimed that the students most likely to succeed are those who are expecting to get an 'A' grade and are willing to say so. They rely on students' conscious, believing that a student's expectations may have at least as strong an association [8] [9]. Raymond Lister, et al [10] in a multi-national research project reported that incapability of students' in entry-level programming is lack of the ability to prob-

lem solving. The full initial report on this study is published as a technical report by Fincher et al [11] and followed by more detail on each component of the study, giving a full analysis of the data and justification of the conclusions by de Raadt, et al [12]; Simon, et al [13]; Tolhurst et al [14]. Simon [15] presents a broad overview of this study's aims, method and conclusion in a separate paper. He explains that there is no accepted measure of programming aptitude therefore we cannot find correlations between performance on simple tasks and programming aptitude.

## Psychology and Mental Models

Johnson-Laird was the first to point out that psychology is everywhere, even in programming. His notion of 'mental model' lies behind this study, and most research on learners [16]. Kessler and Anderson [17] and Mayer [18] all stressed on the significance of mental models in learning. Benedict and du Boulay catalogued the difficulties that novices experienced and highlighted some fairly distinctive views of teaching programming using a mechanistic analogy [19]. Pennington looked at the way that expert programmers understand problem domains and programs and their bottom-up approach to build program [20]. Vikki Fix, at el differentiated mental models of novices and experts [21] Perkins and others described novice learners' problem-solving strategies as "stoppers", and "movers" [22]. Mayer described existing knowledge as a "cognitive framework" and how new information is connected to existing knowledge [23]. Michael V. Doran , David D. Langan used a cognitive-based approach and observed distinctive attributes of the learning process [24]. Dyck and Mayer emphasised on clear understanding of the underlying virtual machine in novice's learning process. [25]. Putnam et al studied impact of novices' misconceptions about the capabilities of computers[26]. Van Someren found that mechanical understanding of the way the language implementations is the key to success[27].

## Misconceptions/Bugs/Common Mistakes

Soloway and Spohrer found that just a few types of bug cover almost all those that occur in novices' programs. They introduced "programming goals/subgoals/plans" [28]. In [29] same authors studied novice's background knowledge and the type of their misconceptions. Adelson and Soloway reported that domain experience affects novices' ability to understand the sort of programming [30]. Jeffrey Bonar , Elliot Soloway found that skill in natural language seemed to have a great deal of impact on their conceptions and misconceptions of programming [31]. In [32] the same authors explained impact of prior knowledge of one on novices' attempts to program in a second language. Shneiderman blamed the different uses of variables. [33]. Samurcay looked at different ways variables are assigned values through assignment statements and describes how how internal variables like initialisation and updating is harder for novice programmers [34]. Benedict and Du Boulay [19] identified misconceptions about variables based upon the analogies used in class, misconception of linking variables by assigning them to each other, misunderstanding of temporal scope of variables, forgetting about initializations. Perkins and Simmons talked about a misconception that students may have about the names of variables [35]. Adamzadeh, at el look

at debugging. They surprisingly found that less than half of the good programmers are good debuggers [36]

# 4. Preliminary Result

In [1] we reported on a preliminary study of 60 subjects. We summarise that report here. The subjects answered a questionnaire, with questions as illustrated in Fig. 1. and Fig. 2. Each question gives a sample Java program, with two variable declarations and one, two or three assignment instructions. The multiple-choice list is based on our initial notion of the mental models that a novice might employ when answering the question. We allowed for 8 models and in subjects' responses we found evidence for 3 more.

| **1. Read the following statements and tick the box next to the correct answer in the next column.**<br>`int a = 10;`<br>`int b = 20;`<br>`a = b;` | **The new values of a and b are:**<br>☐ `a = 30 b = 0`<br>☐ `a = 30 b = 20`<br>☐ `a = 20 b = 0`<br>☐ `a = 20 b = 20`<br>☐ `a = 10 b = 10`<br>☐ `a = 10 b = 20`<br>☐ `a = 20 b = 10`<br>☐ `a = 0 b = 10`<br>☐ **If none, give the correct values:**<br>**a = b =** | **Use this column for your rough notes please** |
|---|---|---|

**Fig. 1. Sample question with one assignment**

Our intention was to discover the mental models that students used when thinking about assignment instructions. We expected that novices would display a wide range

| **4. Read the following statements and tick the box next to the correct answer in the next column.**<br>`int a = 10;`<br>`int b = 20;`<br>`a = b;`<br>`b = a;` | **The new values of a and b are:**<br>☐ `a = 0 b = 20`<br>☐ `a = 20 b = 20`<br>☐ `a = 10 b = 0`<br>☐ `a = 10 b = 10`<br>☐ `a = 30 b = 50`<br>☐ `a = 0 b = 30`<br>☐ `a = 40 b = 30`<br>☐ `a = 30 b = 0`<br>**Any other values for a and b:**<br>`a = b =`<br>`a = b =`<br>`a = b =` | **Use this column for your rough notes please** |
|---|---|---|

**Fig. 2. sample question with multiple assignments**

of mental models, and that as time went on the ones who were successfully learning to program would converge on a 'correct' mental model that corresponds to the way that a Java program works. We anticipated that we would administer the questionnaire early in the course and then again to the same subjects after the assignment and sequence topic had been taught. We wanted further to correlate the results of our questionnaire with the students' marks in a final course test or examination.

In the first test we could hardly expect that students would choose the Java model of assignment, but it rapidly became clear that despite their various choices of model, in the first administration they divided into three clear groups:

- About a half used the same model for all, or almost all, the questions. This was the *consistent* group.
- About a third used different models for different questions. This was the *inconsistent* group.
- About a sixth refused to answer all or almost all of the questions. This was the *blank* group.

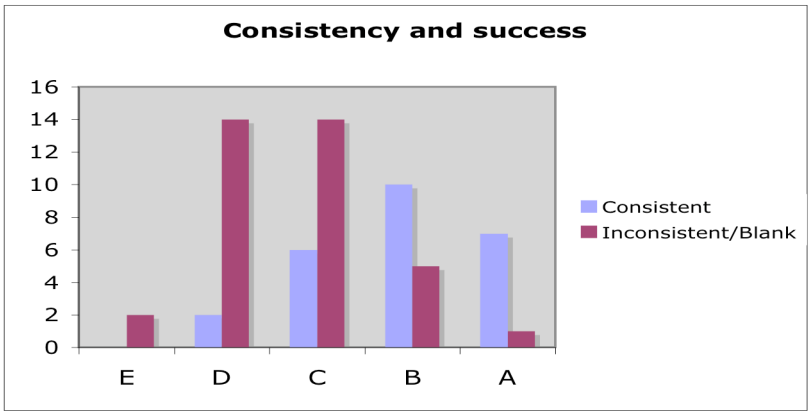
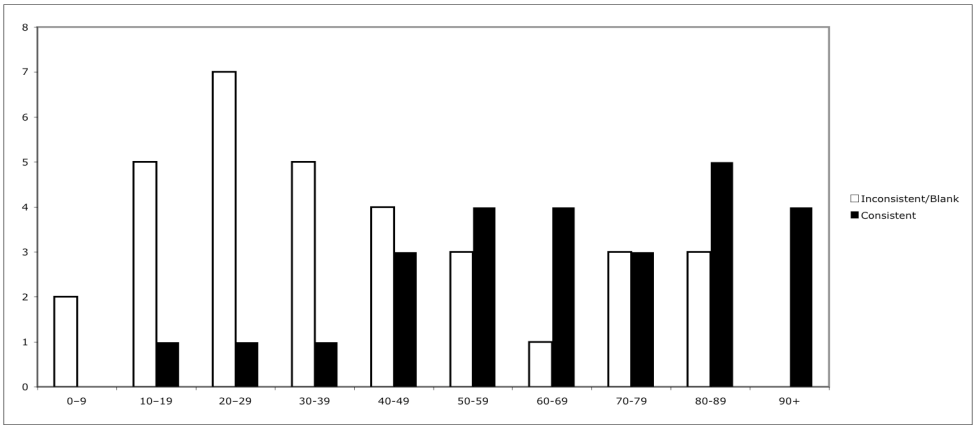


**Fig. 3. first test versus official course result**

**Fig. 4. first test result versus more stringent in-course exam**

In the second administration most subjects became consistent and only one (a non-attender) inconsistent. That is, students learnt to answer the questions in our test.

When we correlated the results of the first test with the official course results, we found the result shown in Fig. 3. There is clearly some separation of populations – the consistent hump is centred on B, the rest on C/D. But exam results are subject to fudging, especially to make it easier for weak students to pass. When we correlated the same test with the more stringent of the official in-course exams, we found the distributions (Fig. 4): a wider separation of humps but a more complex picture, which we hesitate at this stage to explore any further since the number of subjects is still relatively small.




| Model | Description | Answer/s |
|---|---|---|
| M1 | The value of **b** is given to **a** and **b** changes its value to zero. **a <- b  b <- 0** | **a = 20  b = 0** **3rd Answer** |
| M2 | The value of **b** is given to **a** and **b** keeps its original value. **a <- b  // b unchanged** | **a = 20  b = 20** **4th Answer** |
| M3 | The value of **a** is given to **b** and a changes its value to zero. **b <- a  a <- 0** | **a = 0  b = 10** **8th Answer** |
| M4 | The value of **a** is given to **b** and **a** keeps its original value. **b <- a  // a unchanged** | **a = 10  b = 10** **5th Answer** |
| M5 | The sum of **a** and **b** is given to **a**, and **b** keeps its original value. **a <- (a + b) // b unchanged** | **a = 30  b = 20** **2nd Answer** |
| M6 | The sum of **a** and **b** is given to **a**, and **b** changes its value to zero. **a <- (a + b)  b <- 0** | **a = 30  b = 0** **1st Answer** |

| | | |
|---|---|---|
| **M7** | The sum of **a** and **b** is given to **b**, and **a** keeps its original value. **b <- (a + b) // a unchanged** | **a = 10 b = 30**<br>**not predicted** |
| **M8** | The sum of **a** and **b** is given to **a**, and **b** changes its value to zero. **B <- (a + b) a <- 0** | **a = 0 b = 30**<br>**not predicted** |
| **M9** | **a** and **b** keep their original values.<br>**a unchanged // b unchanged** | **a = 10 b = 20**<br>**6th Answer** |
| **M10** | Assignment is a simple equation, and then all equal values of **a** and **b** are acceptable. | **a = 10 b = 10**<br>**And**<br>**a = 20 b = 20** |
| **M11** | **a** and **b** swap their values simultaneously.<br>**a <- b -> a gets b's value**<br>**b <- a -> b gets a's value** | **a = 20 b = 10**<br>**7th Answer** |

**Fig. 5. Mental model of a java assignment a = b**

# 5. Towards an objective analysis

The consistent / inconsistent / blank assignment which is the basis of our preliminary result was rather subjective. When we received offers from an Australian and a Canadian institution to replicate our experiment, it became necessary to find a more objective means of assessment. I have prepared three tools of assessment: a list of mental models; an answer sheet, associating mental models with subjects' answers; and a mark sheet which displays a subject's overall performance. All these materials are available to view in my website [1].

## Mental Models of Single Assignment

The models we observe are shown in Fig. 5. Model M10 (equality) overlaps with models M2 (right-to-left copy) and M4 (left-to-right copy): the difference is detected by noting multiple answers (see Fig. 6).

## Mental Models of Multiple Assignment

Multiple assignment questions require a mental model of sequential / independent / simultaneous execution, exemplified in.

| Models | Description | Example |
|---|---|---|
| **Sequence** | **M1** applies sequentially through both statements:<br>**L1) a <- b and b <- 0 then a = 20 b = 0**<br>**L2) b <- a and a <- 0 then b = 20 a = 0** | **Single answer**<br>**a = 0 b = 20** |
| **Independent** | **M1 I (M1+Independent)**<br>Model is **M1** that applies independently for each individual line. | **Multiple answers**<br>**a = 20 b = 0** |

| | | |
|---|---|---|
| | **(L1)** The value of **b** is given to **a** and **b** changes its value to zero. **L1) a <- b and b <- 0**<br>**(L2)** The value of **a** is given to **b** and **a** changes its value to zero. **L2) b <- a and a <- 0** | **a = 0 b = 10** |
| **Spontaneous-single** | **M1 Ss (M1+Simultaneous+single)** Same as **(M1 I)** subjects only interested on Left-hand-side values of statements and ignores the right-hand-side values.<br>The value of **b** in **(L1)** and value of **a** in **(L2)** are ignored.<br>**L1) a <- b and b <- ignores**<br>**L2) b <- a and a <- ignores** | **Single Answer**<br>**a = 20 b = 10** |

**Fig. 6. Additional mental models for a question with multiple assignments**

## Answer Sheet for Single Assignment Questions

In the answer sheet for Q1-Q3 (single assignment questions) there are ten single-tick boxes (M1 to M11) and one double-tick box (M10). If the subject gives one tick, we use a single-tick box. If they give two ticks in the positions specified, we use the double-tick box. We can't interpret anything else (Fig. 7).

| Question | Answer/s | Model/s |
|---|---|---|
| 1.<br>int a = 10;<br>int b = 20;<br>a = b; | a = 20 b = 0 | M1 |
| | a = 20 b = 20 | M2 |
| | a = 0 b = 10 | M3 |
| | a = 10 b = 10 | M4 |
| | a = 30 b = 20 | M5 |
| | a = 30 b = 0 | M6 |
| | a = 10 b = 30 | M7 |
| | a = 0 b = 30 | M8 |
| | a = 10 b = 20 | M9 |
| | a = 20 b = 10 | M11 |
| | a = 20 b = 20<br>a = 10 b = 10 | M10 |

**Fig. 7. Answer sheet to single question**

## Answer Sheet for Multiple Assignment Questions

In multiple assignments (Q4 onwards) there is more complexity. The answer sheet of question 4 demonstrates in Fig. 8. First some of the models are decorated with I(Independent) or Ss(Simultaneous + single) that explained in Fig. 6. Independent (I) models have very rarely been observed in novices' papers while Ss(Simultaneous + single) appears more frequently.

| Question | Answer/s | Model/s |
|---|---|---|
| 4.<br>int a = 10;<br>int b = 20;<br>a = b; | a = 0 b = 20 | M1 |
| | a = 20 b = 10 | M1 Ss / M2 Ss / M3 Ss / M4 Ss / M11 Ss |
| | a = 20 b = 20 | M2 |
| | a = 10 b = 0 | M3 |

| b = a; | a = 10 b = 10 | M4 |
|---|---|---|
| | a = 30 b = 50 | M5 |
| | a = 30 b = 30 | M5 Ss / M6 Ss / M7 Ss / M8 Ss |
| | a = 0 b = 30 | M6 |
| | a = 40 b = 30 | M7 |
| | a = 30 b = 0 | M8 |
| | a = 10 b = 20 | M9 / M11 |
| | a = 20 b = 20<br>a = 10 b = 10 | M10 / M2 I / M4 S |
| | a = 20 b = 0<br>a = 0 b = 10 | M1 I / M3 I |
| | a = 30 b = 20<br>a = 10 b = 30 | M5 I / M7 I |
| | a = 30 b = 0<br>a = 0 b = 30 | M6 I / M8 I |
| | a = 10 b = 30<br>a = 30 b = 20 | M7 I |
| | a = 20 b = 10<br>a = 20 b = 10 | M11 I |

**Fig. 8. Answer sheet to multiple questions**

## Mark Sheet

We present here an objective marking method which we hope will lead us to more precise and more finely-graduated predictions. This method is being trialled in at least one experiment, and we hope that by the time of the conference we will be able to describe the results. The new mark sheet is illustrated in Fig.9. Each column here represents a single model, when joins to an adjacent column with a common logical perception, creates more general and less specific concept of joined models.

Joining the models that we think are relatively close and carrying similar perception is still remains some subjectivity about our prejudice that how we interpret the relatively close models. The way that I design the mark sheet I believed that M1 and M2 models are relatively close while some on else could see the M1 and M3 models comparatively closer. We should bear on mind that prejudice in some sense is a part of us and we do what we believe. Also this study is still in its initial stage and has long way to go and lots to discover. We tick the mark sheet's according the detected models in the answer sheet. For Q1-Q3 (single assignment questions) we tick the relevant model and for multiple assignments (Q4 onwards) instead of just ticking the corresponding model column on the mark sheet, we put the "I" or "Ss" next to the tick. The logical explanation of these marks illustrated in Fig. 7. We didn't make any use of that decoration in our analysis so far, but we thought somebody might do one day. Second, some of the single-tick boxes give alternative models. In this case we ticked all of the alternative models on the mark sheet. Then, when we've marked all the questions, try to maximise the coherence of the subject's answers by inking in on of the pencil ticks on each row, so as to maximise the numbers in the summary row (labelled C0 on the mark sheet). Subjective marking is needed to decide what to do with not-entirely-blank scripts. In that time we thought of having our first rule.

*Rule 1: A consistent response to Q1- Q3 (all the ticks in a single column or in two adjacent columns) can be considered non-blank, but if all we get is three ticks all*

*over the place and nothing else, it's blank. If we could get consistent responses to all the double-assignments or the triple-assignments, then that was non-blank too.*

Using joined columns; we can investigate four different levels of consistency in the rows that represent by labels *C0*, *C1*, *C2* and C3. *Level C0* contents of the 11 single models and demonstrates the highest rate of consistency while sliding toward level C3 leads to lower rate and poorer sign of consistency.

*Level C1* contents of 4 columns that each is created by joining two adjacent models, logically carried common concepts. M1 and M2, M3 and M4, M5 and M6, M7 and M8. Each of these new columns logically approved Assignment, assigning value to the left or to the right. *Level C2* contents 2 columns that each is created by joining 4 adjacent models, logically carried common concepts. M1 and M2 and M3 and M4, M5 and M6 and M7 and M8. Each of these new columns logically approved Assignment, assigning value to the left and to the right. *Level C3* contents of a single column that created by joining 8 other models, logically carried common concepts. M1 and M2 and M3 and M4 and M5 and M6 and M7 and M8. The new column logically approved assignment.

| Q | Assignment | | | | | | | | No effect | Equal sign | Swap values | Remarks |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | Assign-to-left | | Assign-to-right | | Add-Assign-to-left | | Add-Assign-to-right | | M9 Ss I | M10 Ss I | M11 Ss I | |
| | M1 Ss I | M2 Ss I | M3 Ss I | M4 Ss I | M5 Ss I | M6 Ss I | M7 Ss I | M8 Ss / I | | | | |
| 1 | | 1 | | | | | | | | | | |
| 2 | 1 | | | | | | | | | | | |
| 3 | | 1 | | | | | | | | | | |
| 4 | | | 1 | | | | | | | | | |
| 5 | 1 | | | | | | | | | | | |
| 6 | | | | | 1 | | | | | | | |
| 7 | | 1 | | | | | | | | | | |
| 8 | 1 | | | | | | | | | | | |
| 9 | 1 | | | | | | | | | | | |
| 10 | | 1 | | | | | | | | | | |
| 11 | | | 1 | | | | | | | | | |
| 12 | 1 | | | | | | | | | | | |
| C0 | 5 | 4 | 2 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | |
| C1 | 9 | | 2 | | 1 | | 0 | | 0 | | | |
| C2 | 11 | | | | 1 | | | | | | | |
| C3 | 12 | | | | | | | | | | | |

**Fig. 9. Mark Sheet**

## Interpretation of mark sheet

Two different algorithms have been considered to interpret the mark sheet and make an objective decision on subjects' level of consistency. In the first method in order to identify the C level we look at the pattern of subject's adapted models (ticks) in the mark sheet. Fig. 11 illustrates the tree structure of this algorithm. According to this instruction the consistency level in the sample illustrated in Fig. 10 is C3, as 5 different models (M1 to M5) have been used by subject that leads the tree to it's forth level (C3).

In the second method we use mode analysis of the numeric figures accumulated in C0 to C3 (Fig.11). In each C row, the mode of the accumulated figures represents the numeric value for graduate level of consistency. Mode of figures in level C0 (0, 1, 3, 5) is 5 and in C1 (0, 1, 3, 8) is 8. This increases in C2 and C3 to 11 and 12 respectively. This is a typical scenario that we should expect to happen during the marking more frequently. While the mode value in both C2 and C3 is relatively high, choosing one or another can create subjectivity that should be clarified by an explicit rule. I introduced the following protocol to keep this decision process objective.

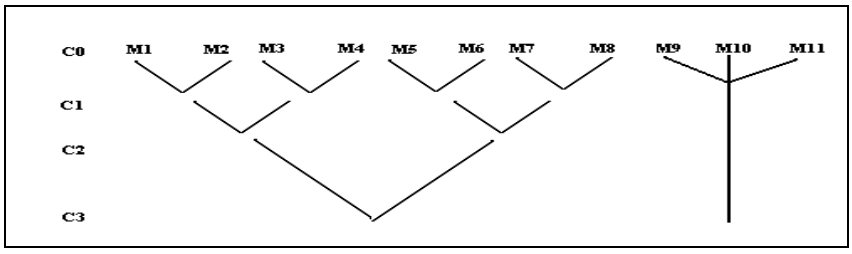



**Fig. 10. Tree structure of models in mark sheet**

*Rule 2: Any C level can be considered as subject's level of consistency if:*
*Mode value in C level >= abs (no. of answered questions * 80%) and*
*no. of answered questions >= abs (no. of questions * 80%)*

According to the above rule the subject in the sample is consistent in C1 level. This method creates around 20% flexibility in C level of subject's that answered 80% of the questionnaire, while in tree method there isn't any space for subject's mistakes. These methods are still debatable and open to new ideas.

# 6. Further work

We are looking for receiving fresh data in our new mark sheet from Australia and Canada. Surely, the fresh data analyses, will tell us more about joined columns and interpretation algorithms introduced in this mark sheet. The result can effect our instru-

ments in the series of test that will be administered widely at the beginning of the next academic year in Middlesex University with more than 250 participations in introductory to programming course. Any possible of subjectivity in marking method will be monitored closely. While a mechanical marking system has always reckoned as an ideal solution but never been materialised, design and implantation a software to read the mark sheet and evaluates the subject's C level is desirable.

## 7. Conclusion

The result of our preliminary study demonstrated that our categorisation method is more likely to be used as a reasonable predictor of success in introductory programming. There were few issues of subjectivity in interpretation of our result that I have identified and attempted to clarify. I can say that the test instruments now are highly objectified and prepared for the next experimental phase of this study. This study also suggests more exploration of other possible tools and instruments to capture the same categorisation, in order to examine success prediction. I emphasis on clear understanding of novice's mental models that is a key element to build empirical tools to measure programming ability.

## References


1. Dehnadi, S. and R. Bornat. *The camel has two humps*. in *Little PPIG* 2006. Coventry, UK: http://www.cs.mdx.ac.uk/research/PhDArea/saeed/.
2. Cross, E. *The behavioral styles of computer programmers*. in *Proc 8th Annual SIGCPR Conference*. 1970. Maryland, WA, USA.
3. Mayer, D.B. and A.W. Stalnaker. *Selection and Evaluation of Computer Personnel – the Research History of SIG/CPR*. in *Proc 1968 23rd ACM National Conference,*. 1968. Las Vegas, NV, USA.
4. Cross, E., M. , *Behavioral styles of computer programmers - revisited*, in *Proceedings of the ninth annual SIGCPR conference*. 1971, ACM Press: Illinois, United States.
5. McCoy, L.P. and J.K. Burton, *The Relationship of Computer Programming and Mathematics in Secondary Students.* Computers in the Schools, 1988. **4**(3/4): p. 159-166.
6. Wilson, B.C. and S. Shrock, *Contributing to success in an introductory computer science course: a study of twelve factors*, in *Proceedings of the thirty-second SIGCSE technical symposium on Computer Science Education*. 2001, ACM Press: Charlotte, North Carolina, United States.
7. Besie, C., et al., *An Examination of Age, Race, and Sex as Predictors of Success in the First Programming Course.* Journal of Informatics Education Research, 2003. **5**: p. 51-64.
8. Rountree, N., et al., *Interacting factors that predict success and failure in a CS1 course*, in *Working group reports from ITiCSE on Innovation and technology in computer science education*. 2004, ACM Press: Leeds, United Kingdom.

9. Rountree, N., et al. *Predictors For success in studying CS*. in *Proceedings of the 35th SIGCSE technical symposium on Computer science education*. 2004. Norfolk, Virginia, USA: ACM Press.
10. Lister, R., et al., *A Multi-National Study of Reading and Tracing Skills in Novice Programmers*. 2004.
11. Fincher, S., et al., *Programmed to succeed?: a multi-national, multiinstitutional study of introductory programming courses*, in *Computing Laboratory Technical Report 1-05*, C. University of Kent, UK., Editor. 2005.
12. de Raadt, M., et al., *Approaches to learning in computer programming students and their effect on success.* Higher Education in a changing world: Research and Development in Higher Education, 2005. **28**: p. 407-414.
13. Simon, et al. *The ability to articulate strategy as a predictor of programming skill*. in *Proc Eighth Australasian Computing Education Conference,*. 2006. Hobart, Australia.
14. Tolhurst, D., et al. *Do map-drawing styles of novice programmers predict success in programming? A multi-national,multi-institutional study*. in *Proc Eighth Australasian Computing Education Conference*. 2006. Hobart, Australia.
15. Simon, et al., *Predictors of success in a first programming course.* Eighth Australasian Computing Education Conference, Hobart, 2006.
16. Johnson-Laird, P.N. and W. P.C., *Thinking Reading in Cognitive Science*. First ed. 1977: Cambridge University Press.
17. Kessler, C.M. and J.R. Anderson, *Learning flow of control: Recursive and iterative procedures.* Human-Computer Interaction, 1986. **2**: p. 135-166.
18. Mayer, R.E., *Thinking, Problem Solving, Cognition*. 2 ed. 1992, New York: W. H. Freeman and Company Second Edition ISBN 0716722151.
19. Benedict, J.H. and B. Du Boulay, *Some difficulties of learning to program.* Journal of Educational Computing Research, 1986. **2**(1): p. pp. 57-73.
20. Pennington, N., *Stimulus structures and mental representations in expert comprehension of computer programs*. Cognitive psychology, 1987. **19**: p. 295-341.
21. Fix, V., S. Wiedenbeck, and S. Jean, *Mental representations of programs by novices and experts*, in *Proceedings of the SIGCHI conference on Human factors in computing systems*. 1993, ACM Press: Amsterdam, The Netherlands.
22. Perkins, D.N., et al., *Conditions of Learning in Novice Programmers. Studying the Novice Programmer. E. Soloway and J. C. Spohrer.* Publishers: Hillsdale, NJ, Lawrence Erlbaum Associates, 1989: p. 261-279.
23. Mayer, R.E., *The psychology of how novices learn computer programming. In E. Soloway and J.C. Spohrer, editors, Studying the Novice Programmer.* Publishers: Lawrence Erlbaum Associates, Hillsdale, 1989.
24. Doran, M.V. and D.D. Langan, *A Cognitive-Based Approach to the implementation of the introductory Computer Science Courses: lesson learned.* Proceeding of the twenty-six SIGCSE technical symposium on Computer Science Education, 1995. **1995**: p. 218-222.
25. Dyck Jennifer, L. and E. Mayer Richard, *BASIC versus natural language: is there one underlying comprehension process?*, in *Proceedings of the SIGCHI conference on Human factors in computing systems*. 1985, ACM Press: San Francisco, California, United States.
26. Putnam, R.T., et al., *A Summary of Misconceptions of High School Basic Programmers.* Journal of Educational Computing Research, 1986. **2**(4).
27. Someren, V. and W. Maarten, *What's Wrong? Understanding beginners' problems with Prolog.* Instructional Science, 1990. **19**(4/5): p. pp 256-282.
28. Spohrer, J., C. and E. Soloway, *Novice mistakes: are the folk wisdoms correct?* Commun. ACM, 1986. **29**(7): p. 624-632.

29. Soloway, E.a.S., *Some Difficulties of Learning to Program. In book studying the Novice Programmer.* Publishers : Lawrence Erlbaum Associates 1989, 1989: p. pp 283 - 299.
30. Adelson, B. and E. Soloway, *The role of domain experience in software design.* IEEE Trans. Softw. Eng., 1985. **11**(11): p. 1351-1360.
31. Bonar, J. and E. Soloway, *Uncovering principles of novice programming*, in *Proceedings of the 10th ACM SIGACT-SIGPLAN symposium on Principles of programming languages*. 1983, ACM Press: Austin, Texas.
32. Bonar, J. and E. Soloway, *Pre-Programming Knowledge: A Major Source of Misconceptions in Novice Programmers.* Human-Computer Interaction, and fall, 1985. **1**(2): p. 133-161.
33. Shneiderman, B., *When Children Learn Programming: Antecedents, Concepts and Outcomes.* The Computing Teacher, 1985. **12**(5): p. 14-17.
34. Samurcay, R., *The Concept of Variable in Programming: Its Meaning and Use in Problem-Solving by Novice Programmers.* Education Studies in Mathematics, 1985. **16**(2): p. 143-161.
35. Perkins, D.N. and R. Simmons, *Patterns for Misunderstanding: An Integrative Model for Science, Math, and Programming.* Review of Educational Research, 1988. **58**(3): p. 303-326.
36. Ahmadzadeh, M., D. Elliman, and C. Higgins, *An analysis of patterns of debugging among novice computer science students*, in *Proceedings of the 10th annual SIGCSE conference on Innovation and technology in computer science education*. 2005, ACM Press.