Why Tertiary Students Largely Fail Beginning Computer Programming Courses at the Polytechnic of Namibia: Different Perceptions, Common Recommendations.

Dr. Bernd Kiekebusch, *Polytechnic of Namibia;* Aina Tulimekondjo Nghipangelwa, *Telecom, Namibia*

Abstract

Results for beginning computer-programming courses at the Polytechnic of Namibia show high failure rates. This study explores perceived reasons by lecturers and students, in particular with regard to the impact of the chosen first programming language. Based on pilot studies under similar circumstances documented elsewhere, the authors recommend that the computer language be adapted to fit the students' capacities rather than the industry-desired final outcome. Based on the background of beginner students at the Polytechnic of Namibia it is suggested to use LOGO in the first semester and continue afterwards with industrial-strength languages.

Background

Students of computer programming at the Polytechnic of Namibia (PoN) are much more likely to fail than to pass their first course. While opinions about the causes for this problem vary, and several strategies have been employed over the past years, the high failure rates persist. Currently, there is no consensus among PoN staff as to what the root causes are and consequently what methods could be used to improve results.

The problem is possibly exacerbated, because the PoN is a new institution in a new country. Until Namibia's independence in 1990 the school system (of then South West Africa) was administered by South Africa, and university-level education was available only there and almost exclusively to the white community. In 1990 the official language was changed from Afrikaans to English, and secondary education adopted the Cambridge system with little preparation for the teachers. The University of Namibia was founded shortly after, and the PoN, Namibia's University of Science and Technology, began operating in 1995. Both institutions had to start anew and had to rely heavily on foreign lecturers that were willing to assist under short-term contracts – at the same time students were first in their families, burdened with all the problems of lack of role models, low language and science proficiency, and scarce financial resources.

The PoN has offered courses in Information Technology (IT) since its founding.

For the initial eight years, an introduction to programming with the "C" language was given during the first semester. Due to consistently low pass rates (50-60%) the curriculum was changed in 2003 to begin with an introduction to algorithms followed by programming with "C" in the second semester. Whereas programming now experienced high pass rates (above >90%), the majority of students failed algorithms on first attempt (pass-rate 35-45%), but they often completed most or all of the non-programming courses. (Kiekebusch, collected class records 2002-2005). In 2006 another curriculum change permitted students to attend programming without passing algorithms and it appears that success rates for programming dropped again.

Failing the first programming course has many negative consequences: the students feel discouraged (especially as this occurs in their first year) and some find that they are unable to continue their studies because their source of funding (a parent, relative, bursary or loan) won't pay for a repeat course. In the authors' experience, it is also frustrating for lecturers and adds to the teaching workload of the IT department.

Difficulties in learning a first computer programming language are a global phenomenon, however, as an Internet search about the topic easily demonstrates. The causes are attributed to a variety of factors depending on local conditions, the background preparation by the students, as well as the choice of the first computer language. This study tries to identify prevalent causes at the PoN, to explore perceptions for the failure rate by both lecturers and students, and to draw conclusions and recommendations from collected data, personal experience, and references to relevant international publications.

However, the students' preparedness or lack thereof must be acknowledged also. Almost all PoN students were educated in Namibia's government-run school system, which has expanded greatly since the country's independence in 1990. Admission criteria for IT include: 25 points on the IGCSE scale, "E" in English, "D" in Mathematics, and an adaptation of the Myers-Briggs aptitude test (or a mature-entry test for older applicants). Still, many come with plenty of difficulties in English writing (English is usually their second language, if not their third or fourth), and a background in Mathematics that is mostly equivalent to only the British O level, or grade 12 in the USA. But this is not atypical in the developing world, and would seemingly affect the students' performance in many other courses, not just beginning Computer Programming. Note, however, that the students' academic preparedness – their strengths and gaps – was not the focus of this study.

Demarcations

It is assumed that the PoN represents a large number of institutions of higher learning in developing countries that struggle to deliver quality education with limited resources to relatively under-prepared students. This exploratory study is restricted to its School of Information Technology (SIT), albeit with the expectation that the conclusions and recommendations could apply more broadly.

Furthermore, for the purpose of this study, materials from the literature were subjectively selected for two themes only: (a) teaching programming to novices, and (b) languages either taught at the PoN already (C++ and JAVA) or considered to be suitable for beginners (BASIC and LOGO). This does not reflect a value judgement on other languages, but rather a restriction to what the authors consider relevant for the PON.

Environment

The Curricula:

During 2008, about 600 students were enrolled for qualifications in Information Technology. There are three tracks: Business Computing, Computer Systems and Network Administration, and Software Development. The highest-level degrees are Master and Bachelor of Technology in IT, but certificates and diplomas are given on yearly levels for exiting students, currently under revision (Prospectus 2008 & 2009 of the PoN). All courses are offered in full-time and part-time (evening) mode, however after the 3rd year demand occurs almost only for part-time studies because most students are already employed elsewhere. The curricula for the first three years have fixed structures, with electives and choices made available only towards the end. An industry practical is obligatory usually during the 6th semester. Accreditation by SERTEC/HEQC (the Higher Education Quality Committee of South Africa) was obtained in 2003.

The Programming Language:

The curriculum for the first year currently requires an Introduction to Algorithm Design (and data structures) followed by a first programming course in "C". The purpose is to familiarise students with structured approaches to problem analysis and design of step-by-step solutions, to be followed by programming in the rigorously formalised "C" language. The choice of "C" stems from the PoN's mission to offer practical-oriented education to meet perceived industry requirements.

The Facilities:

Within the framework of the general PoN network, SIT controls 11 computer laboratories with capacities ranging between 10 and 27 seats, of which two have specialised equipment, and three others are dedicated to service courses in computer user skills for non-IT students. Other venues are borrowed from the general pool of lecture halls of the PoN. Equipment is mostly based around MS Windows XP, and a 4-year hardware replacement policy is in place, but not always adhered to. Labs are used for teaching and therefore unavailable for general practice at least 75% of the time, some not available at all. For security reasons, labs are closed during off-hours. A library with an "internet café" for students (less than 100 seats) is available also, but is shared by all 9000+ students of the PoN.

Because first year students are unlikely to have access to computers at home or at a job, their learning must occur entirely within the classroom setting, using whatever time they are able to get at one of the PoN's computers – and available practice hours in the labs are very limited. However, this practice is just what students need to become familiar with design and testing of logical structures that are required to create error-free computer programmes.

The Lecturers:

SIT has between 25 and 35 lecturing staff in 4 departments ranging from tutors to professors, complemented by occasional visiting guest lecturers and contract lecturers for service courses. Only a few of them are actually involved in teaching introductory algorithms and programming – all in the Department of Basic Computer Skills.

The large number of repeating students increases class sizes and adversely affects individual attention to struggling students. Valuable teaching experience often cannot be retained, because the number of full-time staff fluctuates depending on contract lengths and work permits for the complement of foreign lecturers. And while it is the policy that trained staff covers all introductory courses, the department has difficulties in ensuring that there are sufficient lecturers available for all the scheduled courses.

An increase in pass-rates would significantly relieve some of these pressures currently experienced.

The Students:

Success rates for a three-year diploma are low: Less than 5% of students graduated within the time stipulated by the curriculum during 2002-2003, and as of 2008 it

takes on average about 5.2 years of study to obtain a 3-year qualification, where the greater part of time (3.5 years) is spent in completing 1st-year subjects. ("Measuring Student Success", internal report by Management Information Office PoN, October 2008). It is the authors' observation that students who pass the first year's courses are strongly motivated and graduate more quickly than those who don't. Repeated attempts to pass the introduction to algorithms and/or programming seem to be a significant obstacle – some students trying 3 to 5 times. This suggests that the students' initial experiences within the PoN School of Information Technology determines much of their later experiences within the SIT, as well.

What are the reasons for such a high initial failure rate? What can be done to improve not only the students' chances to finish their studies, but also to alleviate the tremendous burden placed on the PoN resources (and the students' own) by repeating courses?

Research Design

The investigations were pursued in three phases:

(1) An extensive literature review was based on publicly available web sites to explore which types of problems and recommendations are already documented, and which might be applicable to the situation at the PoN. With approximately 1.5 million references to the topic in a typical Google search, obviously only a limited selection could be included, mostly from reputed institutions of education. The results were used to prepare for phases 2 and 3.

(2) Based on prepared questionnaires, responses and comments were solicited from lecturers and students during the second semester of 2008.

All lecturers involved during the recent three years in beginning programming courses were approached to collect qualitative data about the environment in 2008; only five lecturers responded and were interviewed face-to-face following an outline of 13 pre-set questions. The information gained was combined with the authors' own observations, informal discussions with other lecturers and students in class, and collected data from previous years.

Students received 23 questions – however, participation was limited as only 11 older students responded. Others were approached, but either did not agree to face-to-face interviews or omitted answers to questions they felt intrusive into their privacy (as verbally expressed to the co-author). Nevertheless, received responses still seem valid and worthwhile to consider in an exploratory study, even though for a more quantitative approach the content and format of the questions would have

to be modified. Research findings were supplemented with results from informal conversations by the authors (especially Dr. Kiekebusch with his own students over the past nine years).

(3) An attempt was made to compare programming languages for suitability in teaching first-time students. Included were only the main languages taught at the PoN, "C / C++" and JAVA, and two languages frequently recommended for beginners: BASIC and LOGO.

Results from the Interviews

Student Preparedness

From the interviews it becomes clear that both students and lecturers feel strongly that the majority of beginners are unprepared for studies in IT.

Many chose IT without knowing what it entails. This information emerged during the face-to-face interviews and in conversations with the authors. Informally, students offer several reasons for their chosen course of study: some say they chose IT because it has the reputation of financially rewarding employment, others say because persons of authority decided for the student (parents, teachers, tribal chiefs, etc.). Still others say that they thought it would be "interesting" and were encouraged because their qualifying marks from secondary school were "good enough". Within the sample of interviewed students, about 45% did have some prior knowledge of IT. But disproportionately, these respondents were advanced students who had already succeeded to some extent. They indicated during the interviews that they felt their less successful classmates had far less or no prior knowledge.

Subject selection is a problem not only among IT students, it seems. An informal survey by the PoN English department in 2004 revealed about 80% of students across all qualifications at the PoN expressed that they were studying a subject they did not choose or did not want. (Undocumented verbal communication during a staff training session in 2005.)

The Namibian secondary education does not emphasise crucial skills that are needed by IT students. In particular, the students and lecturers interviewed felt strongly that critical or abstract thinking, problem solving skills and the rigorous testing of logical structures were neglected during previous education. These are also not skills that they developed at home, or in the community. Various reasons were given: cultural traditions that prohibit children from being inquisitive or creative, and children are taught not to question authority (including the teacher, if they do not

understand something). Primary and secondary school learners are often taught by rote and some teachers themselves exhibit low proficiency levels in critical subjects such as English, mathematics and science. Some students attributed their problems to a culture of obedience in their childhood (hence, studying something they would not have chosen themselves) and to a lack of support and understanding by family members for the time and effort needed to advance one's education.

First-year issues

Beginning students attend an introduction to algorithms followed by the first programming course in "C". These courses are closely linked and form the foundation for further studies.

All students surveyed identified significant difficulties during this time:

- Inability to describe simple processes in detail (as required) step-by-step
- Inability to de-compose problems into smaller sub-problems
- Inability to write with the required precision
- Inability to construct mental process models
- Difficulties to apply rigorous formal syntax
- Difficulties with the English terminology and even the pronunciation used by lecturers
- Inability to understand "what lecturers want", because "they do not provide step-by-step instructions" for repetition
- The feeling of "being lost" and/or "being bored"

Seemingly stemming from these fundamental obstacles, students also identified problems related specifically to "C" programming issues: multi-dimensional arrays, structures, pointers, memory allocation, functions, recursion, sort and search, and object-oriented concepts like inheritance.

Lecturers in the majority concur with these findings. However, their interpretation of student behaviour is quite different from that of the students.

Problem Lecturers' perception about students Students' own perception content not understood not trying hard enough; lack of class attendance; lack of interest unable to understand despite best effort no learning above and beyond class materials lack of interest, minimal effort, work for passing grades only not needed if not required, not enough time

study skills not developed, no note taking, no independent effort not learned in secondary school, too high expectations practical and/or homework not done laziness overcrowded labs, theory not understood, discouraged by lack of success students read very little, even prescribed textslaziness, missing reading culture, lack of interest books unaffordable, library has insufficient stock, expectation that lecturers provide condensed hand-outs impact of course presentation students are unprepared, have low language skills, are not motivated low teaching skills, unclear explanations, little support outside class, foreign accents teaching style lecturers teach like anywhere else, but very practical-oriented too high or foreign standards, too much theory use of own resources should have own computers (instead of other items like mobile phones) computers and network connectivity too expensive, risk of theft, no electricity at home

In summary, lecturers tend to see the causes for lack of success in students as:

- Low level of study skills
- Too little interest and motivation

By contrast, students emphasise:

- Difficulties to understand (content, language and presentation style)
- Lack of problem solving skills
- Too few available resources
- Too high expectations

Other issues also emerged, on which both lecturers and students agree:

1. The equipment and the available software environment for programming are appropriate. But there is not enough access to the equipment.
2. The level of English proficiency of many students is too low to follow lectures during the first year. The inability of many students to clearly write English hinders proper note-taking and makes it difficult for them to use these notes as a preparation for examinations.
3. Concepts of abstract thinking, individual project work, and team collaboration are under-developed from the students' previous education.
4. Programming itself is core to IT education. (Dissent has been heard previously from some lecturers and students, but it did not show during this research).

5. Programming should be taught in languages that are actually used in industry – however, that is the goal for completion of studies, not necessarily for struggling beginners.

6. Individual support from staff to students outside the classroom is minimal and reaches only students who take the initiative to ask.

7. Facilities are over-crowded and not enough computer lab time and space is available for individual practice. This coincides with the request to make labs available during nights and weekends with proper security and supervising tutors. (Ideally students should own their own computers, but currently that is not considered feasible.)

8. While there is agreement that the current concentration on structured thinking during the introduction to algorithms is important, many feel that the method of delivery could be different (for example to include programming examples, which is currently not possible due to unavailability of computer labs).

9. The teaching of programming should foster self-confidence and create enjoyment in the students, and thereby provide motivation for more effort to succeed.

Discussion and Concurrence from Literature outside Namibia

The search for relevant material about teaching programming to beginners was restricted to publicly accessible web sites on the Internet. No references were found that deal specifically with the situation in Namibia.

Although computer languages are designed to relieve the programmer from the intricate and tedious coding in binary object code or using the mnemonics of assembler code, many of the languages are geared more to the technical purposes, for which they were created, than for the novice programmer to learn concepts and understanding of principles. Therefore it is obviously important that the choice of computer language should be adjusted to the learners' capabilities and at the same time support the building of mental models and understanding of the processes involved.

"A language that doesn't affect the way you think about programming, is not worth knowing." (Perlis, 7-13)

The PoN's first programming language is C++, followed in later classes by JAVA. Lecturers often like to assume that the students are able to cope; they passed secondary school after all; they must be clever enough. However, C++ and JAVA for

teaching purposes may not be optimal. For example, in their analysis of suitability of programming languages for learning, Papp-Varga, Szlàvi, and Zsakó conclude: "... the statement-oriented method is unsuitable for teaching programming languages ..." (Papp-Varga 170). C++ and JAVA as well as BASIC fall into this category. However, a first programming language should be selected for pedagogical, not for industrial usefulness.

Corroborating the problems encountered at the PoN, teachers often say that the features that make these languages successful for industrial use are actually overwhelming to the novice learner. The concerns mentioned are as follows: (a) The dense, scientifically oriented code structure bears little resemblance to natural languages like English; (b) the text-based format with its rigorous spelling requirement is difficult to master, especially for students whose use of spelling in their spoken language is inconsistent (e.g. for speakers of a second or third language); (c) the invisibility of step-by-step intermediate results, (d) the use of pre-processors, multiple-dimensional arrays, declaration of variables, and block structure, and (e) in case of C++, the requisite memory management, pointers, and multiple inheritance factors. These echo the common concerns by beginning PoN students, who find it difficult to formulate precise statements, in particular whole sequences of instructions.

Lecturers at the PoN claim that most beginning students have only a basic concept of mathematical logic and operations, neither as progressions (lines of numbers) nor as geometric objects (matrices, number plane), but rather as something almost magical. Consequently it is difficult for the students to imagine theoretically in their mind only, how an algorithm would perform step by step, especially when indexing, memory allocation, pointers and other such items are needed that are not directly accessible to observation.

Problems of this type are not unique to Namibia. A literature search on these issues revealed that similar difficulties are discussed in the learning theories of Jean Piaget and the more computer-oriented educational work of Seymour Papert. While their theories and recommendations are often formulated in terms of childhood education, as Ruth Sower (1-6) showed, they apply equally to underprivileged students and those from developing countries.

Namibian tertiary-level students are by no means children anymore, but their abstract reasoning, critical thinking, and mathematical skills need further development. Just as offering courses to upgrade the students' abilities in the English Language and basic Mathematics is necessary, it should be therefore also a priority for the PoN, in particular the School of Information Technology, to enhance the basic skills and

abilities of reasoning.

The need has been recognised at the PoN, as evidenced by the inclusion of "thinking" training, according to Edward de Bono into the classroom. No connection has yet been made, however, to the possible impact that a first programming language could have on this process. Nevertheless, it appears imperative that the first programming language be chosen in such a way that it develops and shapes the thinking processes of the students towards acquiring the capacity to deal with complexity, rather than confront complexity outright (and thus, too often fail in the process).

Most Information Technology lecturers at the PoN consider BASIC as the preferable choice for an "easier-to-learn" first language. It certainly has advantages: It has an easy-to-learn structure, it is widely available and its newer dialects are integrated into Microsoft's products. As an interpreted language it offers insight into step-by-step results. Nevertheless, it still requires a statement-oriented method. An even bigger drawback is that, while it makes use of the language easier, it does not help improve analytical and reasoning skills. A BASIC programmer, who cannot logically structure a programme, will do no better in C++ or JAVA later in his or her career-studies.

Taking this into consideration, and again focussing on the work of Seymour Papert, the LOGO language poses a possible alternative (LOGO Foundation). LOGO was created primarily to provide an environment for children for the development of reasoning and communication skills, using computers for active learning in a playful way. In its basic form, it uses "turtle graphics" to let the learner explore visually (and by auditory means) the effects of various commands in real time, thus increasingly building up concepts of programming and programme structures. Studies in various populations from early childhood in under-privileged school districts, developing countries and culturally distinct groups (Gillespie; Sower) have demonstrated rapid success when using LOGO as a complementary educational tool.

Conclusion and Recommendations:

Within the context of the PoN, the research and analysis boldly suggest that, given the difficulties experienced by most first year students, the teaching of an easier-to-learn logical computer programming language combined with sufficient support, should improve results, namely acquiring basic reasoning skills and with it stronger motivation due to more successes. Moreover, the simplest and probably most effective way to achieve this improvement would be to offer more group support sessions during the first year (e.g. on study skills, question-and-answer,

and promotion of team-work among students), and to re-design the introduction to algorithms course in such a way that the advantages of LOGO could be fully utilised, making it a truly preparatory course on critical logic and reasoning. More emphasis on mathematical skills, logical thinking and abstract thinking can – and should – be integrated into the first year's teaching.

This conclusion leads to a three-part recommended strategy:

(1) Teach the incoming students at a level that fits their educational knowledge, use an introductory language like LOGO, and provide them with more study skills and related supports.

(2) Use the first year to achieve a solid foundation in reasoning and basic programming skills. Switch to industry-strength languages in later semesters.

(3) Designate a course as preparatory course for programming and provide the necessary resources that permit students to practise basic reasoning and programming skills extensively and with tutorial support.

These recommendations can be first steps to achieve larger numbers of more successful students in Information Technology, which is a goal at the PoN, needed by the country of Namibia and its industry, and can give the students themselves a fairer chance to graduate.

Dr. Kiekebusch has taught at the School of Information Technology of the Polytechnic of Namibia since 2000. (bkiekebusch@polytechnic.edu.na) . Ms. Nghipangelwa conducted the research as part of her Bachelor of Technology thesis at the Polytechnic during 2008. She works as systems analyst at Telecom Namibia (nghipangelwaa@telecom.na)

References

"BASIC" Computer Hope.com. 12 September 2008. <http://www.computerhope.com/jargon/b/basic.htm>

Bellis, Mary. "History of BASIC – (Beginner's All Purpose Symbolic Instruction Code)". 3 December 2008. <http://inventors.about.com/library/inventors/blbasic.htm>

Bennedsen, Jens. "Becoming a programmer requires more than to master a programming language", 27August 2008. <http://www.spop.dk/papers-ws1/JensBennedsen.pdf>

Danielsson, Lorenzo E. "Understanding algorithms for novice programmers." 22 September 2008. <http://lorenzod8n.wordpress.com/2007/05/20/understanding-

algorithms-for-novice-programmers/>

Degelman, Douglas, Ellen J Brokaw and John U Free. "Effects of LOGO Experience and Grade on Concept Learning and Creativity." Journal of Educational Computing Research. 2, 1986: 199-205. 4 April 2009.

<http://www.vanguard.edu/uploadedfiles/faculty/ddegelman/logo.pdf>

"Edward de Bono" Wikipedia. 5 April 2009. <http://en.wikipedia.org/wiki/Edward_de_Bono>

Garner, Stuart. "Learning Resources and Tools to Aid Novices Learn Programming." 16 Sep 2008.

<http://proceedings.informingscience.org/IS2003Proceedings/doc/036Garne.pdf>

Gillespie, Catherine Wilson and Sally Beisse. "Developmentally Appropriate LOGO Computer Programming with Young Children." Information Technology in Childhood Education Annual. Association for the Advancement of Computing in Education (AACE). 1 January 2001.

4 April 2009. <http://www.accessmylibrary.com/coms2/summary_0286-10350764_ITM>

Gorenburg, Michael. "CIS 22 – Data Structures: C++ vs. JAVA." 16 September 2008. <http://acc6.its.brooklyn.cuny.edu/~cis22/java/jvcpp.html>

Gupta, Diwaker. "What is a Good First Programming language?"

16 September 2008. <http://www.acm.org/crossroads/xrds10-4/firstlang.html>

Huang, Guowei. "The JAVA Language." 1996. 3 November 2008.

<http://ei.cs.vt.edu/book/chap21/javalag.html>

Kiryakoza, Steven. "The LOGO Programming Language." 12 Dec 1997.

12 October 2008 <http://www.engin.umd.umich.edu/CIS/course.des/cis400/logo/logo.html>

Klotz, L, P Sobalvarro and S Hain. "The Terrapin Logo Language." 1981.

4 April 2009. <http://www.sydlexia.com/logo.htm>

LOGO Foundation. "Welcome to the Logo Foundation web site."

4 April 2009. <http://el.media.mit.edu/Logo-foundation/>

"Logo Programming Language." 3 December 2008. <http://www.engin.umd.umich.edu/CIS/course.des/cis400/logo/logo.html>

Matthíasdóttir, Ásrún. "How to teach programming languages to novice students." International Conference on Computer Systems and Technologies - CompSysTech'06. 12 October 2008. <http://ecet.ecs.ru.acad.bg/cst06/Docs/cp/sIV/IV.13.pdf>

Medve, Anna. "Logo Elements of Methodology to Develop Structured Programming Terminology." University of Veszprém, Hungary, 1996.

4 April 2009. <http://eurologo.web.elte.hu/lectures/medve.htm>

"Myers-Briggs Type Indicator" Wikipedia. 4 April 2009. <http://en.wikipedia.org/wiki/Myers-Briggs_Type_Indicator>

Palumbo, David B. "Programming Language / Problem-Solving Research: A Review of Relevant Issues." Review of Educational Research, 60.1, 1990: 65-89. December 2008. <http://rer.sagepub.com/cgi/content/abstract/60/1/65>

Papert, Seymour. Mindstorms: Children, computers, and powerful ideas. New York: Basic Books, 1980.

Papert, Seymour. "Teaching children thinking." Journal of Structural Learning, 4, 1975: 219-229

Papp-Varga, Zsuzsanna, Péter Szlàvi and Lásló Zsakó. "ICT teaching methods – Programming languages.". Annales Mathematicae et Informaticae. 35, 2008: 163-172, Eötvös Loránd University, Budapest, Hungary. 4 April 2009.

<http://www.ektf.hu/tanszek/matematika/ami/2008/ami2008-papp-szlavi-zsako.pdf>

Perlis J. "Epigrams on Programming." SIGPLAN Notices 17.9, September 1982: 7-13.

Piaget, Jean. The child's conception of the world. Patterson, N.J.: Littlefield, Adams, 1960.

Piaget, Jean. The origins of intelligence in children. New York: International University Press, 1952.

"Programming language" Wikipedia. 12 October 2008. <http://en.wikipedia.org/wiki/Programming_language>

Polytechnic of Nambia. Prospectus for Undergraduate Studies.Windhoek: Polytechnic. 2008.

Polytechnic of Nambia. Prospectus for Undergraduates Studies. Windhoek: Polytechnic of Namibia. 2009

Undergraduate: <http://www.polytechnic.edu.na/academics/docs/prosp.pdf>

Postgraduate: <http://www.polytechnic.edu.na/academics/docs/pprosp.pdf>

Sower, Ruth. "Towards Achieving an Interactive Education Model for Special Needs Students: The Computer Writing Project from Native American Students." Journal of American Indian Education. 29.3, 1990.

4 April 2009. <http://jaie.asu.edu/v27/V27S1tow.htm>

APPENDIX

Interview Questions

Lecturers

1. Why do think students struggle to learn programming languages for the first time? Please provide some example.
2. Why is programming language difficult to master, especially to the first year students?
3. Does the Polytechnic provide a proper supporting environment?
4. Do you think that the Polytechnic has effective tools to teach programming to the beginners?
5. What suggestions and recommendations would you make to improve the programming environment?
6. Do you have knowledge of other programming languages?

   If yes

a. Which languages would you recommend for the beginners? Please provide some reasons for your choice.

b. Which programming features would you avoid for the beginners?

7. Do you have any formal training about methodology for teaching "programming" or do you rely on practical/personal experience?
8. What do you think about the methods of teaching programming at the Polytechnic?

a. How can it be improved?

9. Do you think the students get support in addition to formal lessons?(tutors, lecturers, team work, self-direct, study online, other)
10. Do you see specific difficulties in teaching programming language?
11. In your class, what percent of your students are motivated to learn programming?

12. How do you encourage interaction in class?

13. Do you have other comments on the topic?

Students

1. Did you ever hear about "programming" in high school?

2. Did you have any idea what "programming" was when you started to study IT at the Polytechnic of Namibia?

3. Do you think programming is essential for IT studies?

   If not: Do you think IT degrees should be awarded without required programming?

   If yes: Why? What would justify it?

4. Did you face any difficulties during your programming classes?

5. If you could be specific, what difficulties did you encounter during your programming classes?

6. What was the most difficult or challenging tasks you faced in programming?

7. Any specific functions or topics you had difficulties to master?

List the three most difficult concepts for programming (in your opinion).

8. Do you think the Polytechnic of Namibia gives the best introductory programming course? (Please explain why)

9. Are there any introductory programming languages that you think the Polytechnic of Namibia should offer?

10. How many programming languages do you know? (Could you please list them.)

11. How many do you feel comfortable to programme? (Could you please list them.)

12. How difficult do you think it would be to pick up another programming language? (if you can just explain why)

13. Do you like programming?

- If not: what specifically do you dislike?
- If yes: what specifically do you like?

14. How often do you practise?

15. Why did you take that period of time for practice?

16. Do you depend on the Polytechnic facilities for practice?(Please explain why you depend or not depend on the Polytechnic of Namibia's facilities.)

17. Do you get enough time to practise at the Polytechnic of Namibia?(Please explain why you don't get / get enough time to practise.)

18. Do you suggest that the Polytechnic should restructure their programming languages?(Please explain why you think so.)

19. Do you think that the Polytechnic should introduce an introductory programming language?(Please explain why.)

20. Do you suggest any programming language that the Polytechnic should offer to the first year students?

21. In your opinion, why do you think programming is hard for the first year students?

22. Any type of tools and techniques you think the Polytechnic should consider when offering programming?

23. In real life, a programming task is only successful, if the programme runs successfully.

However, the Polytechnic marks on a percent basis, and it is possible that a student passes a course without ever completing a program. Do you think that is justified?

If yes, why?

If no, what should the Polytechnic do?