Minding gaps within 'the bubble': challenges of unusually many under-prepared electrical engineering students

S. Winberg[†]
† Dept. Electrical Engineering, University of Cape Town, 7701 Rondebosch, South Africa, email: simon.winberg@uct.ac.za

The first year electrical engineering course had a significant increase in student number in 2009, compared to previous years. Although the 2009 students had good NSC grades, their performance showed that many of them were under-prepared for university study. This unusually high intake of weaker than normal students has been dubbed the 'bubble'. This paper identifies challenges posed by the 'bubble' and discusses ways they were addressed, focusing on changes to the course structure and the students' learning experiences. The methodology involved acquiring data from course material and assignment results. The findings indicate that students in 2009 experienced different difficulties to those in 2008, in particular difficulties at a fundamental level related to interpretation and written technical explanations. Students in 2008 and 2009 both showed competence in understanding program language syntax and semantics, and effective use of office software and online tools.

1. Introduction

The department of Electrical Engineering at the University of Cape Town experienced an unexpectedly high intake of 260 first year students in 2009, the largest intake in over ten years (Leather 2009). The planned average intake was 180 for 2009. A number of lessons were learned from the unexpectedly high 2009 numbers. This paper involves a case study of a first year engineering course that experienced an 88% increase in students in 2009 (a total of 167 students) at the start of the academic year, compared to 89 students at the start of 2008 year. This study focuses on the 'EEE1003W Computing for Electrical Engineers' course, which is a core course for two of the three electrical engineering programmes (EBE 2010).

The intake of first year students to the Electrical Engineering department had remained fairly predictable from the year 2008 and previously. Figure 1 provides a bar chart showing first-year enrolment numbers from 2006 to 2009. As the figure shows, there was a 49% jump in intake numbers between 2008 and 2009.

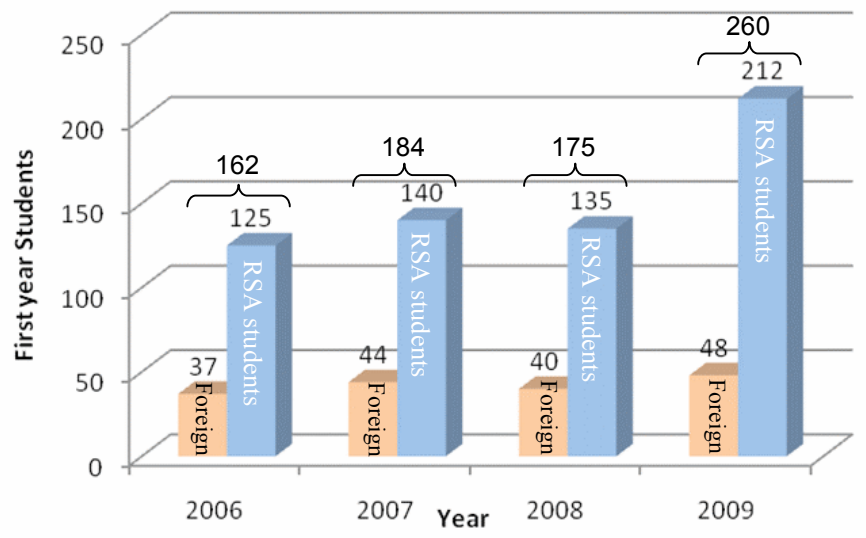

Figure 1: Electrical engineering first year intake for 2006 to 2009 spit between South African and foreign students

While the 2009 first years did achieved the necessary National Senior Certificate (NSC) scores to gain entry into the programme, it was observed in the context of the EEE1003W course that an unusually high percentage of these students were under-prepared in comparison to previous years. This paper focuses on the challenge of teaching an unexpectedly large cohort of under-prepared students. The Electrical Engineering department colloquially dubbed this predicament (i.e., the large intake of many under-prepared students) the 'bubble' in our system (Electrical Engineering department staff meeting, 26 Nov 2009). This 'bubble' is expected to create unusual complications though the systems of the department. For example, right from the start, the size of the bubble caused greater demands on lecturers, teaching support, lecture venues, lab space, and timetabling, among other problems. In retrospect, the logistic issues involved in addressing the larger group were fairly minor compared to the complex issues related to the wide range of students' abilities, in particular their differing levels

of mathematics and academic English proficiency, and their understanding of scientific approaches and problem-solving methods.

This paper is structured as follows: Section 2 presents a short background to the course investigated; Section 3 continues by describing the methodology used to record and analyse the preparedness of the students and to track changes to the course and the students' progress; Section 4 presents results; and Section 5 concludes the paper by reflecting on the results and discussing recommendations and likely future challenges.

2. Background

The Computing for Electrical Engineers course (course code EEE1003W) is a 16 credit HEQF level 5 (first year) course that focuses on teaching programming and the use of spreadsheet software to solve basic electrical engineering problems. Students learn how to use word processors, spreadsheets, do internet searches, and use web-based resources for documentation and problem-solving purposes.

The course draws on areas of knowledge and skills that the students would normally be expected to learn both in their high school education and from other first-year university courses. High school subjects that the course builds on include algebra from Mathematics, problem-solving and experimentation techniques learned in Physical Science, and comprehension and writing skills learned in English. Most of the students are not expected to have done Computer Studies at high school. The EEE1003W course also draws on skills that the students learn in other courses, in particular the use of vectors in Physics 1, complex numbers in Mathematics 1 and basic circuit design and analysis techniques which they learn in Electrical Engineering 1. The course is structured in such a way that coursework and assignments that draw on skills learned in other university courses are only presented after these skills have been taught in the relevant co-requisite courses.

The EEE1003W course is divided into two semesters. In the first semester, students learn Python (Python.org 2010) as a 'gentle' lead-in to programming (Davison 1995), and make use of office software such as Microsoft Excel or OpenOffice Calc (OpenOffice.org 2010) to confirm results and operations. They use Microsoft Word to write reports. In the second semester, students switch to ANSI C programming (Harbison and Steele 1994). During the second semester, the students can use a combination of Python and Microsoft Excel to confirm results and operation of their C programs; they are also expected to use some of the more advanced word processor features (e.g., iamges, URLs and tables) to write more complex reports for problem-solving tasks.

The design of EEE1003W was structured around both industry needs and needs of courses the students do in later years. Industry needs are mostly related to effective use of office software and C programming; these requirements were based on meetings with industry representatives and recommendations from the literature including Seltzer and Bentley (1999), Nagurney (2001) and Darzentas and Nicolle et al. (2004).

Changes made to the course structure and students' learning experiences in 2009 involved: duel teaching (running two classes at the same time in different venues due to the large class size), extra classes, tiered tutorials, a larger number of less-complicated laboratory tasks, more testing, and breaking projects into smaller, better detailed and illustrated parts. Students' learning experiences were widened, providing support via a larger variety of online collaboration tools together with more opportunities for face-to-face interaction. Teamwork was revised so that certain tasks encouraged weaker students to learn from stronger students, while other assignments avoided stronger student doing the work for weaker students.

3. Methodology

The methodology for this project involves three sequential phases: 1) gap modelling, 2) a comparison of gap models, and 3) skills mapping to visually track the progression of students in the course. The gap modelling phase involved acquiring data from the course to construct a model that represented common learning challenges or learning 'gaps', and their relation to how the course was restructured, as well as how learning experiences were adapted to accommodate these difficulties; this modelling process is explained in Section 3.1. A gap model was developed for the 2008 version of the course, which also resembles how the course was offered in previous years. A second model was constructed for the 2009 version of the course to reflect influences of the bubble. Sections 4.1 and 4.2 present the resultant gaps models. The comparison of gap models shows how the course changed (i.e., how the gaps and areas of students' strengths changed); this process for this comparison is described in Section 3.2. Skills matrix mapping complements the gap models by showing the progression of students through the course. For this analysis, a baseline for each student entering the course was first established by plotting the student's engineering preparedness using their results for the National Benchmark Tests (NBTs). This was followed by analysing the students' progression through the course based on assignment and test

marks. The analysis cumulates in a final performance placement for each student based on their final marks. Section 3.3 elaborates on the method of skills mapping.

3.1. Gap modelling method

The method described below for establishing a gap model is influenced by the approach used by Novak and Cañas (2008) and (Miller, Imrie et al. 1998) for concept mapping in curriculum evaluation and planning, and by Spinuzzi's (2002) genre ecology mapping technique. For the purpose of this paper, the concept mapping approach is adapted for use in capturing the relationship between course activities and learning difficulties. A gap model is developed for a course running in a particular year, and is performed once the course has been completed for that year. Data is collected while the course is running. The lecturer is requested to document structural elements of the course and record activities performed during the course. Most of these records are either generated by the lecturer during his or her usual course preparation duties, or are implicitly generated when course resources are created and online correspondence is handling. Accordingly, the data comprise files and documents archived by the course lecturer or added to the course website. Once the data has been collected, it is grouped into categories. This grouping process provides a means of abstracting multiple occurrences of similar features into collections handled collectively (following an object-oriented approach to systems thinking (Sage, Armstrong et al. 2000)). For the purpose of this paper, eight categories of course-related data were used, namely:

1. Syllabus descriptions: the course syllabus handed out to students in the first lecture, together with documents describing the syllabus to be examined in tests.
2. Lecture material: lecture slides, notes and handouts.
3. Test material: marked students' test and exam scripts.
4. Assignments: course assignments, projects descriptions and documentation related to laboratory practice and tutorials, including assignments and reports submitted by students.
5. Teamwork logs: records (e.g., email and logs from lecturers and names on assignments) that reflect the composition of project and tutorial teams.
6. Records from teaching tools: archived forum postings, and online polls.
7. Other: archived email messages and announcements sent from lectures to class or to the lecturer.
8. Mark sheets: marks allocated to students for the various course activities.

Once the course material was categorised, the data in each category was analysed to determine common areas where students showed strengths or weaknesses. The syllabus documents (category 1 data) were first investigated to develop a list of subject areas where students were expected to develop skills – this list, in relation to subject literature, was used to identify where bridges existed between students' prior knowledge and the new knowledge being developed. The lecture material (category 2 data) was used to determine which aspects of the subject matter were focused on, or repeated and thus likely to reflect knowledge areas that students found difficult to understand. In terms of the test material and assignments (category 3 and 4 data), exam scripts were focused on, and these were studied in relation to the syllabus listing to determine how students had answered questions, developed solutions, and to identify frequently occurring difficulties that students experienced. The mark sheets (category 8 data) were used to determine the relative importance of the different parts of the syllabus, and to find how well students learned the needed skills. Teamwork records (category 5 data) were used to gain insights into the composition of student project teams – these records were used to determine team profiles, for example whether strong students combined with weaker students (such team compositions possibly causing unrepresentative marks). Records that were generated by teaching tools (category 6 data) were studied to identify further areas of difficulty. The other forms of data (category 7 data) were primarily used as a means to gain further insights into students' problems and strengths.

From the above analysis, a list of the most important knowledge areas (corresponding to parts of the syllabus) was obtained. Lists of students' common areas of strengths and weaknesses were produced for each knowledge area, with a focus on the more commonly occurring strengths and weaknesses.

This information was visually represented in the graphical 'gap model' as follows: each knowledge area was drawn as a rounded box enclosing keywords describing the knowledge area; each kind of weakness (i.e., learning difficulty) was drawn as an octagon enclosing keywords identifying the weakness; and each area of strength was drawn as a rectangle enclosing keywords describing the strength. Each knowledge area was drawn in the middle of the diagram, strengths drawn in top section, and weaknesses in the bottom section. Lines linked weaknesses and strength to knowledge areas in which these strengths or weaknesses were found. Duplicate weakness or strength boxes were not added to the model; instead the same box was linked to the knowledge area concerned. This method resulted in models with many inter-crossing lines linking the various blocks; however, as the next section explains, it is the number of lines joined to each block that is more relevant to the evaluation process,

rather than being able to see blocks as interconnected (since blocks with many connections identify a common area of weakness or strength). Section 4 shows the models produced using this method (see Figure 2).

3.2. Gap models comparison method

At first inspection it was clear that there were similarities and differences between the models. However, the following explicit method was followed to perform the comparison. The model comparison method worked on the two models produced, following the method described in Section 3.1. This approach identified the required knowledge areas of the curriculum, and mapped the students' strengths and weaknesses in relation to these. The following labelling approach was used for both models independently:

1. The knowledge areas with the most number of connections to weaknesses were found. For the purpose of simplification, the three main areas of difficult were identified. These knowledge areas are labelled W1 to W3 (in order of priority).
2. The three knowledge areas with the most number of connections to strengths were found. These knowledge areas are labelled S1 to S3 (also in order of priority).
3. The three weaknesses with the most lines are labelled 1 to 3 as per step 1.
4. The three strengths with the most lines are labelled 1 to 3 as per step 1.

The annotated models now indicate the knowledge areas of greatest strength or weakness, as well as the most commonly occurring weaknesses and strengths regardless across all knowledge area.

3.3. Skills matrix mapping

For the skills mapping approach, a baseline map was produced using results of the NBTs and results of the department-administered computer literacy assessment test (the first practical test perform in the EEE1003W course). A second map was developed using the students' marks for the final written exam and their combined computer practicals mark. Section 4.3 shows the maps that resulted.

The first map was developed by visually representing each student in one of four quadrants on a two-by-two skills matrix according to their results for the NBTs and computer literacy assessment test. The average of all three NBTs was used (i.e., academic literacy, quantitative literacy and mathematics scores). Each student that received an average NBTs result of 51% or above was located in one of the upper quadrants. Each student that obtained computer literacy results of 70% or more were placed in one of the quadrants on the right-hand side. In order to make the visual maps easier to interpret, students were grouped into set of 10 and an icon representing a group ten was placed in the diagram. Following this approach, groups of students in the upper right-hand-side quadrant correspond to students with high potential for achieving a high mark in the course.

The second map was developed in a similar way to the first. Students were located vertically based on their final exam results and horizontally based on their combined computer practical and project mark. Students that failed the course were placed in the lower left quadrant. Students that passed the course and achieved 50% or more for the exam were placed in a upper quadrant. Students that passed the course and got 60% or more for their combined practical and project mark were placed in a quadrant on the right.

4. Results

The results are presented in this section. Section 4.1 presents the gaps models developed from the EEE1003W 2008 course data. Section 4.2 gives the gaps models developed from the EEE1003W 2009 course data. Section 4.3 shows results of the gaps models comparison. Section 4.4 gives results of the skills mapping analysis. Section 5 continues with a discussion of the results and presents conclusions for this paper.

4.1. Results for 2008

The analysis of syllabus documents (category 1 data as defined in Section 3.1) resulted in a list of over fifty knowledge areas. This list was cut down to a more manageable number of a dozen areas by generalizing related areas. These generalized knowledge areas are listed below and specific instances of syllabus content that fits within the areas are described in parentheses.

1. Use of word processing software (e.g., using Microsoft Word to write reports).
2. Use of spreadsheet software (e.g., check calculation results, check results produced by programs).
3. Computer history (brief history of computers and their use in solving engineering problems is covered in the course).

4. Program design and planning (e.g., drawing flowcharts, writing pseudo code, describing steps of computation, understanding difference between top-down vs. bottom-up design approaches).
5. Explaining programs and computer-based solutions (e.g., explaining programs, what they do, how they solve a particular problem).
6. Identifying tasks suitable for computer-based solutions (e.g., ability to determine if it is possible, and with the effort, developing a computer-based solution to an engineering problem).
7. Programming fundamentals (e.g., variables, control structures, data types, compound data types, defining functions, recursion, modules, libraries, user input, displaying results, reading and writing files, measuring performance of programs, concept of pointers, and code reuse).
8. Essential Python programming skill (ability to write and run simple Python programs).
9. Essential C programming skill (ability to write, compile and run simple C programs).
10. Writing programs to solve engineering tasks (most of the learning and assessment tasks are separated between C and Python coding tasks – these are generally more complex tasks drawing on a broad range of a student's design and programming skills as well as electrical engineering knowledge).
11. Debugging code (finding and correcting errors in code).
12. Object-Oriented concepts (e.g., understanding of basic object-oriented programming concepts and ability to represent these using Python programs).

For each of these knowledge areas, common strengths and weaknesses were determined by looking through the course resources described in Section 3.1. Fourteen main strengths were found. These observations included certain tools being learned quickly, particular kinds of test questions generally being answered well, and ability to effectively develop simple programs. Six main weaknesses were observed, such as difficulty in learning advanced techniques, and misinterpretation of questions. There were clearly many other strengths and weaknesses observed (e.g., some students had difficulty expressing solutions), but as mentioned in Section 3.1 this analysis focuses on commonly occurring strengths and weaknesses. Figure 2 shows the resultant gaps model.

The gaps model in Figure 2 shows that the three areas of greatest strength were: 1) essential Python programming; 2) identifying tasks suitable to computer-based solutions, and 3) essential C programming. The knowledge areas of greatest weakness were: 1) essential C programming, 2) debugging code, and 3) use of object-oriented concepts. Interestingly, essential C programming was observed to be both a strength and a weakness; this is likely due to some aspects of C programming were found particularly challenging (especially use of pointers as per test answers) whereas other areas were found comparatively easy (e.g., displaying results). The most common strengths were 1) effective use of office software, 2) effective use of web and search engines, and 3) ability to extend techniques learned from examples to other problems. Weaknesses included 1) difficulty in applying advanced techniques, 2) difficulty in using debugging tools and 3) misinterpreting questions.

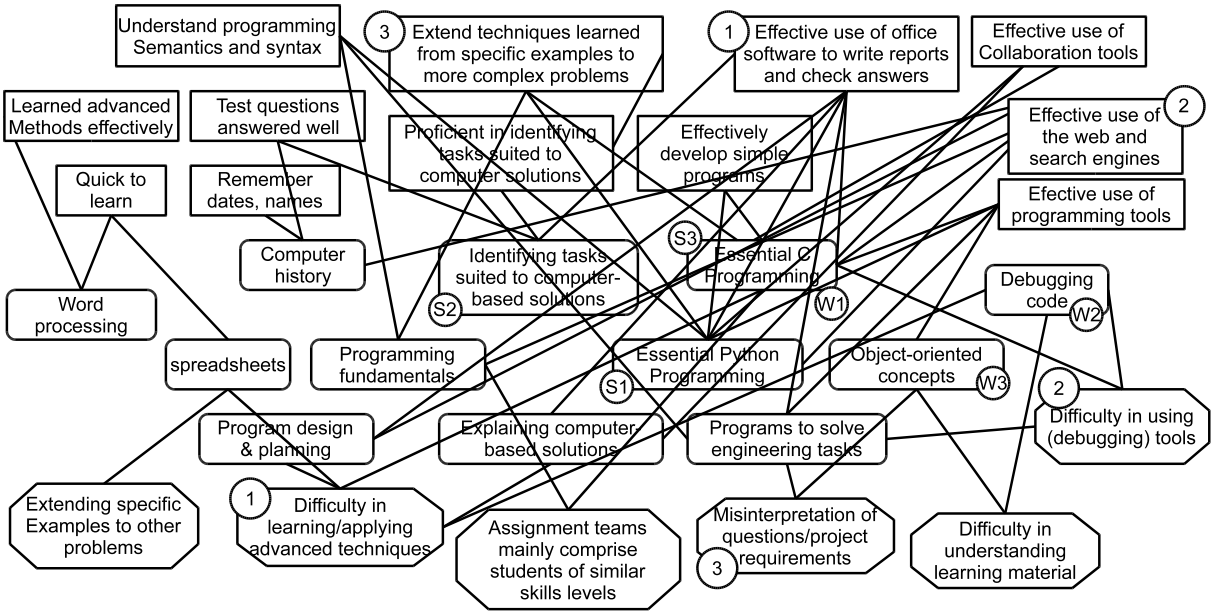

Figure 2: Gaps model for EEE1003W in 2008

4.2. Results for 2009

The 2009 syllabus documents for EEE1003W were largely the same as in 2008. Differences in implementation included more projects and laboratory assignments in 2009. In 2009 there were significantly more tutorial and

homework exercises, in addition to extra tuition classes that supplemented regular lecturers and tutorials. Consequently, there were not only more students, but each student did more assignments (albeit smaller assignments) than in previous years. Figure 3 shows the gaps model that resulted from analysing the 2009 data.

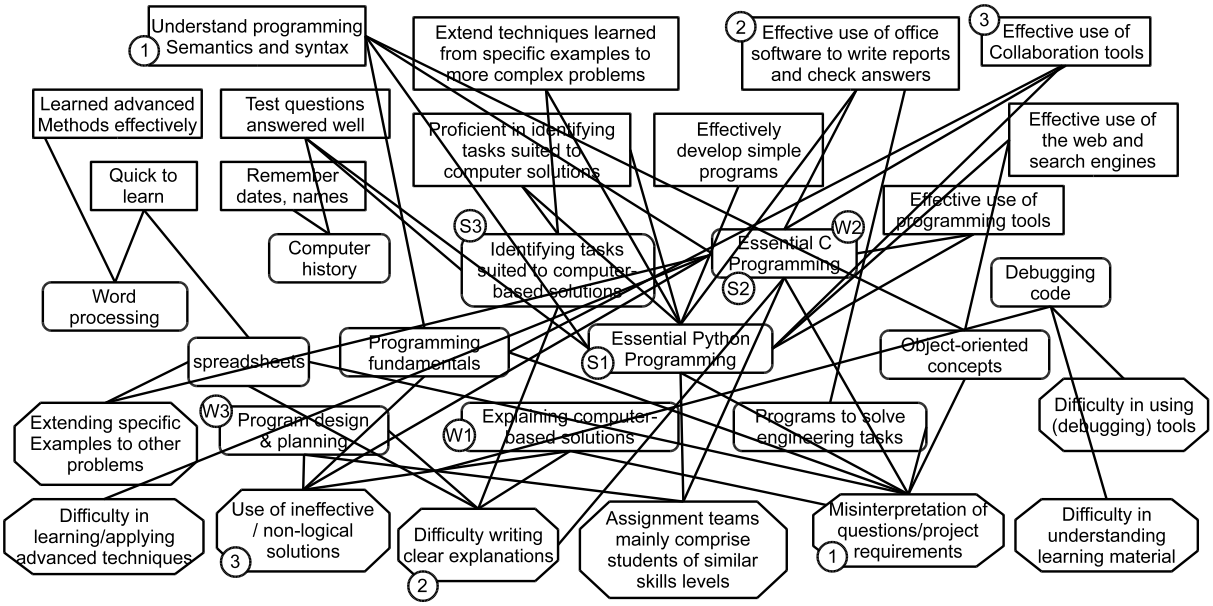

Figure 3: Gaps model for EEE1003W in 2009

Analysis of the gaps model shows that the knowledge areas of greatest strength were: 1) essential Python programming, 2) Essential C programming and 3) identifying tasks suited to computer-based solutions. Commonly occurring knowledge areas of weakness included: 1) explaining computer-based solutions, 2) essential C programming, and 3) program design and planning. The greatest strengths were: 1) understanding program language syntax and semantics, 2) effective use of office software, and 3) effective se of collaboration tools. The main weaknesses were: 1) misinterpretation of questions and project requirements, 2) difficulty in writing clear explanations, and 3) use of inefficient or non-logical solutions.

4.3. Comparison of gaps models

The two gaps models shows show that students in both 2008 and 2009 showed the same set of knowledge area strengths, namely that they effectively learned the fundamentals of Python programming and C programming, and showed competence in identifying tasks suited to computer-based solutions. There was a slight difference in that 2008 students collectively showed stronger C programming ability than those in 2009.

There were differences between in the most commonly occurring knowledge areas of weaknesses. In 2008, the greatest area of weakness was C programming. In 2009, the weakest area was explaining computer-based solutions. In 2008, object-oriented concepts and debugging code were identified as other areas of significant weakness; but neither or these areas featured as significant weaknesses in 2009. In contrast, students in 2009 had difficulty with program design and planning, whereas in 2009 this was not identified as a significant area of weakness. There are many potential reasons for this discrepancy. The 2008 students worked on larger and more complex C programming projects, and this could be why C programming was indicated as the greatest weakness. However, in reflection to the examination scripts and project work investigated, this difference is more likely attributable to the 2009 having more frequent difficulty in writing and explaining program solutions, and possibly not understanding what is asked of them. The 2009 list of difficulties supports the previous point, in that the greatest weaknesses were listed as misinterpretation of questions or requirements, students having difficulty in writing clear explanations, and students using inefficient or non-logical solutions. Students in 2008 showed a different profile of difficulty, such as difficulties in applying advanced techniques and using debugging tools.

4.4. Results of skills mapping

Results of the skills mapping technique are shown in Figures 4 and 5. Figure 4 shows the baseline map, and Figure 5 shows the map that reflects the skills profile of the class at the end of the course.

The baseline map shows a positioning of 152 students in the course. Although there were 167 students at the start of the year, 15 of the students did not have a complete set of NBTs (i.e., they may have been repeating the course, taken leave of absence, etc.) and are therefore not represented in the diagram. Figure 4 shows that 93

students received averaged NBTs scores above 50%, but of these only 33 got high marks for computer literacy. The upper quadrants have been labelled as 'potential' and 'high potential' to respectively reflect that students with good NBTs are expected to do well (as highlighted by (MacGregor 2009)), and students that achieve good NBTs and also have good computer literacy are expected to do particularly well for the course. The lower right quadrant is labelled 'at risk' because these students have some computer literacy and therefore might still make the grade for the course even though their NBTs were low. The lower left quadrant is labelled 'low potential' because students in this quadrant are unlikely to pass the course because they showed both poor computer literacy and weak results for the NBTs.

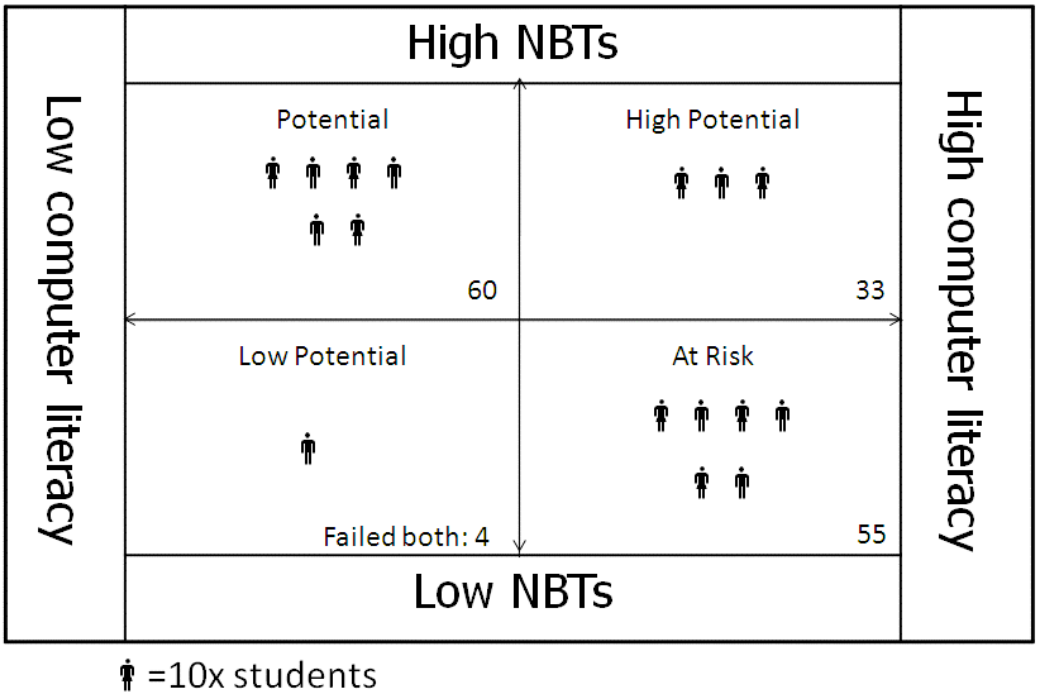

Figure 4: Skills map 1: placement of students based on their NBTs and computer literacy results.

Figure 5 shows the skills mapping of students at the conclusion of the course. Not all the students completed the course. Students that transferred to other programmes or dropped out are not shown on the second map 2 (Figure 5). The figure shows 150 students. The discrepancy in 17 students (between 167 who started and 150 students at the end) is attributable to students dropping the course or transferring before the final exam. The results shown in Figure 5 include students that did not receive a duly performed rating and could not write the final exam.

As is shown in Figure 5, 127 students passed the course, and 23 were recorded as failing. Of the students that passed, 84 (66%) achieved 50% or above for both the final exam and combine computer project and practical marks. 32 (25%) of the students that passed got above 50% for the exam but below 50% for the project and practical work. The remaining 11 students (9%) failed the final exam but did sufficiently well for their project and practical work to enable them to pass the course. Consequently 43 (34%) of the students that passed still exhibited some form of weakness in relation to computing.

In relation to marks achieved by individuals, the all students in the 'high potential' quadrant remained in that quadrant, and were for the most part joined mainly by students who were initially in the 'potential' quadrant. Students in the 'low potential' quadrant remained in that quadrant and were generally joined by students from the 'at risk' quadrant (all but one of the students originally in the 'low potential' quadrant dropped the course and are not recorded in Figure 5). Students in the 'at risk' quadrant tended to remain in that quadrant or move into the 'low potential' quadrant.

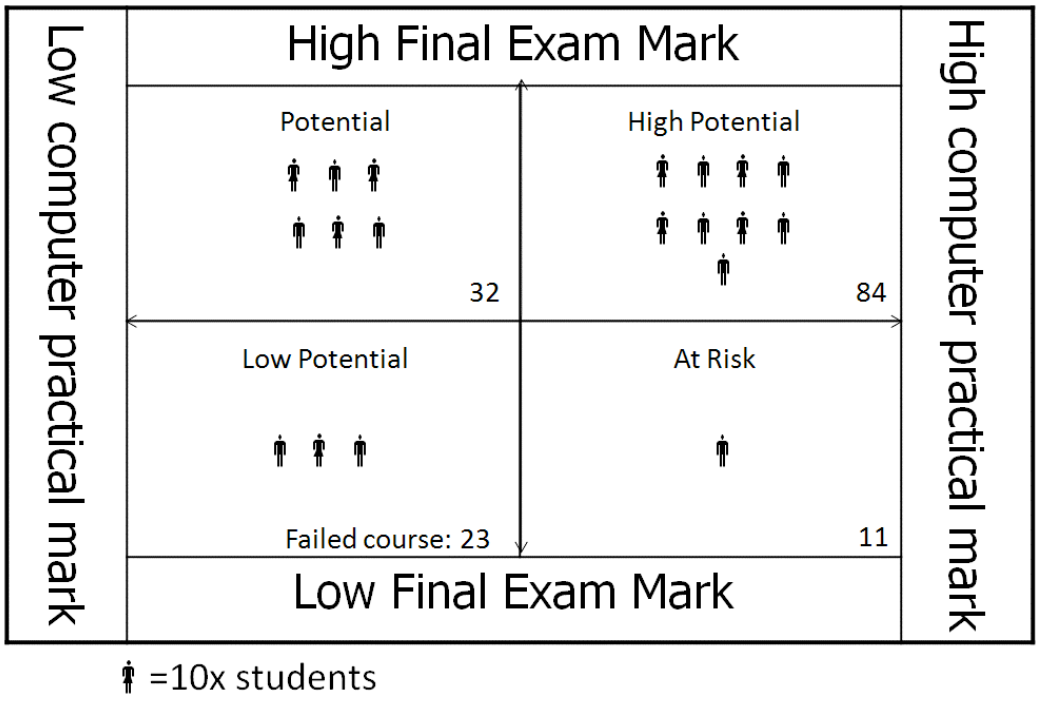

Figure 5: Skills map 2: placement of students based on their final course marks.

5. Reflections and conclusions

This study has shown that major difficulties student experienced in the course in 2009 were: 1) misinterpretation of questions and not understanding requirement or problem descriptions; 2) difficulty in explaining their solutions; and 3) using poorly developed, or non-logical, solution strategies to solve complex problems. Students in 2008 had a different set of difficulties that were more related to technicalities of programming, in particular applying advanced techniques and using debugging tools. It consequently seems that students in 2009 had more difficulties at a fundamental level, which overshadowed difficulties related to higher-level tasks that were more pronounced in the previous year. This finding may allude to students coming into the course without sufficient basic skills that are needed to comprehend the questions asked and to express complex solutions. These problems may correspond to findings from the 2008 NBTs, where weaknesses in mathematics proficiency and quantitative literacy were noted (MacGregor, 2009).

Common areas of students' competence included: 1) understanding of program language syntax and semantics; 2) effective use of office software (e.g., word processors and spreadsheet programs); and 3) good online collaboration skills. These competencies were seen in both 2008 and 2009, and shows evidence that the students can effectively learn computer and programming skills.

The 2009 pass rate for EEE1003W was only slightly lower than previous years (although this does not take into account the 6% drop-out of students early in the year which was slightly higher than previous years). The 2009 bubble has deflated somewhat, and many of the weak students have progressed to second year. The first-year intake come down to 219 this year (B. Downing, email to author, 10 February 2010), closer to the planned target of 180 students.

The matrix mapping approach showed that the cohort of students that passed their NBTs, with or without good computer proficiency, successful completed the course. Students with intermediate scores for the NBTs, and good computer skills, generally completed the course successfully. However, students with below average NBTs did not pass. Similarly, students starting with low results for the NBTs, but high computer proficiency, generally did not pass the course. Consequently, a recommendable strategy to improve the pass rate may be to ensure that students within the 'at risk' quadrant are provided extra support to develop the fundamental skills that they need to understand complex questions and to express and explain technical solutions.

While the progression shown using the matrix mapping approach indicated that students in the bubble generally did achieve improvement. However, the second skills map, which was based on the final exam and practical work, and supported by the gaps models, indicated that the learning difficulties described above remained predominant at the end of the course. Consequently, these problems are likely to reoccur in the students' next year of study.

Acknowledgements

Thanks to Robyn Verrinder and Mohohlo Tsoeu for access to EEE1003W 2009 course material, and for their time spent discussing the course with me. Thanks to Renee Smit for provision of 2009 NBTs results. Thanks to Heather Leather for providing first-year intake statistics.

References

Darzentas, J., C. A. Nicolle, et al. (2004). "Identifying core knowledge and skill sets for model curricula: update."

Davison, A. (1995). "Teaching C after Miranda." Funtional Programming Languages in Education: 35-50.

EBE. (2010). "Faculty of Engineering and the Built Environment Student Handbook." Retrieved 8-Aug-2010, from http://www.uct.ac.za/downloads/uct.ac.za/apply/handbooks/fac_ebe_2010.pdf.

Harbison, S. P. and G. L. Steele (1994). C: a reference manual, Prentice Hall PTR Upper Saddle River, NJ, USA.

Leather, H. (2009). Electrical undergraduate enrollment figures, Faculty of Engineering and the Built Environment.

MacGregor, K. (2009). "South Africa: Shocking results from university tests." World University News – Africa Edition(35).

Miller, A. H., B. W. Imrie, et al. (1998). Student assessment in higher education: a handbook for assessing performance, Routledge.

Nagurney, L. S. (2001). Teaching introductory programming for engineers in an interactive classroom. 31st ASEE/IEEE Frontiers in Education Conference, Session S2C, Reno, NV.

Novak, J. D. and A. J. Cañas (2008). "The theory underlying concept maps and how to construct and use them." Florida Institute for Human and Machine Cognition Pensacola Fl, www. ihmc. us.[http://cmap. ihmc. us/Publications/ResearchPapers/T heoryCmaps/TheoryUnderlyingConceptMaps. htm].

OpenOffice.org. (2010). 10-Aug-2010, from http://www.openoffice.org/.

Python.org. (2010). "Python Programming Language – Official Website." Retrieved 10-Aug-2010, from http://www.python.org/.

Sage, A. P., J. E. Armstrong, et al. (2000). Introduction to systems engineering, Wiley New York.

Seltzer, K. and T. Bentley (1999). The creative age: Knowledge and skills for the new economy, Demos.

Spinuzzi, C. (2002). Modeling genre ecologies. Proceedings of the ACM 20th annual international conference on computer documentation (SIGDOC). Toronto, Ontario**:** 200-207.