Determining suitable programming language for the Bachelor of Technology (IT) curriculum

Johnson Dehinbo
jdehinbo@yahoo.com
Department of Web & Multimedia Applications, Tshwane University of Technology, Pretoria, 0001, South Africa

Abstract

Various programming languages are being taught in tertiary institutions in South Africa leading to repetition. Also, there is a low pass rate for some of these languages. This study aims at identifying a suitable programming language. Previous studies have generally compared programming languages without reference to specific use. Also, some comparisons did not use various criteria with various methodologies. Survey and programming experimentation were used in this study. Questionnaires were administered to respondents using any of the four languages: C++, Java, Visual Basic, and Pascal in different study groups. For the experimentation the line of codes (LOC) for solutions to a given problem using each of the programming languages were determined. It was found that Pascal is simple to write for beginners, but not suitable for complex tasks. VB is found easy to use under pressure and has the smallest line of code (LOC) making it also easier to learn. C++ and Pascal also have reasonable LOC (8). Java however, has the longest LOC, making it more difficult for beginners to learn. Like VB, Java is found suitable for complex jobs and is considered very flexible. The study recommends C++, Java and VB, using a systematic combination to achieve the desired result.

Keywords: Programming languages, teaching, curriculum, classroom

1. BACKGROUND OF THE STUDY

Introduction

The overall goal of this study is to determine a suitable programming language as part of the curriculum of the new Bachelor of Technology degree in Information Technology at tertiary institutions in South Africa. Adequate consideration need to be given to the choice of a programming language that will be easy to learn and use for first year students. This programming language should offer some prospects for future industrial use.

This study aims to throw light in the choice of a programming language. The purpose is not to persuade users that one programming language is better than another but to assist users in making a more informed choice.

The problem leading to the study

Information Technology (IT) graduates need adequate knowledge of programming. There are various programming languages such as BASIC, FORTRAN, Pascal, C, C++, Visual Basic (VB) and Java, which are being taught in tertiary institutions. Vinoski (2004) indicates that the many languages and vendors from which one can choose does not make choosing the one that will best solve one's problem an easy task. This is important in view of Sebesta's (1996, p.2) statement that the depth at which we can think is influenced by the expressive power of the language in which we can communicate our thoughts. Sebesta (1996, p.3) further illustrates that the language in which programmers develop software places limits on the kinds of control structures, data structures, abstractions and algorithms they can construct.

Kelleher and Pausch (2005) illustrate various systems that have been designed as a result of such barriers to programming. Similarly, Bishop (1999) mentions Trott (1997) stating that, "As time progresses, we see more and more new programming languages of all kinds appearing on the market". All aim to provide improved functionality. If the language used does not have the necessary features it will not be a good choice to use to teach programming concepts, and will reduce the quality of designed systems. It will also result in students taking longer to develop their programs, and time allocated for laboratory sessions is limited. Also, many programmers, when given a choice of languages for a new project, continue to use the language with which they are most familiar, even if it is poorly suited to the new project. It is therefore very important for student to be trained using a programming language that will be most suitable for their future tasks.

This could be why Lim (2002) states that information systems / computer science departments must reexamine their curricula in order to prepare students to face the challenge of being productive in a computing world that is now swamped with programming technologies. Up to now, institutions teach various programming and scripting languages. These led to repetition and confusion being reported by students. Some students resorted to absenteeism, stating that they had learnt the concepts now being taught in other programming languages. However, they usually ended up failing such subjects.

Coupled with the above is the poor performance of students in C++ as a subject in the former Technikon Northern Gauteng (now Tshwane University of Technology) in the last few years. Therefore, the second problem relates to the possibility of low levels of comprehension of programming by students. Especially for novice or beginning programmers, comprehension is very important (Lahtinen et al., 2005; Wiedenbeck, 1999). Low levels of comprehension could arise if it takes students a long time to comprehend the logic and syntax of programming language used. This is analogous to learning driving using manual car instead of automatic.

Unfortunately, choosing a suitable programming language is not easy. Prechelt (2000) indicates that when it comes to the advantages and disadvantages of various programming languages, programmers and computer scientists alike usually hold strong opinions. Moreover, Apte et al. (2003) note that a study of existing literature showed varying conclusions about the superiority of one language over another. However, Wiedenbeck et al. (1999) indicates that analyzing programming languages, paradigms, and development platforms is still very important for understanding how different programming styles affect the learning of novice programmers.

The research questions

In line with the above, the study seeks to address the following research questions:

- How could we choose a language that is easy to use and easy to learn for first year students?
- How could we choose a programming language that is suitable for future complex tasks?

Objectives of the study

In order to answer the research questions, the objectives of the study in clear, measurable and achievable/manageable terms are:

1) Investigate related studies that compare similar programming languages and dynamic Web platforms.
2) Investigate the ease of use, ease of learning and suitability for complex tasks of the programming languages.
3) Establish experimentation for Computing the Line of Codes (LOC) for the selected programming languages.

Importance and use of the study

The study would enable tertiary institutions to offer a programming language that is easy for beginners to learn and of relevance and importance to current developments in Information Technology. In addition, Students, professionals, software organizations, Web application developers, researchers, in the development of their applications might benefit.

2. LITERATURE REVIEW

Introduction

The overall purpose of this section, according to Marshall and Rossman (1989), cited in Creswell (2003), is to relate the study to the larger ongoing dialogue in the literature. This fills gaps and extends prior studies.

Previous comparisons without criteria

Various previous studies have compared programming languages. However, most of these studies did not use any criterion as a basis for the comparisons. Kruse (2003) illustrates the differences in the strengths and weaknesses of Personal Home Page (PHP) and Java. Klopper (2003) compares Personal Home Page (PHP), Active Server Page (ASP) and Java Server Page (JSP) in terms of their advantages and architectures. Comparison must be based on a variety of factors supported by scientific facts and results. This is in line with Ashenfelter's (1999, p.105) assertion that before analyzing tools, it is worth discussing how to evaluate them.

Bishop and Hurter (1999) examine some of Java's competitors, namely the Scripting languages: Tcl/Tk, Perl and Python, with the following results. Python incorporated the features of Modula-3 into its scripting syntax thus making it suitable for "programming in the large", unlike Tcl and Perl. Some of the distinguishing features of Python as reported by Bishop and Hurter (1999) include the fact that programs written in Python are typically much shorter than equivalent C or Java programs for several reasons. Also, the high-level dynamic data types allow you to express complex operations in a single statement. Moreover, statement grouping is done by indentation instead of begin-end brackets. Lastly, no variable or argument declarations are necessary.

The comparisons in these studies are based on intuition rather than scientific facts. Scientific evidence is required to support this assertion.

Other related studies emphasizing smaller codes

In an empirical comparison of seven programming languages, Prechelt (2000) observed that designing and writing programs in the scripting languages, namely Perl, Python, Rexx or Tcl, takes no more than half as much time as writing it in C, C++, or Java. Moreover, the resulting program is only half as long. He therefore concluded that the scripting languages offer reasonable alternatives to other full programming languages, and they may offer significant advantages with respect to programmer productivity for reasonably small programs.

Bishop and Hurter (1999) also confirmed that a Java version of a server program in Bishop (1998) is nearly four times as long as its Perl's version. This is assumed to make Java difficult to learn. The lesson learnt here is simply that platforms that enable programs with smaller Lines of Codes will be easier to learn, but this needs to be within the context of other relevant criteria based on scientific facts.

Moreover, most studies that compare platforms concentrate on performance criteria (Vinoski, 2003) although some measure performance using various factors. In the next section, we will examine the move towards frameworks for performance comparisons.

Towards frameworks for performance comparisons without

Vinoski (2003) realizes that various comparisons of programming languages concentrate on performance comparisons. Renaud, Bishop, Lo, van Zyl and Worrall (2003) reported on some studies stating that various metrics can be used to measure performance of algorithms in distributed systems. These includes response or waiting time, synch delay, number of messages exchanged, throughput, communication delay, node fairness, Central Processing Unit (CPU) cycle usage, and memory usage. They however used response or waiting time.

Cooper (2001) estimated the response time for Personal Home Page (PHP), Active Server Page (ASP), ColdFusion and Java Server Page (JSP), with ColdFussion having the best performance. In comparing the performance of Java Servlets, Java Server Pages (JSP), Active Server Pages (ASP) and PHP, Dehinbo (2005) also found PHP having the best performance. Marshak and Levy (2003, p.3) also evaluated platforms only in terms of user-perceived latency.

It should be pointed out however, that performance has somehow been overemphasized in various studies. This view is shared by researchers such as Vinoski (2003), who agrees that a suitable framework for comparison should involve other relevant fac-

tors. Vinoski (2003) explains that people check only those qualities that are easily measurable, such as performance. He goes on to say that an interesting side effect of this is that it has unintentionally led many programming language users to presume that "high performance" is the same as "high quality". Meanwhile, such works could be entirely meaningless, depending on the nature of your application.

The need for criteria in comparisons

Various comparisons of programming languages and Web-based platforms have been made in the past. Some of these studies propose the use of a combination of two or more factors for their comparisons.

Although Van Hoff (1997) evaluates only Java as a programming language, he evaluates it in terms of simplicity, familiarity, and object-oriented features with more emphasis on single inheritance. Hartman (2001) examines some tools for developing dynamic Web sites, namely ASP, PHP and ASP.NET. He mentions that one factor that complicates choosing a scripting environment is the issue of culture among developers. He believes this has a lot to do with the ideological camps to which they belong, such as Java versus PHP. He mentions very few developers are equally willing to work in both camps, or who can talk about "the other" technology without a trace of disdain.

According to Hartman (2001), the second factor that complicates choosing a scripting environment is the future scalability and functional requirements. Hartman's study arrived at these conclusions; ASP is a commercial technology while PHP is an open-source technology. ASP is somewhat easier to learn whereas PHP enhances flexibility. Moreover, ASP is limited to IIS/PWS on Windows while PHP runs on a multitude of servers and platforms. In terms of the comparison, it is important to note that Hartman (2001) uses terminology or factors such as perception of developers, performance and efficiency of platforms, ease of learning, and cost and scalability of the platforms. These factors are not, however, exhaustively investigated.

In a study comparing the performance of programming platforms for generating dynamic Web content, Cecchet et al. (2003) evaluate three dynamic Web platforms, namely PHP, Java Servlets and Enterprise Java Beans (EJB). It was found that EJB had higher latency values (worse performance) than that both PHP and Java Servlets, while Java Servlets had higher latency values than PHP which has lowest latency giving best performance out the three (Cecchet et al., 2003).

In terms of ease of development, Cecchet et al. (2003) explain that PHP scripts are easy to write because they can be seen as an extension of the HTML language that embeds code directly into an HTML page. However, they voice the concern that the database interfaces of PHP are ad hoc and code maintenance for database is awkward as new code must be written for each new database access.

Thus, Cecchet et al. (2003) evaluated platforms in terms of performance and ease of development. Their focus however is not aimed specifically at determining suitability based on specific use, such as teaching specified concepts. Moreover, they compared EJB along with other non-EJB architectures. Therefore, Cecchet et al. (2003) did not compare platforms with similar architectures, as EJB's architecture is more specialized than that of PHP and Java Servlets.

Summary

Choosing the appropriate tool should involve exhaustive evaluations of various options based on various relevant criteria that are backed by scientific facts and results. Some previous comparisons are based on intuition, while others use some criteria. In some cases, the recommended criteria have not been exhaustively tested. Other comparisons were not aimed at a specific need. This study fills this gap in the body of knowledge by being unique in the following ways:

This study does not agree that it is sufficient to list the advantages and strengths of each programming language. Rather, the advantages and strengths of each programming language should be examined and ranked in the light of certain desired qualities relevant to that specific use.

Most importantly, this study attempts to solve the research problems using survey and experimentation, thereby providing scientific evidence and exploiting the strengths of both methodologies.

3. RESEARCH DESIGN AND METHODS

This study uses a combination of two different approaches to obtain the data. According Dix et al. (1998, p.440), the best way to find out how a system meets users' requirements is to "ask the user". Therefore, survey will be used in this study. This will be supplemented with a programming experimentation which in line with McMillan and Schumacher (2001, p.32), will involve manipulating an algorithm by implementing it using the syntax of selected languages.

Programming Experimentation

For the experimental part, respondents were given a small problem to be solved using each of the programming languages; C++, Java, Visual Basic, and Pascal. The Line of Codes (LOC) for each solution in each language was measured.

Establishing validity and reliability of the experimentation: It is important that a measuring scale or instrument be consistent and reliable. It should produce more or less the same accurate results every time it is applied, even by different persons (Coertze & Heath, 1997, p.78). To increase the validity of this experimentation, all inactive command statements are removed from the programs.

Scope and assumption of the experimentation: The programming for the experimentation will be limited to simple programming. It is assumed that most respondents would be able to write program to answer such question.

Survey

The stratified survey approach was used. The questionnaires contain closed questions. This is necessary in order to solicit information including: ease of learning, ease of use under pressure, suitability for complex jobs, problems commonly encountered, programming language preferences, flexibility and efficiency.

Population: The population for the survey is all programmers and students using the four programming languages in their works and studies. Due to financial and time constraints, this was limited to respondents in software organizations within the economically vibrant Gauteng region as well as web lecturers, programmers, researchers in major higher institutions in the Gauteng region in South Africa.

At an estimated value of 3 programmers per organization, and 100 software organizations in the Gauteng region, we have about 300 subjects. At an estimated value of 20 lecturers per institution, and 25 higher institutions in the region, we have about 250 subjects in South African higher institutions. These give a total of about 800 subjects. The intended population is stratified into 8 groups consisting of the group of students and the group of experienced programmers currently using C++, Java, Visual Basic and Pascal. This is necessary in order to have an adequate representation of students and experienced users of the various programming languages.

Sampling method: As stated by Corbetta (2003, p.211), sampling should enable the results obtained by studying the sample to be extrapolated to the whole population. The population is divided into strata and samples taken by randomly administering questionnaires to respondents in major tertiary institutions and industries in the Gauteng region.

According to Corbetta (2003:216), for a 95% confidence level, a population size of about 800 requires a sample size of 120. Therefore a sample size of about 100 to 120 would suffice.

Data collection: A total of 160 questionnaires were distributed. In all, a total of 110 questionnaires were duly completed and returned.

Questionnaire's measuring scale: Dhyani et al.(2002) states that when one can measure what one is speaking about, and express it in numbers, one has demonstrated tangible knowledge about it. Therefore, using close-ended questions, the measuring tool will have values on a scale of 1 to 5.

Establishing reliability and validity of the survey: The research assistants were trained before starting to collect data. The questionnaire was piloted on computer lecturing staff, to ensure it is clear and unambiguous. Sampling groups were chosen to include both the practicing programmers and learners. The study was concluded within the time frame of 6 months as information obtained after such period may no longer be valid due to technological changes.

Scope of the survey: The sampling groups were chosen to include respondents from the Pretoria, Midrand, Sandton and Johannesburg areas. These areas form a significant part of the IT driving hub in South Africa.

Overall analysis to achieve the result

The average scores for the responses will be obtained for each of the questions in the survey, as well as the score assigned in the experimentation. The language with the highest total score is selected.

4. DISCUSSION OF THE RESULTS

The survey results are first presented. This is followed by the results for the experimentation to estimate the Line of Code (LOC) for a simple programming exercise written in the various languages. To obtain a clearer picture of the responses, the scaling system is grouped. The information is then presented in the form of bar charts as given below:

Ease of Learning

From Figure 1 below, we can see that a total of 95 respondents indicate VB to be either very easy, relatively easy or okay to learn as compared to 75 responses for each of C++ and Java. This is probably true because VB has the "drag and drop program generating" facility in which you simply drag an object such as a Combo box unto the desktop, and VB then translate to equivalent program codes. Pascal however, tops in the "Don't know + No response" bar. This could be due to the fact that it is outdated by now and so only few people know about it.

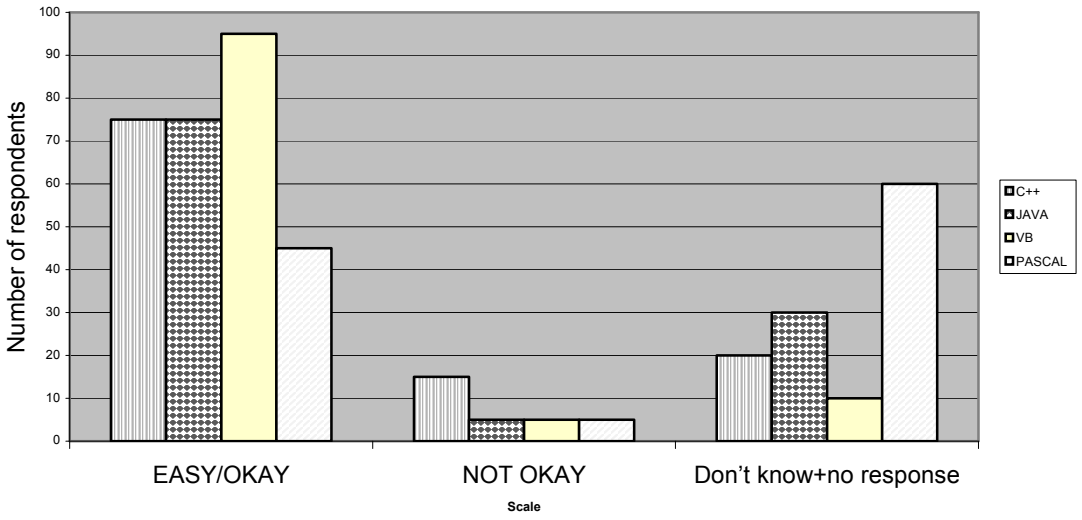

Figure 1. Ease of Learning chart

Ease of use under pressure

From Figure 2 below, more respondents (90) indicate VB to be easier to use under pressure, followed by Java and then C++. This is due to the "drag and drop facility" in VB as well as the windows development environment for both VB and Java which, in addition, also has Microsoft Disk Operating Systems (MSDOS) executing modes. Pascal however runs mostly from the MSDOS prompt.

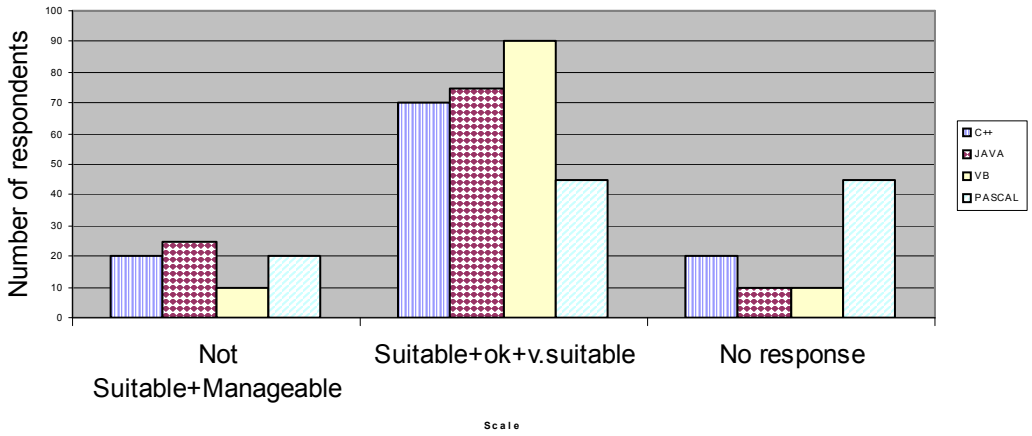

Figure 2. The ease of use under pressure

Suitability for complex jobs

From Figure 3 below, 90 respondents indicate VB to be best suitable for complex jobs, followed by Java and then C++. This is due to the fact that VB and Java can connect to other software tools like internet web pages and databases via javascript, servlets, and middlewares. Again, Pascal is considered less suitable for complex jobs, as those new tools were not yet available when Pascal came to being.

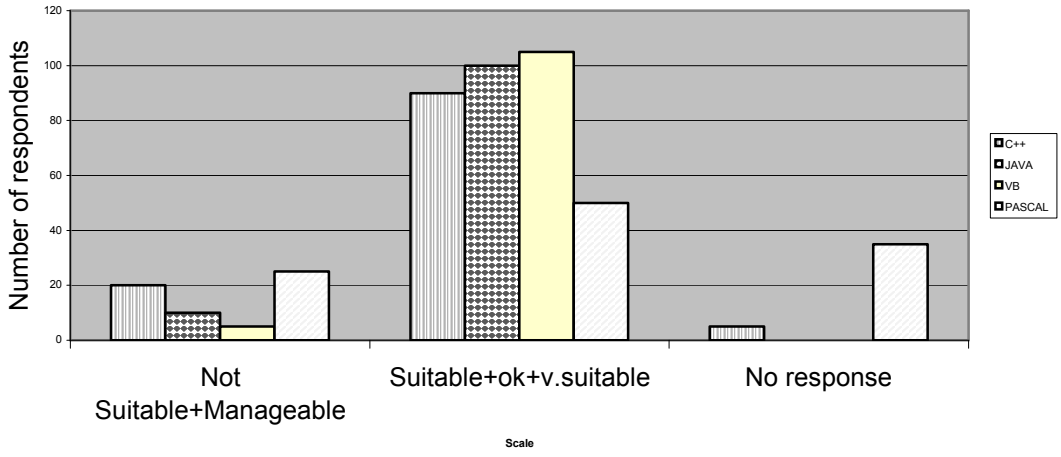

Figure 3. Suitability for complex jobs

Overall rating for the languages

From Figure 4 below, C++ and Visual Basic are considered to be equally best by 80 respondents. Java comes next with 70 responses, probably due to the fact that in comparison with the other programming languages, it is newly being introduced in most schools and in the industry. With respect to the overall rating above, here again Pascal is rated the lowest as a programming language.

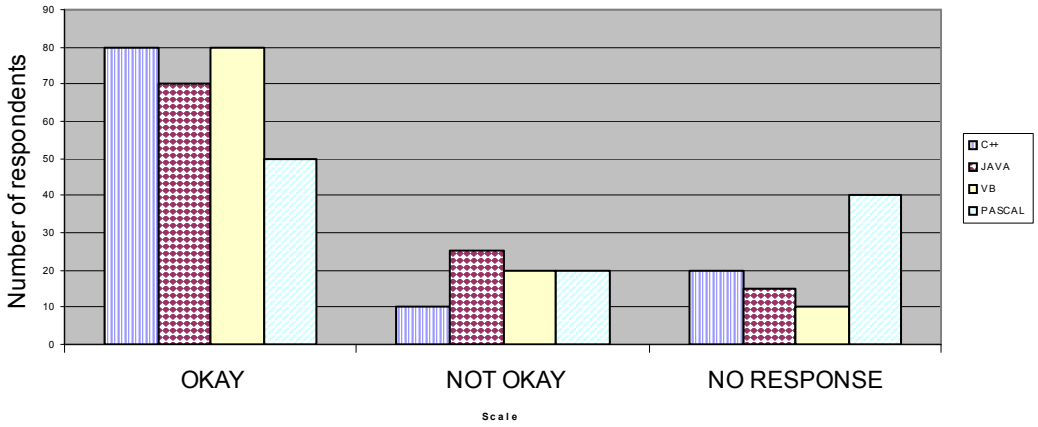

Figure 4. The overall rating for the programming languages

Line of Code (LOC) estimation

From the completed questionnaires, the line of code was estimated for each programming language's solution to the given problem. The minimum line of code for each solution is given below:

Java: LOC = 10

```
import java.lancs.*;
public class CalculateArea
{public static void main (String[]args)throws
Exception
{int length, breath;
BasicIo.prompt("PLEASE ENTER THE
LENGTH OF THE AREA");
length=BasicIo.readInteger();
BasicIo.prompt("PLEASE ENTER THE BREATH
OF THE AREA");
breath=BasicIo.readInteger();
System.out.println("THE AREA OF RECTAN-
GLE =" + length * breath +  "|");
}//end of method main
}//end of class CalculateArea
```

VB: LOC = 8

```
Private Sub cmdCalculate_Click()
Dim intLength As Integer, intbreadth As In-
teger
Dim lngArea As Long
intLength = Val(txtLength.Text)
intbreadth = Val(txtBreadth.Text)
lngArea = intLength * intbreadth
txtarea.Text = lngArea
End Sub
```

C++: LOC = 9

```
#include <iostream.h>
main ()
{       int length, breadth;
        cout << "Enter length of triangle";
        cin >> length;
        cout << "Enter breadth of triangle";
        cin >> breadth;
        cout << "The area of the triangle is
"<< length * breadth<<endl;
        return 0;
}
```

Pascal: LOC = 10

```
Program calcArea (input, output)
var
   length, breadth, area  : integer;
begin
  writeln ( 'Enter length of triangle')
  readln ( length );
 writeln ( 'Enter breadth of triangle')
 readln ( breadth );
 write ( 'The area of the triangle is  ',  length
* breadth);
end
```

VB has the smallest LOC. C++ and Pascal also have reasonable LOC (8) minus the "begin" and "end" statement, making them easy to learn. Java however, has the longest LOC, making it more difficult for beginners to learn. However, it is considered suitable for complex jobs.

5. CONCLUSIONS

It was found that Pascal is simple to write for beginners, but not suitable for complex tasks. VB is easy to use under work pressure possibly due to the "drag and drop program generating" facility. Additionally, VB has the smallest LOC. In addition to the "drag and drop facility", this makes it easier to learn and use under pressure. C++ and Pascal also have reasonable LOC (8) minus the "begin" and "end" statements, making them easy to learn. Java however has the longest LOC, making it more difficult for beginners to learn. Like VB, Java is suitable for complex jobs and is considered very flexible as it interfaces with other web technologies like Java Servlets, and other middleware platforms.

The above findings therefore show that no single language can adequately satisfy all the requirements. But a careful combination of the languages can achieve the desired result. The study therefore concluded that the low pass rate for C++ is not due to C++ being an exceptionally difficult language, as there is no significant difference in the factors studied for C++, Visual Basic and Java.

6. RECOMMENDATIONS

From the above findings and conclusion, C++, Java and VB are all recommendable. VB is recommended for its ease of use under pressure due to the "drag and drop program generating" facility. C++ is recommended because of the fact that for simple programs, C++ has few lines of code (LOC), which would make it easer to learn. Java is recommended for its flexibility and suitability for complex jobs.

This study recommends a systematic combination of the programming languages to achieve the desired result of enhanced comprehension and potential capacity for future complex challenges. Since C++ and Java have similar constructs (Hamilton, 1996), it will be alright for students to start with C++ in the first year and at the second year level, they should move to Java which can do all C++ can do and more with a "gentle" language constructs (Bergin, 1996). Visual Basic can be taught parallel to either C++ or Java in either first or second year respectively.

7. REFERENCES

Apte, V., Hansen, T. and Reeser, P. (2003) "Performance comparison of dynamic Web platforms." Computer Communications. 26 (8) pp. 888 – 898.

Ashenfelter, J. Paul (1999) Choosing a Database for Your Web Site. Wiley Computer Publishing, New York.

Bergin, J. (1996) "Object Technology in the Classroom: Java as a Better C++." ACM SIGPLAN Curricular Patterns. 1 (2) pp. 21-27.

Bishop, J. (1998) Java Gently: Programming principles explained. 2nd edition. Addison-Wesley Longman, London.

Bishop, J. and Hurter, R. (1999) "Competitors to Java: Scripting languages." Proceedings of the South African Computer Lecturers Association (SACLA) conference. Golden Gate, South Africa, 5-8 September 1999, pp. 88-95.

Cecchet, E., Chanda, A., Elnikety, S., Marguerite, J. and Zwaenepoel, W. (2003) "Performance Comparison of Middleware Architectures for Generating Dynamic Web Content." Lecture Notes in Computer Science: Middleware. 2672 (2003) pp. 242-261.

Coertze, D. and Heath, R. (1997) Research Methodology for Technikon Students: A practical Approach. Technikon Natal publishing, South Africa.

Cooper, R. (2001) "Software for managing Web sites." Proceedings of the South African Institute of Computer Scientists and Information Technologists (SAICSIT) Annual conference. September 2001. Pretoria, South Africa.

Corbetta, P. (2003) Social Research: Theory, Methods and Techniques. Sage Publications, London.

Creswell, J. Will (2003) Research Design: Qualitative, Quantitative and Mixed Methods Approaches. 2nd edition. Sage Publications, New York.

Dehinbo, O. Johnson (2005) "The performance of Web-based 2-tier middleware systems." Journal of Issues in Informing Science and Information Technology. 2 (2005) pp. 757-769. Web site. http://2005papers.iisit.org/I59f23dehin.pdf Retrieved 13 May, 2005.

Dix, A., Finlay, J., Abowd, G. and Beale, R. (1998) Human-Computer Interaction. 2nd edition. Prentice Hall, Hertfordshire.

Dhyani, D., Ng, W.K., and Bhowmick, S.S. (2002) "A Survey of Web Metrics." ACM Computing surveys. 34 (4) pp. 469-503.

Hamilton, M.A. (1996) "Java and the Shift to Net-Centric Computing." Computing Practices IEEE. 1 (2) pp. 31-39.

Hartman, H. (2001) "Tools for dynamic Web sites: ASP vs PHP vs ASP.NET." Seybold Report Analysing Publishing Technologies, 15339211 (12) pp. 1-12.

Kelleher, C., and Pausch, R. (2005) "Lowering the barriers to programming: A taxonomy pf programming environments and languages for novice programmers." ACM Computing surveys. 37 (2) pp. 83-137.

Klopper, S. (2003) "Comparing the three scripting languages: PHP, ASP and JSP with each other, in order to use the best option for a specific application." B.Tech Articles, Faculty of ICT, Technikon Pretoria. 1 (2003) pp. 17-22.

Kruse, W. (2003) "A comparing of PHP and J2EE." B.Tech Articles, Faculty of ICT, Technikon Pretoria. 1 (2003) pp. 110-117.

Lahtinen, E., Ala-Mutka, K. and Jarvinen, H. (2005) "A study of the difficulties of novice programmers." Proceedings of the 10th annual SIGCSE conference on innovation and technology in computer science education. Caparica, Portugal. pp. 14-18.

Lim, B.L. (2002) "Teaching Web development technologies: Past, present, and (near) future." Journal of Information Systems Education. 13 (2). pp. 117-123.

Marshak, M. and Levy, H. (2003), "Evaluating Web user perceived latency using server side measurements." Computer Communications. 26 (8) pp. 872-887.

McMillan, J.H. and Schumacher, S. (2001) Research in education. 5th edition. Addison Wesley Longman, Inc, New York.

Prechelt, L. (2000), "An Empirical Comparison of Seven Programming Languages." Computer. 33 (10) pp. 23-29.

Renaud, K., Lo, J., Bishop, J., Van Zyl, P and Worrall, B. (1999) "Algon: A framework for supporting comparison of distributed algorithm performance." Proceedings of PNDP conference. February 2003. Genoa, Italy.

Sebesta, R.W. (1996) Concepts of Programming Languages, 3rd edition. Addison-Wesley Publishing Company, New York.

Van Hoff, A. (1997), "The case for Java as programming language." IEEE Internet Computing. 17 (1) pp. 51-56.

Vinoski, S. (2003) "The performance presumption." IEEE Internet Computing. 7 (2) pp. 88-90.

Wiedenbeck, S., Ramalingam, V., Sarasamma, S. and Corritore, C.L. (1999) "A comparison of the comprehension of object-oriented and procedural programs by novice programmers." Interacting with Computers. 11 (3) pp. 52-282.