Proceedings of the NACCQ 2000
Wellington NZ www.naccq.ac.nz

A Case Study of Portfolio Assessment in a Computer Programming Course

Beryl Plimmer

Manukau Institute of Technology
Manukau City, New Zealand
Beryl.Plimmer@Manukau.ac.nz

ABSTRACT

It is common practice in the creative arts for students to prepare a portfolio of work and for this to be the gauge of their achievement. It's obvious that an exam would not be appropriate for photography, painting or pottery. Clearly the students need to be able to load the film, mix the paint, or spin the wheel, but this does not an artist make.

Simonyi, one of the foundation members of Microsoft and architect of MS Word and Excel, Kildall, the author of the first microcomputer operating system CPM, and Bricklin, the author of VisiCalc, all describe programming as an art, a science and a skill (Lammers 1996).

Educators generally describe programming in the reverse order. Programming requires detailed knowledge of: the programming language syntax (skill) and problem decomposition and algorithm construction (science) (Shneiderman 1980; Linn 1985; Maheshwari 1997).

No direct link is seen between practitioners' 'art' and educators' descriptions of programming.

We have experimented with portfolio assessment in two programming courses. Our goal was to allow the students more freedom in creating programs while at the same time requiring them to demonstrate the required syntax and problem decomposition. This paper will discuss the planning requirements for successful implementation of portfolios, the students' reactions and how well the portfolios met the learning requirements of the courses.

1. INTRODUCTION

Recently there has been a shift to authentic assessment, assessment that mirrors the real-world and is integrated with learning. Assessment where the learners are active participants and the criteria are open and negotiable (McDowell 1999). The goal is to engage learners in the assessment as well as the learning. This engagement will assist the learner to develop better learning and self-evaluation skills that are so vital as we move to a society where life-long learning must be the norm. Portfolio assessment is perhaps the most common form of authentic assessment.

Programming is a combination of skill, knowledge and creativity (Lammers 1996). Traditional teaching and assessment stress the skill and knowledge aspects with little regard for creativity. Creativity is becoming more important for the average programmer with the use of visual environments. We believe that this aspect of programming should be included from the beginning.

NACCQ
FOSTERING COMPUTING EDUCATION IN NEW ZEALAND

2. BACKGROUND

2.1 Learning to Program

Programming is a complex task. There are three interrelated parts to programming, knowledge of the programming language, the ability to break problems down into a algorithm and finally the creative ability be able to see the completed project at the inception stage.

Programming languages and host software with their complex syntax and rules take some time for beginners to master but they become automatic for the experienced programmer. Algorithm construction and problem solving are more challenging activities. The common approach to problem solving is to decompose the problem in small parts and then solve each part. However the depth of the hierarchy that this creates even for relatively simple programs confronts the programmer with complexity ratios in the region of 109 (Dijkstra 1998). Dijkstra claims that the conceptual hierarchies are deeper than humans have ever had to handle before and present a radically new intellectual challenge.

The nature of digital representation imposes another challenge for the programmer. The alteration of one bit of data or one character of program code can mean the difference between success and total failure (Dijkstra 1998).

The final part of programming is the art of creating really great software. Expert programmers can visualise the completed program from the problem definition. They then work by decomposing a problem into parts that they can fit to pre-existing templates or structures (Soloway and Iyengar 1986; Linn and Clancy 1992). Linn (1985) describes these as stereotypical patterns of code that use more than a single language feature. However in order to do this, the programmer first must be able to mentally formulate a solution. This is the creative, artistic part of programming that is the most difficult to master (Lammers 1996).

The challenge for the student was to become proficient at a programming language while at the same time developing problem solving and problem decomposition skills. Alongside this, their creativity needed to be stimulated so that they can mentally compose a problem solution. Bill Gates, Microsoft Corporation, described it thus, "You have to simulate in your mind how the program's going to work and you have to have a complete grasp of how the various pieces of the program work together" (Lammers 1996 p.73). We believe that portfolio assessment is likely to give the student more opportunity to engage in these separate aspects of programming.

2.2 Portfolio Assessment

A portfolio is a purposeful collection of a person's work (Lester and Kroll 1991; Arter and Spandel 1992). There are many ways that portfolios can be assembled. They may be a collection of all work, or a limited selection. They may include planning and intermediate drafts, or only completed work. The selection of work to be included may be the responsibility of the teacher, the student or both.

With portfolio assessment the teacher may specify what the student should do, or what skills he or she should demonstrate. Portfolio assessment has traditionally been used in creative fields such as fine art and music. More recently it has become common in a much wider range of educational settings (Gardener 1992). A well-designed portfolio will meet the teachings goals of engaging the students in the higher order cognitive of activities of reflection and creativity (Arter and Spandel 1992; Gardener 1992; Biggs 1998). At their best, portfolios can act as a silent mentor becoming an instrument of learning as well as a repository and students become a responsible partner in documenting and evaluating their own learning (Gardener 1992).

There are a number of key points that must be addressed when setting up portfolio requirements. Firstly, the purpose to which the portfolio will be put must be clearly defined. Is it to be used for formative or summative assessment? If formative, what percentage of the final mark is it? Secondly, the required content must be agreed on, this may be a list of tasks that must be completed or skills to be demonstrated. The student may be required to reflect on and evaluate their practice, or simply keep a file of their work. Thirdly, the criteria on which the portfolio will be judged must be explicit. The criteria may include the knowledge and skills demonstrated, the reflection demonstrated, the response to feedback, and the number and range of entries (Arter and Spandel 1992; Gardener 1992).

We have used portfolio assessment in two programming courses in our Bachelor of Information Systems degree, the introductory programming course and in a stage two, second programming language course. The criteria for each of the portfolios were quite different. Each was successful but had some problems that have required us to refine the process.

3. USE IN AN INTRODUCTORY PROGRAMMING COURSE

We have run this course three times using portfolio assessment and have refined the process with each offering. The course taught structured programming using C and associated topics such as program development life cycle, numbers systems, program testing and programming languages. The students were required to submit their portfolio three times during the semester. For each submission they included pieces of work that demonstrated mastery of specific topics. The portfolio was 60% of the course mark. There was also a final examination that contributed the other 40%. It covered the theory topics such as number systems and retested the material covered in the portfolio under controlled conditions.

The first submission (week 5) covered basic language syntax, primitive data types, assignment selection and iteration instructions and simple functions. The first time we used portfolios we specified that one program had to be 50 lines of code but did not put a limit on the number of programs that could be submitted. The best students submitted one or two programs and as the number of submitted programs increased, the quality decreased. A few of the students put in ten programs of about six lines of code. The next time the course was offered we limited the number of programs to four. This encouraged to students to write and select programs that were meaningful.

In the second submission (week 11) the students submitted three pieces of work, two programs and an essay. One program was required to demonstrate the use of arrays and strings, and passing of arrays and strings to functions. For the other program they had to find a small program in another 3GL language such as Pascal, Fortran or Cobol that included at least two data types, a selection statement and an iteration statement, photocopy it, and convert it to C. The hardest part of this exercise was finding the right program. Their search got them reading code and recognising the common factors in all structured programming regardless of language. The essay was on a topic associated with programming, program testing has been the most useful essay topic we have tried. Many of the students commented that researching testing techniques made them think more about design and how they were going to test their programs before they started to write code. They also appreciated just how hard it was to write truly correct programs.

The third submission (week 14) was another program. This time they used a data file that had been given to them and demonstrated adding, changing and deleting records and a couple of standard algorithms such as a sort and a control break report.

A tutor and the student marked the first two submissions together. We checked that the criteria had been met and discussed any errors or omissions. Each student was asked to say the grade they believed their work merited. Usually their self-evaluation was fairly accurate, if not we discussed with them the grade they were being given and why. They had the opportunity to improve any of the work from submissions one and two and have it marked again with submission three.

The third submission was handed in at the end of classes so we did not require the students' presences during marking, however by this stage most students valued the feedback they got discussing their work with the marker and chose to attend.

We have found some clear advantages of using portfolios in this course. Students got a real buzz out of being able to write programs from their own choosing. We have had programs on share prices, investment returns, dating agencies, cricket statistics, and games. Some students followed the same theme through all their programs. Others use different problems for each program. Some of the students had trouble thinking up appropriate problems particularly for the first submission; we now include a list of vague ideas, but also suggest they look at some other programming books to get ideas.

Of course there have been some problems. Some students have copied code from a book or the internet. It was very obvious when in the first submission we got code that used advanced techniques and asked the student to explain what it did or how it worked! The penalty was a zero mark. And although we have found it a good way to engage students in problem definition and program construction it did not encourage students to learn the language syntax to the level where it is automatic. We were distressed to find students really learning the language syntax just in time for the final examination when they should have learnt the basics in the first few weeks. The problem with this is that if you are trying to remember the syntax for a for loop, at the same time as understand how a sort algorithm works, you get short term memory overload. We introduced formative tests at the beginning of each lab for the first few weeks, simple tasks like adding two numbers and displaying the result and stressed the importance of rote learning syntax. Most students seemed get the message when the quickest students had finished the program in a minute and they were still going after ten minutes. However this is not a total solution and we may introduce a summative syntax test in about week four.

Prior to the first marking session we worried about being able to fairly attribute marks, particularly in that there were two of us doing the marking. In reality this was not a problem, we were very careful to consult and moderate the first time through. It became very clear that we had a consensus on what constituted unsatisfactory, ok, good or excellent work. For most students there was a close correlation between their portfolio mark and final examination mark. We do get a few exceptions, generally people who did not do their own portfolio work. They usually fail the examination. We have felt comfortable with the final results.

4. USE IN A STAGE TWO SECOND LANGUAGE COURSE

The second course we have used portfolio assessment in has been offered just once, last semester. It was a stage two applications development course where the students learn visual basic and then use it to develop a project using RAD (rapid applications development) prototyping. We used the portfolio for the assessment of the learning of Visual Basic. This constituted 25% of the final mark. The project was 35% and the final examination that covered the theory was 45%.

For this portfolio we simply required the students to complete one exercise from each chapter of the textbook. We went through the exercises and selected 2 - 4 exercises from each chapter that were of about equal difficulty and demonstrated the knowledge and skills of the chapter. The students could choose which of of the selected exercises they submitted. This was clearly very different to our approach in the first course, however our goals were different. The students in this course were 2nd and 3rd year students who had done programming, systems analysis and database courses (as a minimum). We assumed knowledge of programming and problem solving, what we wanted them to do was learn visual basic and practise presenting professional looking programs. We encouraged them to expand on what was in the book and told them doing only what was in the book would give them a mark of about 75%. To get more than this they would have to demonstrate independent learning.

This portfolio was very successful at getting the students competent with Visual Basic. Some of the better students took up the challenge and went way beyond the requirements either in interface design or technical areas, one however, found it too boring to be bothered.

The average students enjoyed it and found it an easy way to learn.

Unfortunately we had dug a real hole for ourselves with the marking. For each student (22) we had ten programs to mark. It took hours! We did not have any problems with plagiarism with this group, as is normal with 2nd and 3rd year students. Next time we will probably adopt a more pure form of portfolio assessment and specify the skill and knowledge we want demonstrated and limit the number of programs per student to two - three.

5. SUMMARY

We have found portfolio assessment to be a useful tool for programming courses. Clearly the way in which we defined the portfolio requirements for these two courses were quite different and appropriate for the learning we wanted to encourage. The advantage of portfolios is that they encourage the students to actively review and select their work. This review process engages them in the higher-order cognitive activities we wish to encourage.

We believe that portfolios are most appropriate when used in conjunction with other assessment methods such as tests or examinations. Although there was a close correlation between portfolio and examination marks we could not rely on the portfolio marks alone in the introductory course because of cheating. Careful planning is the key to all assessment and portfolios are not an exception to this. A well-designed portfolio assessment can provide an enriching learning environment for students. In the words of one of the introductory programming course students “This portfolio is the coolest thing I have ever been allowed to do”

6. REFERENCES

Arter, J. A. and V. Spandel (1992). "Using portfolios of student work in instruction and assessment." Educational Measurement: Issues and Practice 11: 36-11.

Biggs, J. (1998). What the student does: Teaching for enhanced learning in the '90s. Higher Education Research and Development Society of Australasia, Auckland.

Dijkstra, E. W. (1998). "On the cruelty of really teaching computing science." Communications of the ACM 32(12): 1398-1413.

Gardener, H. (1992). Assessment in context: The alternative to standardized testing. Changing Assessment: Alternative views of aptitude, achievement and instruction. B. R. Gifford and M. C. O'Connor. Boston, Klower: 77-119.

Lammers, S. (1996). Programmers at work: Interviews with 19 programmers who shaped the computer industry. Redmond, Microsoft Press.

Lester, F. K. and D. L. Kroll (1991). "Evaluation: A new vision." Mathematics teacher 84: 276-283.

Linn, M. C. (1985). "The cognitive consequences of programming instruction in classrooms." Educational Researcher 14(5): 14-16, 25-29.

Linn, M. C. and M. J. Clancy (1992). "Case for case studies of programming problems." Communications of the ACM .Mar 1992 35(3): 121-132.

Maheshwari, P. (1997). "Improving the learning environment in first-year programming: Integrating lectures, tutorials and laboratories." Journal of Computers in Mathematics and Science Teaching 16(1): 111-131.

McDowell, E. (1999). "Editorial." Assessment & Evaluation in Higher Education 23(4): 335-338.

Shneiderman, B. (1980). Software psychology. Cambridge, Massachusetts, Winthrop Publishers.

Soloway, E. and S. Iyengar, Eds. (1986). Workshop on empirical studies of programmers. Human/Computer Interaction. New Jersey, Ablex Publishing Corporation.