Choosing the "Best" Programming Language?!

Leila Goosen
University of Pretoria,
South Africa

lgoosen@gk.up.ac.za

Elsa Mentz and Hercules Nieuwoudt
North-West University,
South Africa

snsem@puk.ac.za
nsohdn@puknet.puk.ac.za

Abstract

This paper starts out by outlining the results of a literature study used to create a list of criteria to choose a first programming language for high schools. The researchers made every effort to ensure the relevance of selection criteria. The study established criteria regarding the development of thinking and programming skills, requirements for the programming environment to make it appropriate for learners, new tendencies in programming, issues influencing programming used in practice, affordability, training and resources, and programming for various purposes. A questionnaire based empirical study verified the validity of selection criteria identified in the South African context, with respondents rating criteria according to importance and application. Differences between the reported importance and application of criteria show effect sizes with practical significance. These differences underline important issues applicable to the world of the teachers who will use a chosen language.

Keywords: Selection Criteria, First Programming Language, High Schools, Information Technology

Introduction

Since computers have made their way into educational systems around the world, one of the predominant uses of computers, especially at high school level, is the instruction of learners in various programming languages (Palumbo & Reed, 1991). Several languages are available for teaching programming at an introductory level, and many good arguments are available in favor of adopting any one of the many possible styles of programming in a first course (Ali & Kohun, 2005). The language of choice for introductory programming courses, however, remains controversial, with the problem of selecting an implementation language all too often taking "on the character of a religious war that generates far more heat than light" (SIGCSE, 2001).

The argument should address issues like learners' needs, and the question of which programming language would be most beneficial to learners' development (Palumbo & Reed, 1991). Computer and software performance show rapid and spectacular progress (Garner, 2003; Mehic & Hasan, 2001). The decision on which tool(s) to introduce into the classroom for teaching programming should therefore be taken with great care, as choosing the wrong implementation(s) could mean that not only its use, but also its teaching will be outdated very quickly. Even if one ignored limitations such as support, costs, training etc., it quickly becomes clear that making the choice of which programming language to use at an introductory level, needs to be done in a scientific way by utilizing a wide variety of criteria to help in the decision (Ali & Kohun, 2005).

Prior to the study reported on here, no criteria were evident for selecting a first programming language to teach in high schools. Extensive national and international searches found very little evidence of specific research into this area. Although the purpose of a study by Mehic and Hasan (2001) was to choose a suitable first language for computer science students, the intended population was different and it had very limited relevance with regard to suitable criteria. The aim of the research reported on in this paper therefore was to establish criteria to consider when selecting a first programming language to teach in high schools.

The following section presents a closer examination of the selection criteria as developed from the literature study. Hereafter, the research method is described. An empirical study verified the validity of selection criteria identified in the literature study within the South African context. The results section reports on the outcomes of this empirical study. Distinguishing patterns of differences between the ways in which policymakers, teachers and trainers respectively rated the importance and application of factors occurred for inferential statistical methods. A discussion of the implications of combined results for selecting a first programming language in high schools follows. In light of these findings, we make suggestions to consider regarding further investigations, where applicable. Most of the limitations of this study would relate to the relevance of selection criteria to a specific situation other than the South African context used – we address these first in the next section.

Literature Perspectives

Relevance of Selection Criteria

With regard to the relevance of criteria established for the selection of a first programming language for high schools, we could place emphasis on setting selection criteria that

- will be universal, in that these will have worldwide relevancy in all circumstances
- would be relevant to select a first language for a reasonable period and
- although general in terms of worldwide relevancy, should, in light of the context of this study, specifically suit needs relevant to the South African context.

Developing Thinking and Programming Skills

The needs, knowledge and abilities of learners as novice computer users are significantly different from those of experienced programmers (Garner, 2003). At high school level, learners’ cognitive development levels are growing towards the formal operational stage (White & Sivitanides, 2002). In line with the stated purpose of the subject Information Technology (Department of Education [DoE], 2003a), it can be argued that the subject serves as a problem solving discipline and a tool for thinking and reasoning, centered on the development of solutions to problems (Ali & Kohun, 2005). The most important outcome is therefore the introduction of general problem solving concepts, rather than focusing on teaching the syntax of a specific programming language (Al-Rawi, Lansari, & Bouslama, 2005).

A person's ability to think critically is an important and all-encompassing educational outcome, and it is a key success factor in many kinds of occupations (DoE, 2003a). This explains growing concern for promoting the development of critical thinking skills among learners. Improvements in learners’ understanding of their own thought processes also increases their ability to develop higher order thinking skills and abilities for regulating themselves while solving problems. The language should therefore facilitate the use of the meta-cognitive components of problem solving during instruction.

Due to the didactical principles involved in the teaching and learning of the subject, we should consider whether a particular language used in a first programming course complies with educational aims with regard to preparing learners to have a firm foundation in good programming practices for well-designed programs (Ali & Kohun, 2005). Another primary purpose of using a specific language should be to teach underlying programming principles (DoE, 2003b), as in a field that changes so rapidly, skills quickly date and programming competencies alone are no longer adequate (Barrow, Gelderblom, & Miller, 2002). In addition, the acquisition of the procedural knowledge features of programming, which represent cognitively more demanding skills (White & Sivitanides, 2002), is dependent on programming abstractions (Mehic & Hasan, 2001), such as procedure and data abstraction and top-down design with step-wise refinement.

Learner-Appropriate Programming Environments

Literature presents several arguments in favor of using a language with a high level of control and structure (Walker, 2000). Compiling and coordinating the different parts of a program are some of the greatest obstacles novice programmers face (Garner, 2003). An environment that supports these learners' tasks when designing and implementing their programs can simplify a lot of their work (Garner, 2003). Learners also need to identify and fix problems that they may have within their programs. Their environments should therefore contain compilers that supply well-defined and easily understandable error messages and offer effective debugging tools to locate and correct errors (Ali & Kohun, 2005).

A simple and clear context can promote learners' development of high-level thinking skills (Palumbo & Reed, 1991). It is therefore important that the programming language and its software development environment used at high school level should be suitable for novice programmers (Garner, 2003). Together, they should be easy to learn, sufficiently simple to work with (DoE, 2003b) by offering relative simplicity of commands (Ali & Kohun, 2005), should not frustrate learners because of unnecessary difficulties and features suited to professional programmers (Mehic & Hasan, 2001), or be too expansive.

New Tendencies in Programming

Object-oriented programming (OOP) is one of the presently accepted problem solving and programming paradigms used (DoE, 2003a). Lambert and Nance (1998, p. 30), however, warn that object-oriented design "is not necessarily the easiest and most straight-forward way to learn to solve problems on the computer." Both SIGCSE (2001) and Mehic and Hasan (2001) point to complications that can turn up when using OOP, and the difficulties learners face when learning to program in an object-oriented style. Using an objects-based model also increases the risk of certain didactical problems occurring. Teachers therefore need to exercise special care in introducing OO material in a way that limits the detail involved in many object-oriented programming languages, as this detail could easily overwhelm learners.

Offering learners the opportunity to make use of visual programming language (Ali & Kohun, 2005) provide them with a visual vocabulary that they can apply and understand immediately, offsetting some of the difficulties with regard to OOP mentioned in the previous paragraph.

Using techniques such as encapsulation and inheritance provide for a means of organizing software systems in industry. The use of these technologies might not be necessary at school level, as learners normally do not build such large systems. Especially teachers might not yet be very familiar with these concepts, as these occur in the curriculum towards the end of the last school year - in provinces where languages have only recently been implemented this content might not yet be covered. The need to use these techniques will also not become apparent until user-defined objects, considered a difficult topic, are implemented (Lambert & Nance, 1998). A coping strategy that teachers sometimes employ (Manouchehri & Goodman, 1998) entails concentrating on areas in which their confidence is highest. At the same time, they avoid those subject units that they do not view as important, or those they do not feel comfortable teaching. This discomfort could be due to them not being familiar with the topic, or because they regard as it as difficult.

Issues Influencing Programming Used in Practice

In terms of the software development process, the chosen programming language should have reasonable prospects for continued development (DoE, 2003b) from its developers. It should also have sufficient capacity for database connectivity, as a great deal of programming revolves around programming for databases.

The international standardization of a programming language, together with international trends with regard to programming languages used in high school education in other parts of the world (DoE, 2003b; Walker, 2000), could be considered. When selecting a first programming language for high schools, the following needs to be born in mind with regard to the popularity and increasing demand from industry for specific languages (Mehic & Hasan, 2001):

- The commercial use and/or popularity of particular languages are not essential for selection when learning programming.
- The purpose of the subject in South Africa is not to provide training for industry.
- Many of the languages used for object-oriented programming in industry involve significant complexity (SIGCSE, 2001).

According to Barrow et al. (2002), the trend nowadays is towards the development of systems for the Internet, as a large part of commercial application development involving programming for the Internet and the World Wide Web. The opposite opinion would be that most programming for businesses is for traditional applications, such as salary systems (Mehic & Hasan, 2001). A consideration should therefore be whether the language is suitable and has capabilities for working on and with the Internet to facilitate electronic communication.

Affordability, Training and Resources

An important selection criterion with specific reference to the South African context involves the affordability of the chosen programming language. The financial situation of schools makes it necessary to consider whether schools or education departments will carry the cost for the upgrade of hardware and software (DoE, 2003b). The costs associated with the acquisition of the software required, including the programming language, IDE and database, should be within reasonable reach. In this regard, one ought to consider options in Open Source software.

Teachers new to the subject need to be able to learn the language used in schools during their pre-service training at local tertiary institutions (DoE, 2003b). Resources appropriate to an outcomes-based education (OBE) approach to the subject should be available to teachers, as the implementation of resource materials should smooth the progress of learners towards the achievement of outcomes (DoE, 2003a).

Programming for Various Purposes

In the South African context, the programming language curriculum taught is of a general nature (DoE, 2003b). The language used should therefore be a general-purpose programming language that supports many academic and commercial tools – it should not have been developed for a certain setting, such as scientific or commercial environments.

Research Method

After establishing the criteria as described in the previous section, an empirical study verified validity in the South African context (Goosen, 2004).

Population and Sampling

The population for the study consists of all role players in the curriculum process of Information Technology, who can be involved in various sectors connected to the subject. The 18 **policy makers** (15% of total number of respondents) consisted of members of the Writing Committee for the National Curriculum Statement for Information Technology, curriculum planners and advisors in all provinces and persons involved with Information Technology at provincial level in an administrative capacity. Nineteen (15%) **trainers** based mainly at tertiary institutions as researchers and/or teaching methodologists or subject matter experts, indicated involvement in the training of Information Technology teachers. Both these sectors of the population were small enough to engage all members in the study.

The number of **teachers** from all schools offering the subject (higher grade and/or programming) [N=445] would not only make it difficult to manage all potential respondents, but the size of this sector of the population relative to other sectors is also disproportionately large. We therefore decided to apply sampling to this sector, by contacting only teachers selected as markers for Grade 12 papers, based on their competence, experience and knowledge of subject matter. In three South African provinces, the Eastern Cape, Free State and Northern Cape, due to small numbers of learners, all marking is done by one person each – in these cases, competent, experienced teachers, recommended by Subject Advisors or Chief Examiners, were approached. This sampling resulted in 87 (70%) teachers making up the remainder of respondents.

Design and Instrumentation

The researchers carried out a quantitative field survey by means of a questionnaire, developed specifically for this study, containing Likert type responses and shorter answers to open questions.

The questionnaire presented respondents with 41 statements regarding selection criteria (see Table 1 for statements), and asked them to indicate how important it would be to use each of these criteria for selection when choosing a programming language for South African high schools. The scale used was (1) Not important at all, (2) Fairly unimportant, (3) Fairly important, and (4) Very important.

However, although some or all of the selection criteria listed might be important, it might not be practical to use some of these selection criteria when selecting a programming language. The questionnaire therefore asked respondents to indicate to what extent each of the selection criteria have been applied to select programming languages for South African high schools. Scaling used: (1) Not applied at all, (2) Sometimes applied, (3) Usually applied, and (4) Always applied.

Results

Table 1 represents a summary of results obtained for importance and application, in terms of the mean for each statement, and rankings based on averages. Access to complete results, including breakdown in terms of scaling, and standard deviations, are available at Goosen (2004). The latter also provides access to quotes from respondents in the comments sections of the survey.

Table 1: Results for selection criteria

	Importance		Application	
	Mean	№	№	Mean
Relevance of selection criteria				
Have worldwide relevance.	3.39	33	**26**	2.60
Be relevant for a reasonable period.	3.39	34	**22**	2.65
Suit specific needs relevant to the South African context.	3.32	39	**25**	2.60
Developing thinking and programming skills				
Provide an instructional environment promoting development of problem solving abilities.	3.83	**1**	**1**	3.04
Support good programming style.	3.73	5	**2**	3.02
Provide an instructional environment promoting development of critical thinking skills.	3.65	12	**6**	2.88
Encourage a self-regulated approach to solving problems.	3.62	14	**12**	2.81
Provide an instructional environment promoting development of higher order thinking skills.	3.59	15	**7**	2.88
Facilitate the development of learners’ understanding of their own thought processes.	3.55	20	**19**	2.68
Promote top-down design with step-wise refinement.	3.41	31	**7**	2.88
Encourage programming principles such as abstraction.	3.38	36	**18**	2.69
Adequately match the abilities of learners.	3.37	37	**16**	2.75
Requirements of programming environment for learners				
Be safe, stable, structured and controlled.	3.70	9	**3**	3.01
Provide understandable error-messages.	3.68	10	**4**	2.90
Provide effective debugging tools.	3.65	13	**5**	2.88
Be easy to learn.	3.56	19	**11**	2.81
Not frustrate learners because of features suited to professional programmers.	3.54	21	**13**	2.79
Offer relative simplicity of commands.	3.45	26	**9**	2.86
Suit the needs of novice programmers.	3.44	28	**14**	2.77
New tendencies in programming				
Offer possibilities for object-oriented design.	3.70	**8**	10	2.83
Support new tendencies in programming.	3.57	17	**15**	2.75
Offer possibilities for visual programming.	3.48	24	**20**	2.67
Offer possibilities for encapsulation.	3.43	30	**23**	2.62
Offer possibilities for inheritance.	3.38	35	**27**	2.58
Offer possibilities for polymorphism.	3.24	40	**28**	2.57
Issues influencing programming used in practice				
Have reasonable prospects for continued support from its developers.	3.72	**6**	17	2.70
Support database programming.	3.58	**16**	21	2.66
Follow international trends regarding programming languages used in high-school education.	3.45	**25**	36	2.41
Be popular in industry, where there is a demand for such programmers.	3.44	**27**	32	2.47
Be internationally standardized	3.44	**28**	30	2.55
Be compatible with tertiary establishments’ choice of programming language.	3.41	**32**	35	2.43
Enable Internet programming.	3.24	41	**40**	2.26
Affordability, training and resources				
Be affordable.	3.76	**2**	29	2.57
Enable affordable in-service training of CS teachers.	3.75	**3**	39	2.35
Have learning and teaching support materials and resources available.	3.74	**4**	38	2.39
Have particularly textbooks suited to learning the language at high school level available.	3.71	**7**	33	2.45
Be included in the pre-service training of CS teachers.	3.65	**11**	37	2.41
Have resources appropriate to an OBE approach to the subject obtainable.	3.50	**22**	41	2.26
Programming for various purposes				
Support programming for various purposes.	3.56	**18**	24	2.61
Be able to be used with academic tools.	3.48	**23**	31	2.51
Be able to be used with commercial tools.	3.32	38	**34**	2.45

Relevance of Selection Criteria

According to the respondents in the empirical study, the three items in the questionnaire regarding the relevance of selection criteria comprised some of the least important issues, with averages placing these in the lowest quarter of results – specifically the item regarding suiting needs relevant to the South African context has the third lowest average of all items. In terms of application, the item with regard to relevance for a reasonable period received an average slightly higher than those for the other two items, but all three items still place in the lower half of results, indicating that these were also not applied largely.

Developing Thinking and Programming Skills

Compared to other items regarding selection criteria in the questionnaire, respondents felt that the adequate matching of the adopted programming language to the abilities of learners of Information Technology with regard to both level and nature is one of the least important items. Its average for application however places it in the top half of items.

Respondents strongly agree that the chosen language should provide an instructional environment that promotes the development of problem solving skills, as the averages for this item distinguish it as both the most important and most applied item.

Items regarding the development of higher order and critical thinking skills, as well as encouraging a self-regulated approach to solving problems by promoting strategies for analyzing programming problems and formulating solutions to them, achieved averages placing them in or close to the top third of items for both importance and application.

The average for an item that refers to the development of learners' understanding of their own thought processes and their involvement in meta-cognitive behavior, places it in the middle relative to others with regard to both importance and application.

Respondents supported the opinion that the language should support good programming style, as an item in this regard applied second most and is the fifth most important selection criterion.

"If programmers were well trained in basic principles, they would be able to implement these principles regardless of what language they have to use to do that" (quote from respondent in comment section of questionnaire.) Encouraging programming principles such as procedure and data abstraction and promoting top-down design with step-wise refinement are some of the least important items, but were applied much more, with the latter applied more extensively.

Learner-Appropriate Programming Environments

The item referring to providing learners with a safe, stable, structured and controlled programming environment, together with items regarding the environment providing learners with understandable error-messages and effective debugging tools, obtained the third, fourth and fifth highest averages for application. Although these three items received averages for importance that place them slightly lower, they still place within the top third of results.

Items with regard to the programming language and its software development environment being suitable for novice programmers in that it is easy to learn, offers relative simplicity of commands and does not frustrate learners because of features suited to professional programmers, are not considered as important, as the averages for these items place them in the lower half of results. However, these items were applied to a greater extent than would be expected for low importance, with application averages placing them in or close to the top third of results.

New Tendencies in Programming

The chosen programming language offering possibilities for object-oriented design received averages for importance and application placing it in the top quarter of results for both these sections.

Offering learners the opportunity to make use of visual programming received averages for importance and application placing this item just inside the lower half of results.

The concepts of encapsulation, inheritance and polymorphism were considered to be some of the least important selection criteria, with averages placing them in the lowest quarter of results. These items, although applied slightly more, still obtained averages placing them in the lower half for application.

Issues Influencing Programming Used in Practice

The chosen programming language having reasonable prospects for continued development from its developers has an average that places it as the sixth most important selection criteria, while its average for application is in the higher half of results. Whether or not a language has sufficient capacity for database connectivity also places in the higher half of results for both importance and application.

The international standardization of a programming language places in the lower half of results for importance and in the lowest quarter for application. Considering the popularity and/or demand for specific languages in industry when selecting a first programming language for high schools received similar results for importance and application. The language enabling Internet programming is the least important selection criteria and was applied second least.

Affordability, Training and Resources

Respondents considered the affordability of the chosen programming language to be the second most important item of all. However, considering the affordability of a chosen programming language was applied so little that it places at the lowest quarter of results.

The availability of affordable, sufficient in-service teacher training is considered the third most important item, but inversely, it obtained the third lowest average! Respondents’ answers to open-ended questions in the questionnaire explained these seemingly opposing opinions to some extent. Teachers mention a province where no teacher training was offered in the language being implemented, while four refer to the retraining of educators, three of whom bring up the cost involved in retraining – they feel that these costs should be fully borne by education departments, and not by teachers, as had already occurred in some cases. They also express the feeling that education departments should be responsible for arranging training opportunities.

The ability of teachers new to the subject to learn the language used in schools during their pre-service training at local tertiary institutions, and having learning and teaching support materials and other resources, particularly textbooks, for the language available to teachers, received averages that place them as the eleventh, fourth and seventh most important items. However, the averages for these items for application are well within the lowest quarter of results, with the second item mentioned in this paragraph specifically obtaining the fourth lowest averages in this regard. Although having resources appropriate to an OBE approach to the subject available to teachers has an average placing it in the middle of results, it was in fact applied the least.

Programming for Various Purposes

Using a general-purpose programming language that supports programming for various purposes received averages for both importance and application placing it inside the lower half of results.

The same goes for the average of the importance of being able to use the language with academic tools. The average for the application of the latter, as well as the application average for being able to use the language with commercial tools, places these in the lowest quarter of results, while being able to use the language with commercial tools is the fourth least important item amongst selection criteria.

Summary: Importance and Application of Selection Criteria

The empirical study confirmed the validity of all selection criteria identified, as all items obtained averages for importance that classify them as 'fairly important'. However, the extent to which criteria was applied for language selection was fairly low, with only three items obtaining averages classifying them as 'usually applied' - all other items were only 'sometimes applied'. The correspondence between the rated importance of specific items and the extent to which each was applied varied also widely.

Factor Analysis and Significance of Differences

A factor analysis carried out on statements in the questionnaire with regard to the application of selection criteria yielded seven factors, which explained 75% of variance. These factors were then rotated according to the Vari-max rotation method. As all final communality estimates are larger that 0.5, the factors can be considered to be a successful extraction. Calculation of Cronbach Alpha scores also confirmed the reliability of factors as measuring instruments. For complete results regarding variance and Cronbach Alpha scores see Goosen (2004).

Inferential statistics were used in order to establish whether there were distinguishing patterns of differences between the ways in which different groups of respondents – policymakers, teachers and trainers (see subsection on population and sampling under research method section) - respectively rated the importance and application of factors. The formula for calculating effect sizes

$$d = \frac{\left|\overline{x}_i - \overline{x}_j\right|}{\sqrt{MSE}}$$

(Steyn, 2000) uses the difference between each pair of group means, divided by the square root of the mean square error (MSE) of analysis of variance as obtained from ANOVA test. Values of the F-ratio exceeds the critical value at 10% probability levels ($p < 0.1$) for all factors, indicating that there is a statistically significant difference between at least one pair of means. Cohen (1988) was used to interpret effect size – these are portrayed in Table 2.

Table 2: Interpretation of effect size (d)

Value of d	Indicator	Significance of effect size
$d \approx 0.2$	*	Small effect, no practical significance
$d \approx 0.5$	**	Medium effect - difference might be of practical significance
$d \approx 0.8$	***	Large effect - difference large enough to have an effect in practice

When the differences between the ratings that respondents gave to items with regard to importance and application respectively for each factor were calculated, grouping of respondents as policy makers, trainers or teachers reveal less significant differences between importance and application for policy makers and trainers regarding new tendencies in programming and the relevance of criteria (see Table 3). Policy makers also show less significant differences with regard to issues influencing programming used in practice.

Table 3: Significance of differences between importance and application of factors

Factor	Policy makers		Teachers		Trainers	
Relevance of selection criteria	0.75	**	0.95	***	0.65	**
Developing thinking and programming skills	0.92	***	1.12	***	1.05	***
Learners’ programming environment requirements	0.87	***	1.05	***	1.37	***
New tendencies in programming	0.50	**	0.95	***	0.54	**
Issues influencing programming used in practice	0.72	**	1.31	***	0.89	***
Affordability, training and resources	0.83	***	1.90	***	1.47	***
Programming for various purposes	0.92	***	1.24	***	0.92	***

Discussion

Relevance of Selection Criteria

Use of selection criteria established in this study in situations other than the South African context applicable to this study, might make it necessary to reconsider some of them. Many aspects contemplated within the South African context, including financial considerations, training and support of teachers and resources available to teachers, would probably remain valid for most situations. Consideration of other criteria, such as those regarding the OBE orientation of resources, general-purpose programming, demand and training for industry and tertiary establishments and their expectations, would depend on the applicable context. Finally, the relevance of criteria in the ‘new tendencies in programming’ section would depend on the amount of time that has elapsed since the completion of the study reported on in this paper.

Developing Thinking and Programming Skills

Despite the low importance awarded to the item regarding the adequate matching of the adopted programming language to the abilities of learners of Information Technology, its application corresponds well with literature perspectives relating to differences between learners and experienced programmers.

Respondents agree that the chosen language should provide an instructional environment that promotes the development of higher order thinking and problem solving skills, as well as critical thinking. Specifically, the averages for the item regarding the development of problem solving skills distinguish it as both the most important and most applied item. These findings not only provide support for continued programming language instruction at high school level, but also offer a means for improving the problem solving skills for the group most often criticized for their lack of problem solving ability (Palumbo, 1990).

It is possible that the low importance afforded to abstraction and the promotion of top-down design (Mehic & Hasan, 2001) by respondents might be due to some respondents not being aware of the integral role that abstraction plays in OOP (Miah, 1997). They could also be under the impression that using top-down design forms part of the 'old' way of programming. These notions could be examined in subsequent investigations, and if proven to have substance, need to be addressed during training sessions.

Learner-Appropriate Programming Environments

Results for the item referring to learners being provided with a safe, stable, structured and controlled programming environment agree with arguments presented in literature - together with items regarding the environment providing learners with understandable error-messages and effective debugging tools, the third, fourth and fifth highest averages for application was obtained.

Items with regard to the programming language and its software development environment being suitable for novice programmers in that it is easy to learn, offers relative simplicity of commands and does not frustrate learners because of features suited to professional programmers were applied to a greater extent than would be expected for low importance. Application results conform much better with recommendations from literature.

New Tendencies in Programming

Comparison to literature perspectives could explain results for encapsulation, inheritance and polymorphism. Further investigations could be launched to explore whether teachers avoid these concepts, because they are not viewed as important, or because they do not feel comfortable teaching them, as they are not familiar with them or regard them as difficult. The extent to which encapsulation, inheritance and polymorphism would eventually be used as part of selection criteria would depend on the outcomes of such investigations, as well as the importance afforded to the employment of user-defined objects by role players in a curriculum project.

Issues Influencing Programming Used in Practice

The language enabling Internet programming is the least important selection criteria and was applied second least, which confirms literature perspectives regarding the over-estimation of this capacity.

Affordability, Training, and Resources

In spite of many arguments in literature connected to the affordability of the chosen programming language, as well as the importance afforded this item by respondents, application was very low. It is of particular concern how little all criteria relating to affordability, training and resources were actually applied during language selection - in any future situations the importance of these criteria, specifically with regard to the ability of teachers to eventually successfully implement a language, warrants much more careful consideration.

Factor Analysis and Significance of Differences

The fact that there are generally practically significant differences between the rated importance and application of factors point to a situation where policymakers do not apply issues that are important to the majority of stakeholders (teachers) when choosing a programming language. The fact that these differences are smaller for policy makers underlines this impression. It could indicate that policy makers are under the impression that the issues that they deem to be important dictate the extent to which these are applied.

Conclusions

Results in this study arranged selection criteria as being more important relative to others for those with the highest averages, and less important for those obtaining lower averages. During the selection process of a first programming language for high schools, decision makers could pay closer attention to those criteria identified as most important. At the other end of the scale, they could potentially save time and effort by assigning less importance to criteria with lower averages. Perspectives from literature should however also be considered, especially in cases where these differ from the importance as assigned by respondents in this study.

Importance of the Study

“Computer technology develops at an alarming rate” (Barrow et al., 2002, p. v) and the rate at which computer languages develops is much higher than was seen in the past. The cutting-edge computer industry re-tools more frequently and programmers will cover many more languages in their lifetime than ever before. This implies that while learning practical programming skills in a particular programming language, it is even more important for the learner as novice programmer to develop a sound theoretical understanding of programming in general, and so to prepare for later learning future languages and environments. Education must prepare learners for lifelong learning that moves them beyond today’s technology to meet the challenges of the future.

The rapid changes in computer technology and programming languages also leads to the opinion that in five years' time (or even less!) the language issue will have to be re-evaluated and languages changed again. Criteria established in this study could prove to be indispensable in such a case.

As the results in this study provide a set of fairly generic criteria, they will most likely be useful beyond the immediate context portrayed.

References

Al-Rawi, A., Lansari, A., & Bouslama, F. (2005). Integrating Sun Certification objectives into an IS programming course. *Proceedings of the 2005 Informing Science and Information Technology Education Joint Conference, Flagstaff, Arizona, USA*, 33-48. Retrieved April 25, 2007 from http://proceedings.informingscience.org/InSITE2005/I20f39Rawi.pdf

Ali, A.I. & Kohun, F. (2005). Suggested topics for an IS introductory course in Java. *Proceedings of the 2005 Informing Science and Information Technology Education Joint Conference, Flagstaff, Arizona, USA*, 33-48. Retrieved April 25, 2007 from http://proceedings.informingscience.org/InSITE2005/I19f28Ali.pdf

Barrow, J., Gelderblom, J.H., & Miller, M.G. (2002). *Introducing Delphi programming: theory through practice* (3rd ed.). Cape Town: Oxford University Press.

Cohen, J. (1988). *Statistical power analysis for the behavioral sciences* (Revised ed.), Orlando, FL: Academic Press.

Department of Education. (2003a). *National curriculum statement Grades 10-12 (Schools): Guidelines for the development of learning programmes Information Technology*. Filename: ‘LPG Information Technology (15 Aug 2003)’ extracted from IT.ZIP. Retrieved August 19, 2003 from http://wced.wcape.gov.za/ncs_fet

Department of Education. (2003b). *National Curriculum Statement Grades 10 – 12 (General): Information Technology*. Printed for the Government Printer, Pretoria by Shumani Printers.

Garner, S. (2003). A tool to support the use of part-complete solutions in the learning of programming. *Proceedings of the 2003 Informing Science and Information Technology Education Joint Conference, Pori, Finland*, 343-353. Retrieved April 25, 2007 from http://proceedings.informingscience.org/IS2001Proceedings/pdf/GarnerEBKATool.pdf

Goosen, L. (2004). *Criteria and guidelines for the selection and implementation of a first programming language in high schools*. PhD thesis, North-West University (Potchefstroom Campus) available from http://www.puk.ac.za/biblioteek/proefskrifte/2004/goosen_l.pdf

Lambert, K.A. & Nance, D.W. (1997). *Understanding programming and problem solving with C++*. Minneapolis: West Publishing.

Manouchehri, A. & Goodman, T. (1998). Mathematics curriculum reform and teachers: Understanding the connections. *Journal of Educational Research, 92*, 27-41.

Mehic, N & Hasan, Y. (2001). Challenges in teaching Java technology. *Proceedings of the 2003 Informing Science Conference, Cracow, Poland*, 301-309. Retrieved April 25, 2007 from http://proceedings.informingscience.org/IS2001Proceedings/pdf/MehicEBKChall.pdf

Miah, S. (1997). Critique of the object oriented paradigm: beyond object-orientation. Retrieved January 27, 2003 from http://members.aol.com/shaz7862/critique.htm

Palumbo, D.B. (1990). Programming language/problem-solving research: a review of relevant issues. *Review of Educational Research, 60*, 65-89.

Palumbo, D.B. & Reed, W.M. (1991). The effect of BASIC programming language instruction on high school students' problem solving ability and computer anxiety. *Journal of Research on Computing in Education, 23*, 343-345.

SIGCSE: Special Interest Group in Computer Science Education. (2001). *Introductory courses. Chapter 7*. In Computing Curricula 2001: Computer science volume. Retrieved April 2, 2003 from http://www.acm.org/sigcse/cc2001/index.html

Steyn Jr., H.S. (2000). Practical significance of the difference in means. *Journal of Industrial Psychology, 26*(3):1-3.

Walker, H.M. (2000). *AP CS update*. Retrieved August 10, 2001 from http://www.math.grin.edu/~walker/sigcse-ap/update-6.16.00.html

White, G.L. & Sivitanides, M.P. (2002). A theory of the relationship between cognitive requirements of computer programming languages and programmers' cognitive characteristics. *Journal of Information Systems Education, 13*(1), 59-66.

Biographies

Leila Goosen, PhD, is a lecturer of Information Technology and General Mathematics in the Department of Science, Mathematics and Technology Education in the School for Teacher Training at the Groenkloof Campus of the University of Pretoria. Her research interests are co-operative work in IT, effective teaching and learning of programming and teacher professional development.

Prof. **Elsa Mentz** is an associate Professor in Computer Science Education in the Faculty of Education Sciences at the North-West University (Potchefstroom Campus). In her research she focuses on teaching and learning of computer programming skills. She holds a National Research Foundation grant. She has been actively involved in the training of IT teachers for the past twelve years.

Hercules Nieuwoudt PhD, is an associate Professor and Director of the School of Education in the Faculty of Education Sciences, at North-West University, Potchefstroom Campus. His research focuses on the teaching and learning of Mathematics in meaningful contexts, particularly geometry, Realistic Mathematics Teacher Education and the utilization of dynamic technology in teaching-learning situations. His educational focus is on efficient Mathematics education, research training and strategic teaching for meaningful learning.