The effects of Professional and Pedagogical Program Development Environments on Novice Programmer Perceptions

D.Vogts,[1] A.P.Calitz[2], J.H.Greyling[3]

Department of Computing Sciences, Nelson Mandela Metropolitan University, South Africa.

ABSTRACT

Novice programmers generally have difficulty learning to program and one of the problems contributing towards this is the program development environment used at tertiary institutions. A number of pedagogical program development environments have been developed specifically for novice programmers, but these have not been compared experimentally with professional program development environments. A study was conducted that compared the perceptions of novice programmers using a representative professional program development environment to a pedagogical program development environment during an Introductory Programming module at a tertiary institution. It was found that the use of a pedagogical program development environment had a positive effect on the feelings of achievement and learning while learning to program, while the perceived ease of using the program development environment and the perceived difficulty of practical assignments were not affected.

CATEGORIES AND SUBJECT DESCRIPTORS

D.2.6 [**Software Engineering**]: Programming Environments – *integrated environments;*

K.3.2 [**Computers & Education**]: Computer and Information Science Education – *computer-science education, curriculum.*

KEYWORDS

novice programmers, perceptions, pedagogical program development environment, professional program development environment.

1. INTRODUCTION

Learning to program is a complex process and is one of the reasons that have contributed to the world-wide decrease in IT-enrollments [1]. While learning to program, numerous skills and processes need to be learnt concurrently and many are interrelated [2]. These include learning the syntax of a programming language, learning to problem solve in a manner unfamiliar to many [3] and learning how to use a program development environment to construct, debug and execute programs.

It has been stated that professional program development environments, which are developed mainly for professional programmers, are inappropriate to use when teaching novice programmers to program [4]. Generally, novice programmers feel that they need to "fight the compiler" in order to successfully compile and execute a program, resulting in many novice programmers thinking that programming is about "getting the syntax right" and spending most of their time on this task.

A number of professional program development environments are configurable to a certain degree, making them more usable to novice programmers. However, there are still features required by novice programmers that have not been addressed, such as experience appropriate help documentation and error messages [5]. Therefore, even though professional program development environments are becoming more configurable, there is still a need for specialised program development environments developed with pedagogical aims in mind.

The perceived inappropriateness of professional program development environments being used by novice programmers learning to program has led to the development of many pedagogical program development environments [6]. There are different types of pedagogical program development environments developed, that include:

- constrained programming languages and their associated program development environments [7, 8] and
- program development environments customised for novice programmers [4, 6, 9, 10], typically involving simple GUIs and features specifically designed for novice programmers.

Even though professional program development environments are believed to be non-conducive to learning to program, many educational institutions use professional program development environments in Introductory Programming modules. Tertiary institutions are often faced with external pressures of having to teach programming in a program development environment used "in the real world".

Research needs to be conducted to provide an unbiased comparison between pedagogical and professional program development environments. Unfortunately, it is not possible to

[1] Email: Dieter.Vogts@nmmu.ac.za

[2] Email: Andre.Calitz@nmmu.ac.za

[3] Email: Jean.Greyling@nmmu.ac.za

say conclusively how much better, if at all, a pedagogical program development environment is for a novice programmer compared to a professional program development environment. The reason for this is that very few *comparative* studies have been conducted [11].

Comparative studies compare two or more program development environments, while reducing the amount of factors, other than the environments, that could bias the results [12]. In a study comparing a pedagogical and professional program development environment, the factors that can be controlled include the presenter of the module, learning material, pacing, programming language and practical assignments. In these studies, it is possible to benchmark one environment against another, without the findings having to be kept in context of the study.

A comparative study was conducted at the Nelson Mandela Metropolitan University to compare the effects of a *representative* pedagogical program development environment with a *representative* professional program development environment on novice programmers learning to program. The effects that were considered were:

- *perceptions*;
- *academic performance*; and
- *programming behaviour* of novice programmers.

The *perceptions* of novice programmers relate to how they perceive different aspects of learning to program. It is related to personal experiences of the novice programmer while interacting with the program development environment and the module content. It has been stated that the personal experiences of people learning new content has an impact on the learning process [13, 14]. Bad experiences can lead to poor levels of motivation, resulting in lower levels of learning. It is particularly true that in the early stages of learning new content that self belief in one's ability to successfully complete tasks is malleable. In other words, the initial experiences of a novice programmer learning to program will impact on the self belief and motivation of the novice programmer, which will in turn affect the remainder of the learning process.

Academic performance pertains to how novice programmers performed academically. It relates to the grades obtained, as well as the throughput (pass rate) for the module. This is important as many tertiary institutions are being required to obtain specific levels of throughput.

Finally, *programming behaviour* relates to how novice programmers performed the various tasks required during programming. The behaviour examined included how often novice programmers made mistakes, accessed help and how frequently programs were compiled and executed.

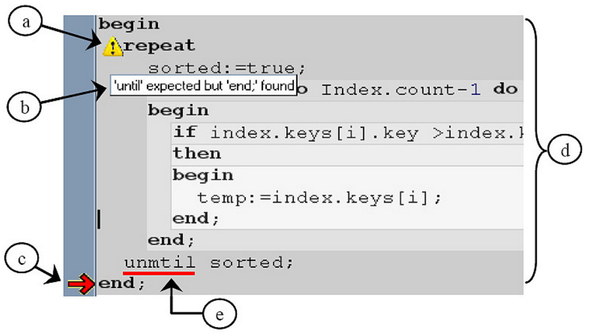
The effect of the program development environment on *academic performance* and *programming behaviour* of novice programmers is reported elsewhere by the authors [15]. This article examines how the *perceptions* of novice programmers differed based on the choice of program development environment used.

The representative pedagogical program development developed, named SimplifIDE, is discussed in Section 2. The design of the experiment is discussed in Section 3, while Section 4 reports on the findings. Finally Section 5 contains a discussion of the findings of the experiment.

2. SIMPLIFIDE

SimplifIDE [16] was created as a program development environment that was representative of a number of existing pedagogical program development environments. SimplifIDE is a plugin for Borland© Delphi 6^{TM}, which modifies the behaviour of the professional program development environment in the following different ways:

- the GUI was simplified, so that only the absolutely necessary components were visible;
- automatic insertion of keywords using a language directed editor;
- interactive structure error indicators (Figure 1a);
- hovering over an error indicator displayed a tool tip with a message (Figure 1b) and a red arrow indicating the token that caused the error to be reported (Figure 1c);
- code structure highlighting indicating related blocks of code (Figure 1d);
- unknown identifiers were indicated with red underlining (Figure 1e);
- upon compilation, help bubbles were displayed for compile time errors, explaining the error in language appropriate to novice programmers and providing links to additional help documentation and lecture notes;
- wizards were added to assist in the creation of programs/classes and functions/procedures/methods; and
- programming behaviour events were logged, as well as snapshots of program code when these events occurred.



```
begin
 repeat
    sorted:=true;
                   o Index.count-1 do
    begin
      if index.keys[i].key >index.k
      then
      begin
        temp:=index.keys[i];
      end;
    end;
  unmtil sorted;
end;
```

'until' expected but 'end;' found

a) Code structure error indicator	d) Code structure highlighting blocks
b) Error description	
c) Location where error was detected	e) Identifier error

Figure 1. Screenshot of SimplifIDE Editor Pane [15]

The features of SimplifIDE were customisable on a per user basis, allowing subjects to experience either the default professional program development environment or the pedagogical version. The representative pedagogical program development environment used was SimplifIDE with all features enabled, while the representative professional program development environment only had event logging functionality enabled.

3. EXPERIMENTAL METHODOLOGY

Subjects were chosen from the Introductory Programming module offered at the Nelson Mandela Metropolitan University in the Department of Computing Sciences, in order that they could be considered novice programmers [2]. Additionally, using a pre-experiment assessment, subjects were further classified as high risk or low risk subjects. High risk subjects were predicted to have a final grade of less than 65%, while low risk subjects were predicted to have a final grade of greater than or equal to 65%, where a final grade of 50% was required to successfully complete the module. The reason for partitioning subjects into high risk and low risk is due to the fact that differences between these categories of subjects have been reported in other studies [2].

Subjects were then randomly partitioned into the control group that used the representative professional program development environment and the treatment group that used the representative pedagogical program development environment. Within each group there were additional strata containing high risk and low risk subjects. Care was taken to ensure that there were not significant differences between groups and similar strata between groups in terms of size, biographical data collected and predicted final grades.

Subjects utilised the respective program development environments on a weekly basis over the duration of the module in order to complete practical assignments. The same lecturer, module content, assignments and assessments were administered to confine the differences between groups to being only the program development environment used.

Each week, subjects from both groups were requested to complete a short questionnaire regarding their perceptions of the previous week's practical assignment. Subjects were asked to indicate their feelings on:

- how much they had achieved;
- how much they had learnt;
- how easy the program development environment had been to use; and
- how easy the practical task had been.

The questionnaires consisted of questions that were answered using a five point Lickert Scale. Answers were in the range from 1 to 5, where 1 indicated total disagreement and 5 indicated total agreement. The number of questions asked remained limited to prevent subjects from not answering the questionnaires due to lengthy questionnaires that needed to be completed each week.

4. RESULTS

Data collected during the experiment was in both paper and electronic format. Biographical information was collected using paper-based questionnaires, while perception questionnaire results were collected electronically.

The data was collected and analysed using profile analysis [17] to determine if any differences were evident over the duration of the module. This involved:

- checking if there were any significant differences detected;
- if there were, then the locations of these were determined;
- if there were significant differences, then a check was made to determine if the results of the two groups were parallel (in other words, do both series move in the same direction). A *non-significant* result was required to indicate parallelism and
- if the series were parallel, determine if the series were on different levels of the Lickert scale. A significant difference indicates that the series are on different levels.

If all the tests succeed, then the profiles of the two groups being compared are totally different. If not all the tests succeed, then there is not necessarily a clear distinction between the two groups.

4.1 Sample Population

Ninety eight subjects were in the initial experimental group and were enrolled for the Introductory Programming module. After withdrawals and module cancellations, there were 32 subjects in the control group and 47 in the treatment group (Table 1).

	Control Group	Treatment Group
Initial	44	54
Cancelled	10 (22%)	6 (11%)
Withdrawn	2	1
Final	32	47
Low Risk	15	19
High Risk	17	28

Table 1. Sample Distribution

A slightly larger treatment group was chosen initially to ensure that after cancellations and withdrawals the groups would be as close to one another in size as was possible. It was thought that more subjects would withdraw from the treatment group than the control group since the professional program development environment was the prescribed program development environment for the Introductory Programming module.

The authors noted that very few subjects withdrew from the experiment or cancelled the module in the treatment group. Compared to the control group, only half the proportion of the treatment group subjects withdrew from the experiment or cancelled the module (Table 1).

4.2 Weekly Practical Tasks

Weekly practical sessions were scheduled for the duration of the Introductory Programming module. At the beginning of each practical session, subjects were asked to complete a questionnaire reflecting on the previous week's practical session. The responses for practical sessions 9 and 10 were combined due to public holidays that occurred during the two weeks. As the module progressed, the complexity of the practical assignments increased and the last three practical assignments (9+10 and 11) consolidated topics of the entire module.

Subjects were asked whether they felt that they had *achieved* “something” during that practical, a low rating indicated that they felt they did not achieve anything, while a high rating indicated that they felt they achieved a lot. This was a measure of whether subjects felt that they had successfully completed tasks.

Subjects in the high risk strata responded at a slightly lower level than those in the low risk strata for both groups, but there were no significant differences between strata within the control group (p=0.2) and treatment group (p=0.16). Since there were not significant differences between the high risk and low risk strata, only the results on a per group basis are shown for brevity in Figure 2. Significant differences are indicated with shaded blocks, with the p value shown above.

Profile analysis of the per group responses indicated that there were differences between multiple variables (p=0.03), specifically practicals 9+10 (p=0.012) and practical 11 (p=0.003). The responses of each group were parallel (p=0.363) and were not on the same level (p=0.010). Therefore the treatment group and the control group had significantly different perceptions about the level of achievement attained during the practical sessions. Subjects in the treatment group consistently felt that they achieved more than those in the control group, whose feelings of achievement gradually decreased over the duration of the module.

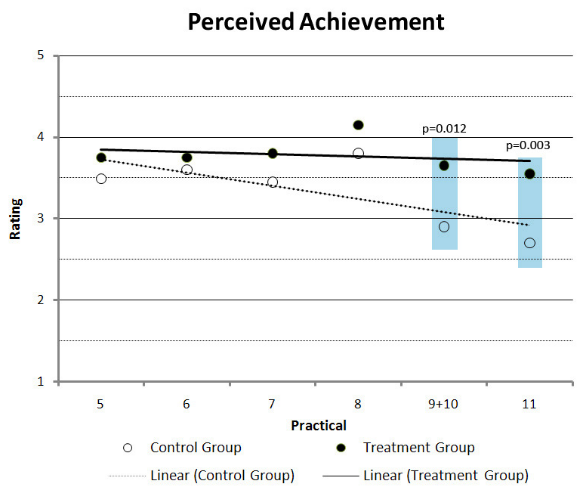


Figure 2. Perceived Achievement during Practical Sessions

Subjects were asked if they felt that they *learnt* something during the week's practical tasks. A low rating indicated that subjects felt they did not learn anything, while a high level indicated that they felt they learnt a lot.

The responses of subjects in the high risk strata of each group were slightly higher than those of subjects in the low risk strata of the same group, but not significantly different for the control group (*p*=0.17) or treatment group (*p*=0.22). The results on a per group basis only are shown in Figure 3.

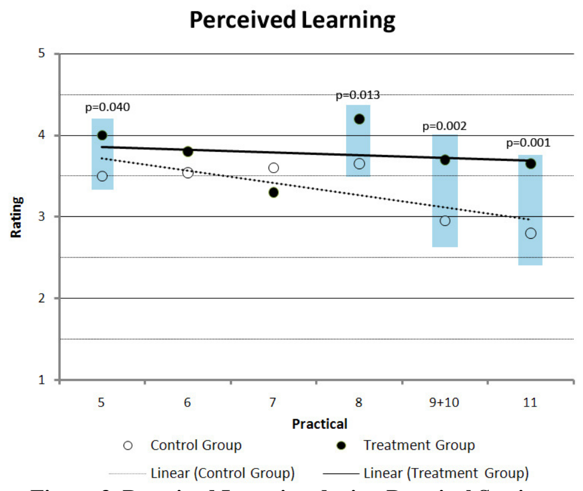


Figure 3. Perceived Learning during Practical Sessions

Profile analysis indicated that there were significant differences between multiple variables (*p*=0.002), specifically practical 5 (*p*=0.040), practical 8 (*p*=0.013), practicals 9+10 (*p*=0.002) and practical 11 (*p*=0.001). The responses of the control group and treatment group were not parallel however (*p*=0.016), due to the line segment between practicals 7 and 8. In practical 7, the responses of the control group were higher than that of the treatment group.

Unlike the perceptions of achievement, there is not a clear separation between the control and treatment group's perceptions on learning. There is strong evidence, however, to suggest that subjects in the treatment group have a much more positive perception of having learnt something than the control group (4 out of 6 practical assignments were significantly different). Similar to the perceived achievement, subjects in the treatment group had a fairly consistent perception of learning during the practical tasks, while subjects in the control group had a gradual decrease in the perceptions of learning over the duration of the module.

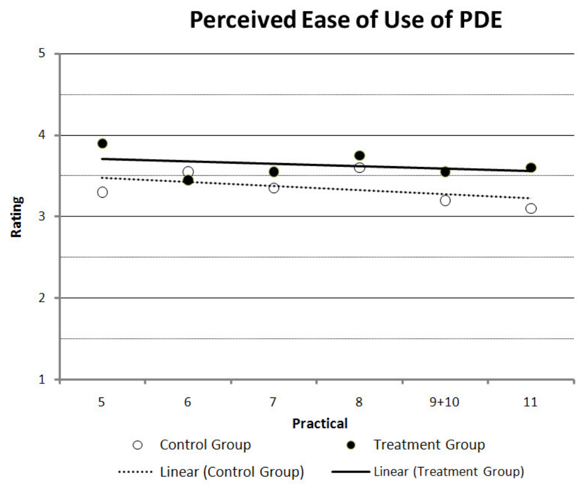


Figure 4. Perceived Ease of Use during Practical Sessions

Subjects were asked to indicate how *easy they found the designated program development environment to use*. A low rating indicated that the program development environment was difficult to use, while a high rating indicated that it was easy to use.

The results of the subjects perceived ease of using the designated program development environment on a per group basis are shown in Figure 4. Once again there were no significant differences on a per strata basis in the control group (*p*=0.56) and treatment group (*p*=0.18), therefore only the results on a per group basis are shown.

Profile analysis indicated that there were no significant differences amongst variables (*p*=0.122). Thorough examination of Figure 4 shows that the responses are very close to one another, although it would appear that subjects in the treatment group found the program development environment slightly easier to use than the control group. Both the treatment and control group had a gradual decrease in the perception that the designated program development environment was easy to use over the duration of the module.

Subjects were also asked to indicate how *easy they thought the practical assignment was*. A low rating indicated that they thought the practical assignment was difficult and a high rating that it was easy.

The results of the subjects perceived ease of completing the practical tasks on a per group basis are shown in Figure 5. Once again there were no major differences on a per strata basis in the control group (*p*=0.2) and treatment group (*p*=0.23), therefore only the results on a per group basis are shown.

Profile analysis showed that there were significant differences amongst variables (*p*=0.038), specifically practical 5 (*p*=0.025). The responses of the control group and treatment group were found to be parallel (*p*=0.05), but also on the same level (*p*=0.316). Therefore the perceived ease of the practical tasks was the same for both groups and the program development environment used did not make a significant difference. There is a gradual drop in the perceived level of easiness of the practical assignments by both the treatment and control groups over the duration of the module.

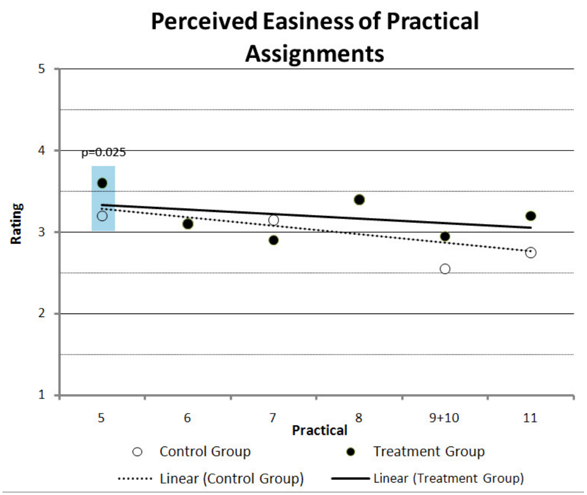


Figure 5. Perceived Ease of Practical Assignments during Practical Sessions

Thorough examination of all the responses indicates that in most cases the treatment groups mean responses were higher than that of the control group, except for practical 7. During that particular week, there were technical issues with the local area network which lead to slow response times. This could have resulted in the lower ratings by the treatment group who were more affected by the slow response times than the control group. A number of treatment group subjects commented about the slow response times and occasional crashes during this practical session.

5. DISCUSSION

Considering all perceptions, it appears that for both groups, perceptions started off on relatively positive levels. Most of the perceptions exhibited a gradual decrease in positivity, moving toward neutrality (a rating of 3 on the Lickert scale), as the module progressed. This phenomena is not unexpected as the complexity of content increases as the module progresses.

There were no significant differences between the perception levels of high risk and low risk subjects, unlike the related study [15]. Therefore the ability level of the novice programmer did not significantly affect their perceptions during the learning process.

The perceived level of achievement attained during practical assignments remained relatively constant for subjects in the treatment group over the duration of the module. Subjects in the control group, however, had a steady decrease in the perceived level of achievement as the module progressed, with the mean perceived level of achievement dropping below neutral for the last practical assignment. The last two practical assignments, in particular, exhibited significant differences between the perceived levels of achievement for the control group and the treatment group. Profile analysis indicated that the responses of each group over the period were different, therefore the use of a pedagogical program development environment while learning to program can have a *significant* effect on the perceived level of achievement of novice programmers while learning to program.

The perceived level of learning that occurred during practical assignments was not shown to be completely separate using profile analysis. Considering the number of significant differences detected (4 out of 6), there is strong evidence to suggest that the program development environment used has an impact on the perceived level of learning. For subjects using the pedagogical program development environment, the perceived level of learning remained relatively constant, while those using the professional program development environment exhibited a gradual decrease in perceived learning levels as the module progressed. Therefore it would appear that the use of a pedagogical program development environment can have a positive effect on the perceived level of learning of novice programmers learning to program.

The perceived ease of use of program development environments by both groups was not significantly different. Subjects using the pedagogical program development environment had slightly higher mean levels of perceived ease of use, with the largest difference occurring during the first practical. One possible explanation of this is that subjects in each group were novices, not having programmed or been exposed to other program development environments before, therefore they did not have a benchmark with which to compare their designated program development environment against. Another possible explanation is that since both the pedagogical and professional program development environments used in the study were basically text-based editors with support features, it is possible that program development environments built around text-based editors are of similar ease of use when programming. Other metaphors of programming, such as iconic programming [2], or different support features might be able to change the perceived ease of use.

The perceived level of easiness of practical assignments was not significantly different for the control and treatment groups, in fact they are considered identical from the profile analysis conducted. In addition to this, the perceived level of easiness dropped as the module progressed and the module content became more complex. Therefore it appears as if the program development environment used does not affect how easy or difficult novice programmers consider solving a problem to be. As with the perceived ease of use of the program development environment, a reason that the perceived level of easiness was the same might be the fact that both environments were text-based editors and inherently employ similar procedures and processes. Another reason could be that neither of the program development environments provided tools to assist novice programmers in creating a solution to a problem, instead they allowed novice programmers to simply to write, compile and debug programs with no assistance during the problem solving phase.

6. CONCLUSIONS

The motivation and self-belief of novices learning to program are important aspects that should not be ignored during the learning process. This study has shown that pedagogical program development environments are beneficial to the perceptions of novice programmers learning to program, specifically the feelings of achievement and learning. The program development environments used did not have any significant effect on the perceived ease of using the program development environment or the ease of completing practical assignments.

More positive perceptions about programming and IT in general, could help alleviate some of the problems experienced in Introductory Programming modules worldwide. Therefore it is important that appropriate program development environments be used when learning to program.

REFERENCES

[1] Clear, T., Edwards, J., Lister, R., Simon, B., Thompson, E. and Whalley, J. 2008. The teaching of novice computer

programmers: bringing the scholarly-research approach to Australia. In Proceedings of the 10th Conference on Australasian Computing Education, vol. 78, pp. 63–68.

[2] Cilliers, C.B., Calitz, A.P. and Greyling, J.H. 2005. The Effect of Integrating an Iconic Programming Notation into CS1. In Proceedings of ITiCSE'05.

[3] Hanks, B. and Brandt, M. 2009. Successful and unsuccessful problem solving approaches of novice programmers. In Proceedings of the 40th ACM technical symposium on Computer Science Education, pp. 24–28.

[4] Reis, C. and Cartwright, R. 2004. Taming a Professional IDE for the Classroom. In Proceedings of SIGCSE'04, pp. 156–160.

[5] Nienaltowski, M.H., Pedroni, M. and Meyer, B. 2008. Compiler Error Messages: What Can Help Novices? In Proceedings of the 39th SIGCSE Technical Symposium on Computer Science Education.

[6] Kelleher, C. and Pausch, R. 2005. Lowering the Barriers to Programming: A Taxonomy of Programming Environments and Languages for Novice Programmers. ACM Computing Surveys, vol. 37, no. 2, pp. 83–137.

[7] Ko, A.J. 2003. A Contextual Inquiry of Expert Programmers in an Event-Based Programming Environment. In Proceedings of CHI2003, pp. 1036–1037.

[8] Bloch, S. and Proulx, V. 2007. Teach Scheme, reach Java: introducing object - oriented programming without drowning in syntax: tutorial presentation. Journal of Computing Sciences in Colleges, vol. 22, no. 6, pp. 88–89.

[9] Kolling, M., Quig, B., Patterson, A. and Rosenburg, J. 2003. The BlueJ System and its Pedagogy. Journal of Computer Science Education, vol. 13, no. 4.

[10] van Tonder, M., Naude, K. and Cilliers, C.B. 2008. Jenuity: a lightweight development environment for intermediate level programming courses. In Proceedings of 13th Annual Conference on Innovation and Technology in Computer Science Education.

[11] de Pascuale III, P. 2003. Implications on the Learning of Programming Through the Implementation of Subsets in Program Development Environments. Doctoral thesis, Virginia Polytechnic Institute and State University.

[12] McIver, L. 2002. Evaluating Languages and Environments for Novice Programmers. In Proceedings of 14th Workshop of the Psychology of Programming Interest Group, pp. 100–110.

[13] Zimmerman, B.J. 1995. Self-Efficacy and Educational Development. Self-Efficacy Changing Societies, pp. 203–231.

[14] Wiedenbeck, S. 2005. Factors Affecting the Success of Non- Majors in Learning to Program. In Proceedings of ICER'05.

[15] Vogts, D., Calitz, A.P. and Greyling,J.H. 2008. Comparison of the effects of professional and pedagogical program development environments on novice programmers. In Proceedings of SAICSIT 2008, pp. 286–295. (Oct 2008). DOI: http://doi.acm.org/10.1145/1456659.1456692.

[16] Vogts, D. 2007. The Evaluation of a Pedagogical Program Development Environment for Novice Programmers: A Comparative Study. Doctoral thesis, Nelson Mandela Metropolitan University.

[17] Stevens, J.P. 2009. Applied Multivariate Statistics for the Social Sciences. 5th edn.