Of the 67 students to complete both tests, 36 stated they had prior programming experience. The most common dominant mental model was *m2* which was used by 34 students, suggesting that their pre-established domain knowledge is potentially influencing the mental model usage.

11 students who used the *m2* model also used the *s3* juxtaposition, with an additional student being classed as using an invalid juxtaposition (*NA*). This suggests that despite previous experience, understanding how variables are handled within a computer is still a potential threshold concept.

74% of students with previous programming experience consistently used a single mental model (*c0*), with all 22 students who used the *m2s1* combination doing so consistently. With 26% of students being inconsistent in their mental model usage it would suggest that even though they have prior programming experience, some students are still adapting to using an appropriate model.

When analysing the results of the 31 students who stated that they did not have any prior programming experience it was discovered that the students were relatively consistent with their mental model usage with 71% consistently using a single (not necessarily correct) model however, only 5 consistently used *m2s1*. There was also a greater variety of dominant models used by students with no prior programming experience as opposed to those with experience, with 5 different models as opposed to 3 being used respectively.

5.3. Second aptitude test

By conducting a second aptitude test at the end of the first semester, it allows the development of students' mental model usage to be investigated. We expected that students who have already adopted their dominant combination would continue to use it with increased consistency. We predicted that students who previously used other mental model combinations will begin to gravitate towards a new model because their domain knowledge begins to adapt as they progress through the course. Constructivism theory positions that students accrue their knowledge recursively by taking an active part in the learning process [2], i.e. completing practical lab sessions, aimed at teaching students the fundamental concepts of programming by putting them into practice.

This could potentially mean a decrease in overall consistency as students may experience cognitive conflict; when a student experiences a discrepancy between their own cognitive structure and an external environment [11], i.e. conflict between a student's original incompatible mental model and the appropriate model used in programming which they are attempting to learn.

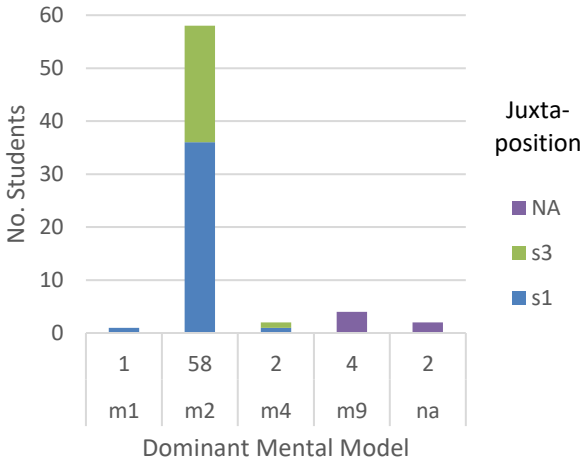

Figure 2. Mental Model / Juxtaposition usage of students during the second aptitude test

In total, 21 students improved their model usage between the two tests, either by switching from an inappropriate model to *m2* (any juxtaposition), or by switching from an invalid model (*NA*) to a known valid (although potentially inappropriate) model. 49 students remained consistent in their usage between the two tests and 6 students became less consistent but continued to use the same dominant model, perhaps due to them beginning to adapt to a more appropriate model.

As Figure 2 shows, the variety of models used by all students in the test group has decreased when compared to the results of the first test (see Figure 1). There has also been a clear uptake in the number of students using the *m2* model. In total, 58 students (86%) are now using *m2*, with over half (56%) the sample group using *m2s1*, a rise of 14% from the first test. The number of students consistently using a single model has slightly decreased from 73% in the first test to 70% in the second.

48% of students consistently used *m2s1* whilst the number of students using *m2s3* (of any consistency) has dropped slightly to 33%, supporting the idea that understanding variables is a potential threshold concept due to the large portion of students not applying an appropriate juxtaposition to the variable swapping operations.

Interestingly, there has been very little variation in the mental model/juxtaposition usage of students with past programming experience between the two tests. The number of students using the *m2* model has remained constant, however, four students who previously used the *s3* juxtaposition have now begun to use *s1*. Consistent use of a single model has also increased amongst students with previous experience from 74% to 77%. The number of students consistently using an appropriate mental model/juxtaposition combination (*m2s1c0*) has also

increased from 22 to 24. This adds support for out theory that as the domain knowledge develops, a student will begin to gravitate towards *m2s1c0* due to the knowledge they have acquired, which in turn reduces cognitive conflict.

Whilst the variations between the two tests were relatively small for students with past programming experience, there are a number of significant changes amongst those with no previous experience.

The most notable difference is the variety of dominant models used by students has decreased with only 4 out of 28 students having a dominant model other than *m2*. However, only 32% of students are using *m2s1* compared to 69% of students with programming experience. The number of students consistently using a single model fell 10% between the two tests, which we believe to be a sign of cognitive conflict. Maier [12] states that before a student can fully understand a concept, a student's understanding must be challenged before they can adapt. Many of the students who had not previously had any programming experience held incompatible models, which must be first actively challenged by allowing a student to gain programming experience, thus allowing a student to discover for themselves what works, and what doesn't, in a programming context.

We believe the reduction in consistency is attributed to students' mental models being challenged, whilst in some cases they may still be using their previously established models, students should be beginning to gravitate towards the appropriate mental model/juxtaposition combination.

5.4. Module grade comparison

The final key element of this investigation is the comparison of students' mental model/juxtaposition usage and their overall grades for the Introduction to Programming module, thus allowing for predictors of success to be established. While grades are available for all students enrolled on the module, the results discussed below refer to the group of students who completed both aptitude tests. Although this limits the sample size significantly, it allows for a better understanding of a student's development through the module, i.e. a student may have originally started using an incompatible model such as *m9* but by the end of the semester they may have begun to use *m2*. For this reason, the discussed results focus on the dominant models identified during the second test.

55% of students who completed both tests obtained a 1st (70% or above), 73% of which used *m2s1c0* as their dominant combination, whilst 14% combination. 9 students achieved a 2.i (60% - 69%) overall, 33% of which used *m2s1c0* and 33% used *m2s3c0*. 12 students a 2.ii (50% - 59%) in the module with *m2s1c0* and *m2s3c0* were only used by a single person each, 33% used *m2s3incon* (either *c1,c2 or c3*) whilst 25% used an inappropriate model inconsistently (*other incon*). Only 6 students obtained a 3rd (40% -49%), primarily students who achieved this grade used *m2s3incon* (33%) or an inappropriate model inconsistently (*other incon* – 33%).

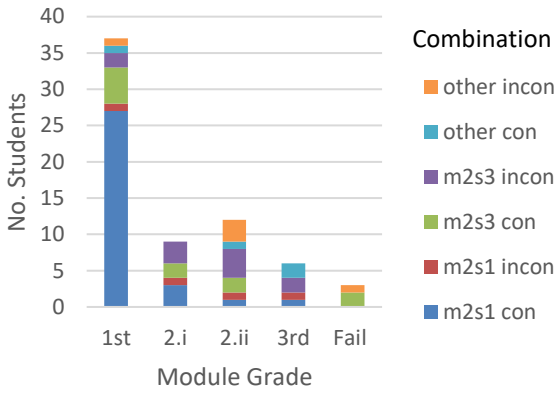

Figure 3. Comparison Mental Model/ Juxtaposition / Consistency and module grades

Three students also failed the module, two of whom used *m2s3c0* and one used an inappropriate model 70% inconsistently. Figure 3 highlights the relationship between mental model/juxtaposition usage, consistency and module grades.

One trend that is immediately identifiable from reviewing Figure 3 is the relationship between the use of *m2s1c0* (*m2s1 con*) and students achieving higher grades. Also, as grades decreased, so did the use of *m2s1c0*. This suggests that the use of *m2s1c0* could potentially be used as predictor of success within the course

A Mann Whitney U test [13] was also carried out to evaluate the significance of the relationship between appropriate mental models and module result, producing a result of $p < 0.05$ and confirming the significance of the data.

5.5. Original participants

Whilst the results discussed previously provide an insight into the development of students throughout the first semester, the sample size is extremely limited due to not all students completing the second test. 123 students took part in the first aptitude test meaning there is a lot of reliable data available, which can be used to support the findings of the primary investigation.

Figure 4 shows the distribution of the dominant mental model/juxtaposition combinations for all students who took part in Test 1. The range of combinations used by students is significantly more varied than that observed in Figure 1, however, a commonality between the two is that *m2* is by far the

most commonly used mental model as 80% of all students utilized *m2* as their dominant model.

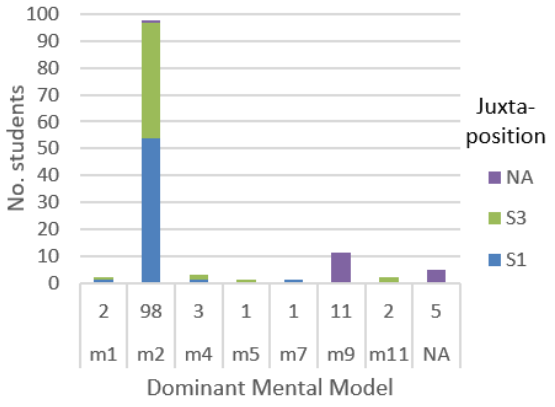

Figure 4. Mental Model / Juxtaposition usage of all students who completed the first aptitude test

There is also an almost equal split between the number of students using the *s1* and *s3* juxtapositions, similar to that observed in the test group. Despite the varied combination uses, 68% of students consistently used a single model with 42% using *m2s1c0*. Of the 123 students who originally took part in the first aptitude test,

73 of the 123 students who took part in the first test stated that they had previous programming experience. 88% of students used *m2* as their dominant model of which 58% used the *s1* juxtaposition. Other than a relatively small percentage of students using other miscellaneous inappropriate models, students who did not use *m2s1* used *m2s3* instead (29%). The high percentage of students using the *s3* juxtaposition; despite their previous programming experience, leads us to believe that the assumption that understanding how variables are handled within a computer is a potential threshold concept for students. 70% of student who had previous programming experience consistently used a single mental model, with over half of students using *m2s1c0*.

50 students who stated that they had no previous programming experience and like the primary test group used a large variety of mental model/juxtaposition combinations. *m2s3* was the most common dominant combination, being used by 44% of student. 68% of students consistently used a single mental model/juxtaposition, however, less than a quarter used *m2s1c0*.

As only 54% of students completed both tests it would not be appropriate to use the results from the second test when comparing the mental model and juxtaposition usage to module grades for the group as a whole. However, by comparing all 123 students' mental model usage from the first test to their module grades, it allows for trends between students' initial mental model usage at the beginning of the course, and the grades they achieve to be established. This not only supports trends discovered with the test group, but also forms a solid foundation for a future implementation of the testing method, which would likely be conducted at the beginning of a course.

54 students (44%) obtained a 1st overall in the module, 18 (15%) achieved a 2.i, 18 (15%) achieved a 2.ii, 16 (19%) achieved a 3rd and 8 (7%) students failed. 8 students also had no corresponding grades, potentially due to incorrect ID numbers being entered at the start of the test.

Out of the 73 students with programming experience, 50% obtained a 1st in the module, the majority of which used *m2s1*, it was also determined that 75% of students using *m2s1* used it consistently (*m2s1c0*). A Mann-Whitney U test [13] produced a significance of $p < 0.05$, further supporting claims of a link between appropriate Mental Model usage and success in the module.

The remaining 50 students who took part in the first test and did not state that they had any pervious programming experience and as a result, their lack of experience has translated into a much greater variety of mental models being used, as well as a smaller proportion of students achieving higher grades than students who had programmed in the past.

Only 34% of students who had no previous programming experience obtained a 1st in the module. However, this was the most commonly achieved grade with 22% achieving a 2.i, 14% achieving a 2.ii, 14% achieving a 3rd and 6% failing the module.

Despite the more varied results; when compared to students with previous programming experience, *m2s1c0* was still the most prevalent model combination amongst students achieving higher grades, with it being used by 47% of students achieving a 1st in the module.

The number of student using *m2s1c0* falls drastically as grades drop, with it being used by 27% of students who obtained a 2.i and then not at all by students achieving subsequent grades. The most prominent model combination used by students who achieved a 2.i and 2.ii is *m2s3c0*; 55% and 57% respectively, whereas students who achieved a 3rd or failed the module primarily used an inappropriate model inconsistently (*other incon)* or *m2s3c0*. A Mann-Whitney U test [13] confirmed the significance in the relationship between module grades and Mental Model/Juxtaposition combination usage by students with a significance of $p < 0.05$.

5.6. Business students

To further investigate the impact domain knowledge has on mental model usage, we ran the test with a group made up of 23 Business Studies students (16 foundation year, 7 first year) and 5 Accounting students (4 foundation year and 1 first year). Whilst

both these subjects involve mathematics, we believed them to sufficiently different from Computer Science and other subjects that require an understanding of abstract concepts, i.e. Physics [17], to provide a basis for comparisons of students' mental model usage.

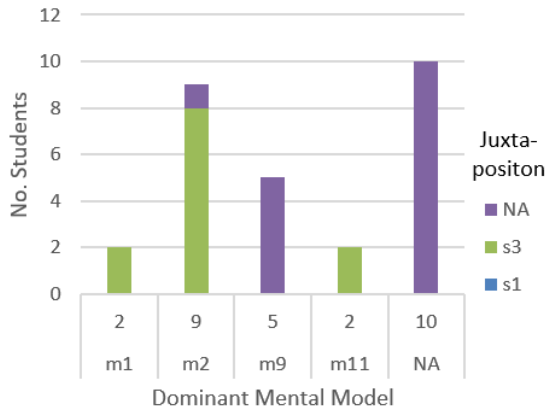

Figure 5. Mental Model / Juxtaposition usage of business students

A relatively large variety of mental models were used by students as shown by Figure 5. However, two mental models in particular attracted the most students – *NA* and *m2*. 32% of students utilised the m2 model, 88% of which used the *s3* juxtaposition. The large portion of student using s*3* combined with the fact that no students used *s1* suggests a misunderstanding of how variables are handled, supporting the idea that variables are a potential threshold concept for students however, the use of *m2* demonstrates that some students; regardless of background, can process compatible mental models which are appropriate for programming.

36% of students were categorized as *NA* meaning that they failed to use a known mental model or entered erroneous data. Interestingly, instead of inputting the value of the variables 4 students submitted code; such as variable names, as their answers. Only 21% of students consistently used a single mental model. However, due to high proportion of students using *NA* it is difficult to draw any statistically significant conclusions.

5.7. Predicting success of failure

At the beginning of a module teachers may not be fully aware of each student's programming ability. Whilst most introductory programming courses start from a basic level, some students, especially students who have attempted to teach themselves to program, may have already developed misconceptions about some of the fundamental concepts of programming, which staff would likely be initially unaware of. If these misconceptions are not identified and rectified by staff, students will construct new models of dubious quality based on their misconceptions [20], making the process of learning to program more difficult for the student, therefore increasing the possibility for failure within the module.

It is therefore important to identify students at the beginning of the module who are most at risk of failure, allowing staff to give them the support they need to overcome their misconceptions and any threshold concepts they may be facing.

The data collected during this research investigation has highlighted a number of factors that may impact on a student's performance within an introductory programming module, making it possible to split students into three distinct categories based on their mental model/juxtaposition usage:

- **On Track** – Consistent use of *m2s1* (*m2s1 con*) indicates an appropriate mental model has been established, giving a student the best chance of success within the course.
- **At Risk** – Students use *m2s1* as their dominant model, yet use it inconsistently (*m2s1 incon*) as students may be transitioning between mental models and may be experiencing cognitive conflict.
  Students who use *m2s3* (either consistently or inconsistently) as their dominant model can also be categorised as 'At Risk', indicating students may be encountering a threshold concept that is blocking their progression within the module. Students have the potential to succeed, however, they are at risk of being held back by threshold concepts and/or cognitive conflict. A student identified as 'At Risk' should be given specific help to ensure they overcome these issues by being encouraged to confront their misconceptions directly [12].
- **Falling Behind** – Students use models other than *m2* either consistently or inconsistently (including *NA*), indicating they have not grasped the concept of how variables are handled and may potentially be experiencing cognitive conflict or being held back by threshold concepts. These students are at the most risk of failure within the module, steps should be taken by staff to address their misconceptions directly [12], as well to help acquire appropriate domain knowledge in order for them to succeed in the module.

By splitting students into three separate categories teaching strategies can be adapted to provide material that is appropriate to each level, for example, students who are classed as 'At Risk' or 'Falling Behind' can be given material aimed at helping them overcome their misconceptions and any threshold concepts they may be encountering. If the same material was given to students who are classed as 'On Track' is possible they may become bored with the module, potentially leading to them underperforming.

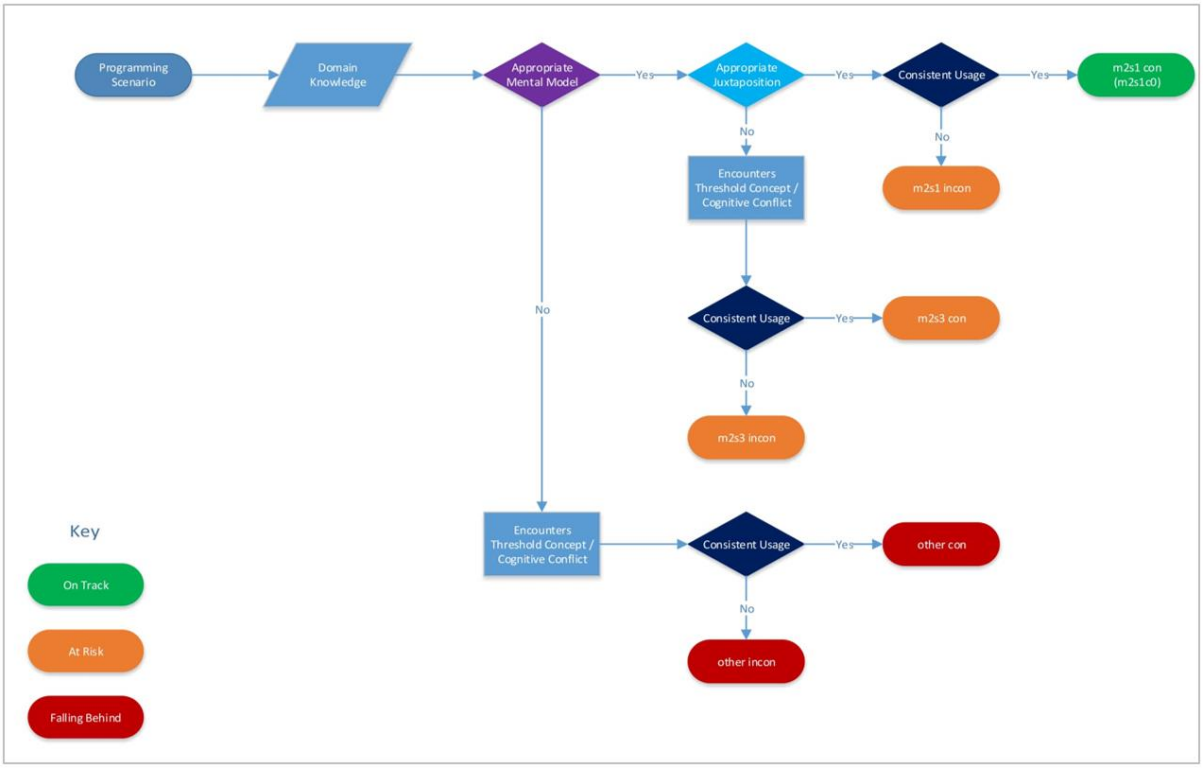

Figure 6. Programming Thought Process (PTP) Model

The thought processes a student goes through when faced with a programming scenario is represented in the Programming Thought Process (PTP) Model. This model is an original contribution of this research and highlights how a student consults their domain knowledge before attempting to approach a programming scenario, which in turn influences the mental model and juxtaposition used by the student.

By also taking consistency into account, the PTP Model (see Figure 6) provides a visual representation of how the thought process of a student relates to the previously described categories, thus allowing staff to identify where a student is encountering problems and provide appropriate support.

6. Conclusions and Future Work

We believe the following conclusions can be drawn from the data collected during this investigation:

- Consistent use of *m2s1* is a predictor of success within introductory programming courses.
- Variable swapping is a potential threshold concept for students; highlighted by the number of students using the *s3* Juxtaposition.
- As domain knowledge develops, students begin to gravitate towards appropriate mental models, however, they may encounter cognitive conflict or threshold concepts that can block their progress.

Despite being relatively successful, there is still future work required to validate the findings of this experiment and to further expand the understanding of the programming learning process.

By only collecting data over a single semester, the reliability and validity of this research are severely constrained. In order to gain a comprehensive understanding of the programming learning process that is valid and highly reliable, the same experiment should be run throughout the course of an entire year, over the course of multiple years and at various institutions.

Students studying a wider range of subjects; such as Media, Music, English Literature, and so on, should also be observed to determine how their mental model usage differs to Computer Science students, and investigate any potential related subjects in order to allow for a better understanding of students' domain knowledge.

Despite the limitations, we believe this research has highlighted the link between appropriate mental model usage and success within an introductory programming module. The data collected during this research has also revealed a potential link between related domain knowledge (from studying subjects such as Mathematics or Physics) and appropriate mental model usage, although further research is needed to validate these results.

7. References

[1] B. Adelson and E. Soloway, The role of domain experience in software design. *IEEE Transactions on Software Engineering*, (11), 1985, pp.1351-1360.

[2] M. Ben-Ari, Constructivism in computer science education, *Acm sigcse bulletin* 1998, ACM, 1998, pp. 257-261.

[3] N. Blackwood, Digital skills crisis: second report of Session 2016–17: report, together with formal minutes relating to the report: ordered by the House of Commons to be printed 7 Jun 2016.

[4] S. Dehnadi, Testing programming aptitude, *Proceedings of the 18th Annual Workshop of the Psychology of Programming Interest Group*, 2006, pp. 22-37.

[5] B. Du Boulay, Some difficulties of learning to program. *Journal of Educational Computing Research*, 2(1), 1986, pp. 57-73.

[6] S. Furber, Shut Down or Restart: Report of the Royal Society into Computing in Schools, *The Royal Society, London*, 2012.

[7] Higher Education Funding Council for England and The Complete University Guide, Dropout Rates at English Universities. Available: http://www.thecompleteuniversityguide.co.uk/news/dropout-rates-fall-at-english-universities/ (Access Date: July 30, 2016.

[8] P.N. Johnson-Laird, Mental models in cognitive science. *Cognitive science*, 4(1), 1980, pp. 71-115.

[9] O. Kerr and N. Danino, Programming Patterns as Potential Predictors of Student Success. *Proc. World Congress on Education (WCE-2017)*, 2017.

[10] E. Lahtinen, K. Ala-Mutka, and H. Järvinen, A study of the difficulties of novice programmers, *ACM SIGCSE Bulletin* 2005, ACM, 2005, pp. 14-18.

[11] G. Lee, and J. Kwon, What Do We Know about Students' Cognitive Conflict in Science Classroom: A Theoretical Model of Cognitive Conflict Process, 2001.

[12] S. Maier, Misconception research and Piagetian models of intelligence, *Proc. 2004 Oklahoma Higher Education Teaching and Learning Conf*, 2004.

[13] H.B. Mann and D.R. Whitney, On a test of whether one of two random variables is stochastically larger than the other. *The annals of mathematical statistics*, 1947, pp. 50-60.

[14] J. Meyer, and R. Land, Threshold concepts and troublesome knowledge: Linkages to ways of thinking and practising within the disciplines, *University of Edinburgh*, 2003.

[15] D.A. Norman, Some observations on mental models, *Mental models*, 7(112), 1983, pp. 7-14.

[16] M. Prensky, Digital natives, Digital immigrants part 1, *On the horizon*, 9(5), 2001, pp. 1-6.

[17] J. Rogalski, and R. Samurçay, Acquisition of programming knowledge and skills. *Psychology of programming*, 18(1990), 1990, pp. 157-174.

[18] M.A. Sasse, Eliciting and describing users' models of computer systems, 1997.

[19] Soloway, E. 1986 Learning to program = learning to construct mechanisms and explanations. *Communications of the ACM, 29*(9), 1986, pp.850-858.

[20] L.E. Winslow, Programming pedagogy—a psychological overview. *ACM SIGCSE Bulletin*, 28(3), 1996, pp. 17-22.

