Determining Areas of Weakness in Introductory Programming as a Foundation for Reusable Learning Objects

Eileen Costelloe, Elisabeth Sherry and Patricia Magee
ITT Dublin, Ireland
eileen.costelloe@ittdublin.ie
elisabeth.sherry@ittdublin.ie
patricia.magee@ittdublin.ie

Abstract: Teaching programming to novices has proved challenging for both learner and lecturer due to the abstraction and complexity of the subject matter. The work described in this paper is part of an EU funded Minerva project called TUPULO (Teaching Undergraduate Programming Using Learning Objects) which aims to address the challenges faced by novice programmers by providing them with an innovative learning tool. This learning tool that is currently under development and rollout incorporates a set of Reusable Learning Objects (RLOs) based on sound pedagogical principles and encapsulated in a Constructivist Learning Environment (CLE), which includes a meta-cognitive interface. The subject matter experts and instructional designers in the local academic partner institutions designed these learning objects. The outputs and findings of the TUPULO project will not only benefit learners in the partner institutions involved, but by being disseminated to the wider educational community, they will also help learners in the domain on a broader scale. This paper describes the preparatory work undertaken in order to establish a set of potential LOs for development based on the student's main areas of weakness.

When attempting to build learning objects for use in any domain the primary consideration should always be the needs and abilities of the learners. This paper describes the work done by the authors in conducting a user needs analysis in order to establish the key problem areas facing learners of introductory programming. A methodology for user needs capture and analysis was produced based on the set of user groups available at the Institutions and the needs of the users were captured and analysed. The methodology was devised to incorporate both quantitative and qualitative analysis of the information available to us regarding students. Exam scripts and corresponding results together with focus group discussions were used in order to ascertain perceptions regarding the course content, delivery, level of difficulty and areas of difficulty in programming. Additional institutional information such as students' leaving certificate points and Maths grade together with students' overall performance in other subject areas were used to investigate possible correlations. The analysis of this data provided some preliminary information on the ways in which students interpret various questions and their conceptual difficulties in understanding certain topics. This analysis leads to the final selection of programming topics for potential development as reusable learning objects.

Keywords: novice programmers, learning objects, programming pedagogy, meta-cognitive support

1. Introduction

Research literature and practical experience of subject experts indicate that teaching programming to novices has proven challenging for both learner and lecturer. Research supports the fact that students find programming difficult. Linn and Clancy (1992) found that "for programmers to develop competency, they need to have good problem solving skills and a thoroughly organised knowledge of a programming language". Problem-solving skills are central to developing competency as a programmer yet these skills seem to be inadequate in the incoming students. Riley (1981) concluded that many students entering college have problem-solving skills that are "woefully inadequate". Henderson (1986) notes that problem solving and analytical thinking are students' major weaknesses in a computer science course. The implementation phase of programming presents additional problems for novice programmers. These include: syntax of the chosen language, programming constructs, development environment and testing and debugging.

In addition, novice learners fail to reflect on their approach to designing solutions to problems and less successful problem-solvers act but do not look and learn from their actions (Gage and Berliner, 1980). Reflection, self-analysis, self-assessment and articulation are essential for the development of the learner's meta-cognitive and independent learning skills. Meta-cognitive skills are activated during learning, making learning easier and facilitating the transfer of learning. Fekete (Fekete et al 2000) and his colleagues acknowledge the importance of reflection in assisting students develop meta-cognitive skills. For example, by explicitly outlining subject goals, getting the students to maintain a reflective diary, students are encouraged to think about what they know, how they learned it and how well it matches the announced goals of a subject. A meta-cognitive interface will be incorporated into the learning tool to assist in developing these skills.

Reference this paper as:
Costelloe E, Sherry E and Magee P (2007) "Determining Areas of Weakness in Introductory Programming as a Foundation for Reusable Learning Objects" *The Electronic Journal of e-Learning* Volume 5 Issue 1, pp 21 - 30, available online at www.ejel.org

These skills are needed where habitual responses are not successful (Blakey and Spence, 1990) and problem solving is one area where it is necessary to develop these skills.

The TUPULO project aims to address some of the challenges faced by novice programmers by providing them with an innovative learning tool, incorporating a set of Reusable Learning Objects (RLOs), based on sound pedagogical principles and encapsulated in a Constructivist Learning Environment (CLE). The Learning Objects will focus on the common areas of weaknesses that are determined by the User Needs Analysis. As indicated the Constructivist Learning Environment will encompass a meta-cognitive interface which will encourage the novice programmers to reflect, self-analyse and elicit articulation of the learner's understanding of certain programming constructs. By eliciting articulation from the learner, the CLE is encouraging reflection "which is an important cognitive activity, critical for effective learning", (Guzdial, 1994). The outputs and findings of the TUPULO project, by being disseminated to the wider educational community, will help learners in the domain on a broader scale, as well as promote the development and use of learning processes and resources that are both innovative and effective. One of the initial core activities of the TUPULO project when it commenced in October 2005 was to classify users according to their generic need, i.e. identify a target group and to conduct a User Needs Analysis. This paper describes these activities and draws conclusions from the research carried out in one of the academic partner institutions, which will inform and direct the remainder of the research project.

The objective of this phase of the project was to establish the target audience and the students' major areas of weaknesses in undergraduate programming, so that they could be targeted and the appropriate Learning Objects could be designed and deployed. ITT Dublin carried out extensive research of their past first year students' examination scripts and analysed the results generated. The results were then collated with the results of the other academic partner institutions in order to determine cross-institutional areas of weaknesses in undergraduate programming in the student sample that could be deemed to be representative of the general population. The students, at both first and second year level, were surveyed to determine their perceptions of the courses they were taking. A number of focus group discussions with the students were also conducted.

2. Classification of participants

A core activity of the project's initial work package was to classify users according to their generic need. Research indicates that novice programmers have a number of difficulties to overcome (Riley op. cit., Henderson op. cit.). The literature also indicates that novices have poor meta-cognitive skills, (A.L. Brown, cited in Gage and Berliner,1980), these are skills which they need to develop in order to become life-long learners and proficient programmers.

This project focuses on a target audience of novice programmers in their first undergraduate year in third-level education. Three Irish third-level institutions, Institute of Technology Tallaght, (ITT Dublin), the Dublin City University, (DCU) and the Institute of Technology Blanchardstown (ITB), participated in the research, see Table 1. Samples of student data from each institution were used spanning three academic years, 2003, 2004 and 2005. These samples were broken down into sub-categories of, students in Semester 1 and students from Semester 2. The current student group from the academic year 2005/2006, at both first and second year level, were also surveyed in ITT Dublin and ITB to determine their perceptions of the courses, and ITT Dublin students were involved in focus group discussions.

Table 1: Participating Academic Partners Numbers of Students Sampled

Institution	Number of Students Sampled
ITT Dublin	157
ITB	167
DCU	311

The ITT Dublin conducted analyses of first year student examination scripts from the years 2003, 2004, 2005, see Table 2

Table 2: Breakdown of Students Sampled at ITT Dublin

Academic Year	Semester	Number of Students
2003	1	17
2003	2	64
2004	1	36
2004	2	20
2005	1	20

A total of 157 students from ITT Dublin were involved in this study to establish key areas of weakness of students of programming at introductory level. Using research literature and the academic participants' experience in the field as guidance, the target audience of novice undergraduate programmers was selected. Once the target audience was determined, the objective of the next phase of the project, which was the

User Needs Analysis, was to establish the students' major areas of weaknesses in undergraduate programming. The identified audience could then be targeted and the appropriate Learning Objects and Constructivist Learning Environment would be designed and deployed. Although this paper primarily focuses on the analysis carried out in ITT Dublin, it is worth noting that the same study was carried out in the other two academic partner institutions.

Each participating academic institution carried out extensive research of their past first year students' examination scripts and analysed the results generated. Qualitative information was gleaned by carrying out surveys and focus group discussions. A comparison of areas of weakness across all three academic institutions was carried out in order to determine common problem areas. The learning objects were then subsequently designed and developed to specifically target the subject areas posing the greatest difficulty for programming students. The choice of content for the learning objects was constrained to some degree by the need to make the topics relevant to each of the partner institutes' software development modules. The remainder of this paper details the extensive research carried out by ITT Dublin into first year students' Software Development examination scripts and presents an analysis of the results generated. The concluding section of this paper will outline the main points of interest from each of the three studies carried out in the partner institutes as part of the user needs analysis.

2.1 User needs analysis methodology

As a first step in the User Needs Analysis process, a User Needs Analysis Methodology was drafted and agreed with the participating academic institutions, as outlined in Table 3 below:

Table 3: UNA Methodology

UNA Methodology
Examination Scripts collation and analysis based on the following: ▪ Categorisation of questions, based on topic ▪ Number and percentage of students who took questions per category ▪ Students' results per question, and sub-question, F, D, C etc. ▪ Students' overall performance in the paper
Student Survey
Questionnaire of 1st year students to ascertain perceptions regarding: ▪ course, content, delivery, level of difficulty and areas of difficulty.
Questionnaire of 2nd year students to ascertain perceptions regarding: ▪ course, content, delivery, level of difficulty and areas of difficulty
Focus group discussions, 1st and 2nd year students to ascertain perceptions regarding problem areas
Institutional Information
Students' overall performance in other subject areas, where available
Leaving Certificate points where available
Course Information
Syllabus, break down of topics and time allocated to same
Teaching methodology, combination of lectures/ tutorials/ laboratories, other tools used
Technology used in teaching of same

The objectives of the UNA were to provide answers, which could address the following universal issues in undergraduate programming:

- Determine main areas of difficulty, ref Point 1 in table above
  - By scripts analysis and student survey
- Why are these areas difficult for students?
  - Student ability, Mathematical grade/Leaving Certificate Examination (Leaving Certificate Examination is the examination that Irish students take at the end of their secondary school education. They achieve Leaving Certificate points based on their performance). point correlation
    - By statistical analysis
  - How taught, examples, practical work, teaching methodology
    - By student survey
- Is Software Development the only area of difficulty in the undergraduate course?
  - Ref point 3a. in table above
  - If so what are the issues?
    - Problem – solving ability? (Ref Student Survey), How students approach the problems? No reflection on approach?
    - Is there a statistical correlation between only fail and Software Development?
    - Glean information from student survey, questionnaire and focus group discussion

2.1.1 Coding schemes

A coding scheme was generated in order to analyse the examination scripts, these codes were specific to ITT Dublin's scripts and the other participating institutions used these codes and added their own specific codes as required, see Table 4 and Table 5.

Table 4: Coding Scheme, Semester 1 (ITT Dublin)

Coding Scheme, Semester 1 ITT	Related Topic
1DArr	1DArray
2DArr	2DArray
CC	Code Comprehension
JC	Java Coding
LP	Loop/iteration
PS	Pseudocode
SEL	Selection
SEQ	Sequence and selection
STR	Strings
TST	Testgrid

Table 5: Coding Scheme, Semester 2 (ITT Dublin)

Coding Scheme, Semester 2 ITT	Related Topic
AP	Access Protection
GSM	Getter/Setter Methods
IH	Inheritance
MTH	Methods
OC	Object Construction
OCMC	Object Creation and method calling
POLY	Polymorphism
SOC	Subclass Object Construction
TH	Theory

2.2 Data collection, analysis and evaluation

A multi-method approach was adopted in this research in order to collect as much data as possible from a variety of viewpoints, which could then be analysed, and in which one could be more confident compared to using a single method approach. By triangulating the data collected by the different methods used, one can be more confident in the research findings, and the more the methods contrast with each other the more confident one can be in the findings, (Cohen and Mannion, 1997). The methodological triangulation included a survey, which included a self-completion questionnaire and focus group discussions, and an analysis of students' examination scripts over a number of semesters from the participating academic partners. In carrying out the analysis of the software development examination scripts at ITT Dublin, a MicroSoft Excel™ spreadsheet was used to store the breakdown of results for each question and sub-question for each student. This data enabled the partners determine the main areas of weakness in terms of student performance. An analysis of this data was performed in order to gather the information required as specified in the User Needs Analysis methodology outlined previously. In addition to the examination script analysis, a survey was conducted to gather data about the students' perceptions of software development. The survey involved first and second year students completing questionnaires and taking part in focus group discussions.

The main purpose of the enquiry was to:

- Determine the students' perceptions regarding the area under study, i.e. software development
- Determine the students' approach to designing software solutions

When designing the self-completion questionnaire, every attempt was made to ensure that it was clear and unambiguous. It was designed so that it would minimise potential errors from respondents and that it would engage their interest and elicit answers as close as possible to the truth. Once the questionnaires were collected, the data was entered into an Excel ™ spreadsheet for analysis. In addition to the questionnaires, focus group discussions were conducted with the students. This technique has been shown to be particularly valuable as it gets at deeper attitudes and perceptions of the attendees in such a way as to leave them free from interviewer bias. In conducting the group discussions the facilitator's guidance was kept to a minimum to maintain the criterion of non-direction. The respondent's descriptions of their experience were allowed full expression, and the range of responses from the students was maximised. The nature of the group discussions facilitated a wide range of responses with the students being able to challenge and extend each other's ideas.

2.2.1 Data collection, analysis and evaluation – semester 1

The authors conducted an analysis of first year student examination scripts from the years 2003, 2004, 2005, ref Table 2. The student sample chosen was a random sample of scripts, which were taken from the student population of each year. The analysis of the scripts for Semester 1 2003 indicated that the main areas of difficulties were questions based on the following topics, ranked in order of difficulty, based on student performance in the examination:

- 1 Dimensional Arrays
- 2 Dimensional Arrays
- Selection
- Pseudocode and Looping

The analysis of the scripts for Semester 1 2004 indicated that the main areas of difficulties were questions based on the following topics, ranked in order of difficulty, based on student performance in the examination:

- Code comprehension
- 1 Dimensional Arrays

- Looping
- Looping with Selection

The analysis of the scripts for Semester 1 2005 indicated that the main areas of difficulties were questions based on the following topics, ranked in order of difficulty, based on student performance in the examination:

- Looping/Theory
- Testing
- Looping with Arrays
- Code comprehension

An overview of these results is shown in Figure 1 below, the higher values indicating higher levels of difficulty.

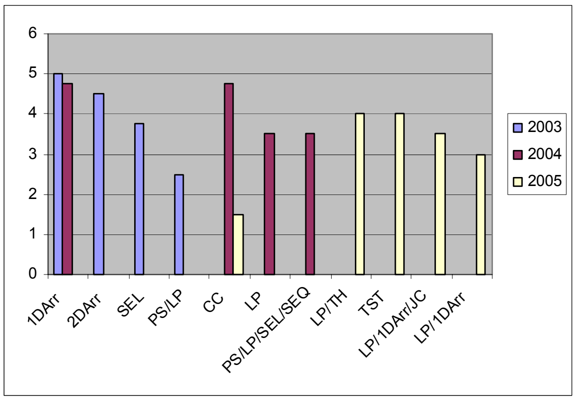

Figure 1: 2003, 2004 and 2005 Semester 1 Analysis of Examination Results (ITT Dublin) indicating main areas of difficulties based on student performance

In analysing the main topics of difficulty from each sample taken, the main areas of difficulties, in terms of programming constructs, for students in semester one in ITT were determined to be:

- Arrays
- Looping
- Selection

The survey at ITT Dublin consisted of a questionnaire and focus discussions. A total of twenty questionnaires were completed and returned. It should be borne in mind that the students who completed the questionnaires were not the same students whose data was included in the scripts analysis. However, they were randomly selected from the first and second year student population. In order to ensure that the students completed the questionnaires individually, their completion was supervised. The completed questionnaires were collected and their results analysed. The questionnaire was designed to gather information relating to the following research questions:

- Determine the students perceptions regarding the area under study, i.e. software development and its concepts
- Determine the students' perceptions regarding their approach to designing software solutions

At ITT Dublin 75% of students surveyed either strongly agreed or agreed that software development was their most challenging module. At ITT Dublin, 90% of those surveyed either strongly agreed or agreed that problem solving ability impacts on their performance see Figure 2.

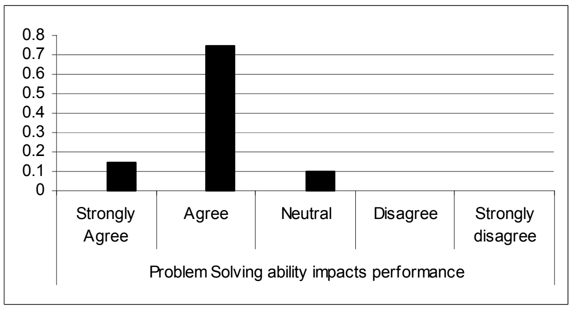

Figure 2: Problem solving ability impacts performance (ITT Dublin)

Only 20% of students at ITT Dublin nearly always think about their approach in designing software solutions, see Figure 3, with 60% only sometimes thinking about their approach and a further 20% rarely or never thinking about their approach. This concurs with the fact that novices have meta-cognitive deficiencies and these skills need to be developed. Consequently, one of the more innovative aspects of the TUPULO project is the design of a meta-cognitive interface to provide an appropriate level of support for the learner in order for them to develop their meta-cognitive skills.

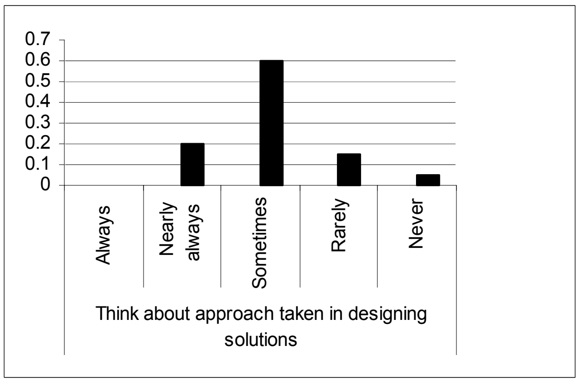

Figure 3: Think about approach taken in designing solutions (ITT Dublin)

The students were asked to rank the difficulty level in a number of programming concepts, e.g. loops, arrays, selection. Only 5% of students perceived loops to be difficult with 95% of those surveyed perceiving them to be either not difficult or easy, see Figure 4.

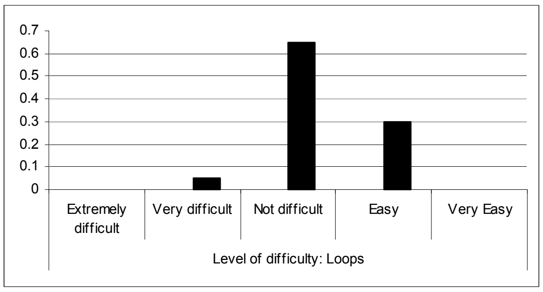

Figure 4: Perceived Level of difficulty: Loops (ITT Dublin)

When surveyed about the perceived level of difficulty of arrays 60% of those surveyed found arrays to be either extremely or very difficult with only 40% indicating no difficulty with the concept, see Figure 5.

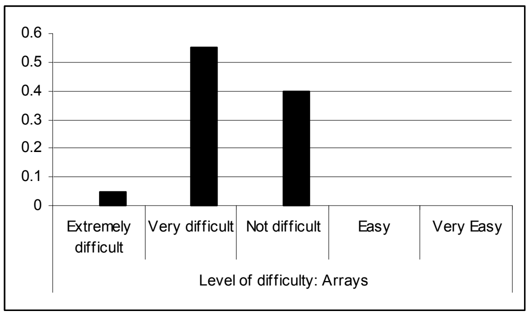

Figure 5: Perceived Level of difficulty: Arrays (ITT Dublin)

The students surveyed indicated no difficulty with the selection construct with 40% indicating that selection was not difficult and the remaining 60% perceiving it to be either easy (45%) or very easy (15%). In triangulating the results from the examination script analysis and the student questionnaires, bearing in mind that they pertain to different student samples, the main area of weakness from the script analysis, i.e. arrays, concurs with the students' perceptions of level of difficulty with 60% of students perceiving arrays as being extremely or very difficult. In terms of the students' perceptions, they rank looping next in difficulty, with 5% indicating difficulty, finally with selection, no one perceived this construct as difficult. These results concur with the results of the script analysis in their ranking of difficult programming topics, as follows:

- Arrays
- Looping
- Selection

2.2.2 *Data collection, analysis and evaluation – semester 2*

In semester 2 the course at ITT follows an object-oriented paradigm and the topics covered are object-oriented topics. The analysis of the examination scripts for Semester 2 2003 indicated that the main areas of difficulties were questions based on the following topics, ranked in order of difficulty:

- Polymorphism
- Getter/Setter Methods
- Object Creation and Method Calling
- Subclass Object Construction

The analysis of the examination scripts for Semester 2 2004 indicated that the main areas of difficulties were questions based on the following topics, ranked in order of difficulty:

- Polymorphism
- Methods
- Code Comprehension
- Object Creation and Method Calling

An overview of these results is shown in Figure 6 below, the higher values indicating higher levels of difficulty.

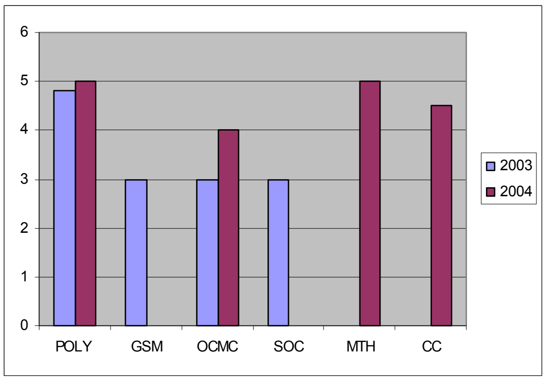

Figure 6: 2003 and 2004 Semester 2 Analysis of Examination Results (ITT Dublin) indicating main areas of difficulties based on student performance.

In semester two the main areas of difficulty at ITT over the period studied, in terms of programming constructs were

- Methods
- Polymorphism
- Objects

These results and areas of weaknesses are specific to ITT Dublin, given the sample of students' examination scripts analysed over the time period outlined.

Twenty-second year students at ITT Dublin completed questionnaires in order to determine their perceptions relating to topics covered in semester two. These students were not, as stated above, the same sample on which the script analysis was based. In relation to the students' perception of the level of difficulty of methods and parameter passing, only 20% of the students perceived these concepts to be either extremely or very difficult with the remaining 80% perceiving them to be not difficult or easy, see Figure 7.

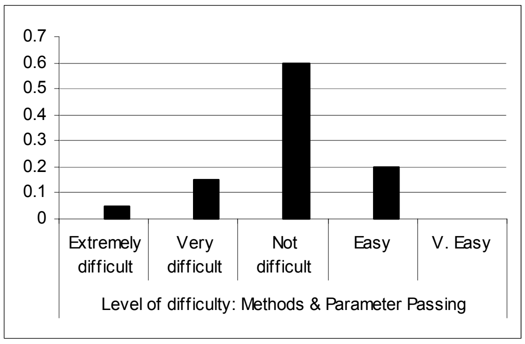

Figure 7: Perceived Level of difficulty: Methods and parameter passing (ITT Dublin)

However when surveyed regarding the perceived difficulty of polymorphism, 55% of the students surveyed indicated that they found it extremely or very difficult, see Figure 8.

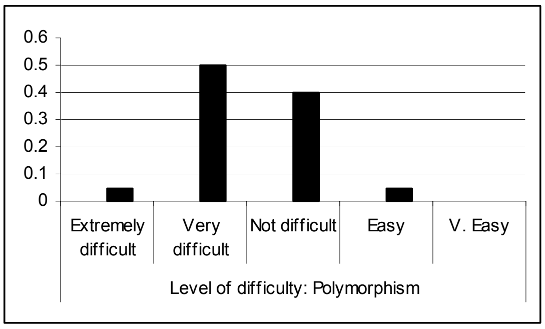

Figure 8: Perceived Level of difficulty: Polymorphism (ITT Dublin)

In relation to subclass object creation 30% of those surveyed perceived this topic to be either extremely (5%) or very (25%) difficult. In triangulating the results from the script analysis for semester two and the second year student survey results, the following observations were made:

In semester two, the main areas of difficulty at ITT determined from the scripts analysis, were:

- Methods
- Polymorphism
- Objects

From the survey the students perceived polymorphism to be the most difficult topic with 55% of students perceiving difficulty. The students then perceived Inheritance to be next in difficulty with 35% of those surveyed indicating that they found the topic extremely or very difficult. Object construction and Subclass object creation both had a 30% perceived difficulty with Methods and parameter-passing producing a 20% difficulty, see Figure 7. The findings of both data analyses match in identifying the areas of weaknesses but the actual and perceived level of difficulty are different. The students perceived methods to be the least difficult but in the analysis of the scripts this topic was determined as one of the main area of weakness.

A sample of the data, where Leaving Certificate Points, (results obtained from a State examination sat by all students at the end of second-level education), and other subject results were available, from ITT was input to a statistical analysis package, MINITAB™ for further analysis. A regression analysis carried out on students' Leaving Certificate points and their final result in the Software Development examination indicated that there was a linear relationship between the two, p = .001. However, given the sample that was analysed, the relationship, Rsq^{2,} 15%, was not very strong, this area requires further research. A more significant relationship existed between a student's final result in Software Development and the number of fails the student had in other modules. A significant p value of .029 was returned; indicating that a student's result in Software Development is a useful result in predicting that they may have fails in other subjects. One can assert that the lower the Software Development result falls, the student is more likely to have more failures in other subjects.

2.3 Focus group discussions

As mentioned earlier the focus group discussion was the tool used for collecting qualitative data from a sample of the user group. These focus groups were facilitated by a moderator and again were done with a random group of 1st and 2nd year students. As larger focus groups can inhibit the participation by some members (Sherraden, cited in Shapiro and Wolff, 2001), the size of both focus groups was kept fairly small at 6-15 people. An interview guide as shown in the first column of Table 6 below was prepared in order to help structure the discussion. The first focus group consisted of 1st year students who were asked some general questions about their course using the above guide. The responses from the group were collated and are presented in the 2nd column of the table below:

Table 6 Focus Group Guide and Responses

Interview Guide	**1st Focus Group (1st Year)**	**2nd Focus Group (2nd Year)**
Do you find Software Development the most difficult module in your programme?	Over 70% agreed	43% agreed
Do you think that the ability to problem solve has a	Over 50% agreed	100% agreed

Interview Guide	**1st Focus Group (1st Year)**	**2nd Focus Group (2nd Year)**
major impact on your performance in Software Development?		
Do you think it is necessary to be good at Maths to perform well in Software Development?	Most disagreed	Most disagreed
Do you think that your performance in Software Development is linked to the result you got in Leaving Cert Maths?	Most disagreed	Most disagreed
When solving a Software Development problem, do you spend time away from the computer designing your solution?	50% agreed that they did not	70% agreed that they did not but may sketch an outline solution
What do you regard as the most difficult concepts in Software Development?	Looping and arrays from semester 1 Polymorphism from semester 2	Methods and general environment set-up from semester 1 Threads from semester 2
Which format do you find the most useful in learning Software Development? a) Lectures? b) Tutorials? c) Labs?	Labs	Labs
If given the opportunity, would you choose to study a computing course, which had no Software Development module?	Over 30% said yes	100% said no

The authors examined the set of responses to the focus group discussion with the 1st year group more closely in order to demonstrate whether they validate even further the earlier quantitative analysis conducted. When the relevant aspects of the focus group were compared with the earlier results of the quantitative analysis, there is significant agreement between both sets of results. For example, over 50% of the 1st year group agreed that the ability to problem solve has a major impact on Software Development performance. The survey conducted with a larger sample group showed 75% agreeing with this statement also. In the focus group over 50% agreed that they did not spend time away from the computer designing a solution while the survey showed 80% as sometimes, rarely or never designing before implementation. With regard to analysis of the examination scripts, arrays, looping and polymorphism emerged as the topics with the highest failure rates.

These three topics were also pointed to in the focus group discussions as areas that cause students the most difficulty from a conceptual perspective. It is interesting to note that on certain issues there was a substantial difference in opinion between the two student groups, which was reflected in their responses to certain questions in the focus group discussions. While 70% of the 1st year group registered that they found software development to be the most difficult module in their programme, only 43% of the 2nd year group felt this to be the case. There may be a number of reasons for this difference in opinion. Interestingly, students reported a greater level of comfort with the software development in 2nd year as opposed to how they felt about the module in 1st year. It could suggest that students in 1st year are still in the process of becoming used to programming which is a totally new subject that they would generally not have encountered before entering third-level.

Another point of note is that over 30% of the 1st year group would choose to study a programme, which contained no software development module, if given the chance. However, not one member of the 2nd year group would choose this type of programme. This seems to suggest that a significant number of 1st year students experience major doubts regarding their choice of programme. In contrast, students in the 2nd year group expressed a greater sense of enjoyment and satisfaction with software development and explained, “it is the ability to write bigger programs that actually do something”, “the project in 2nd year gives us lots of practice at writing Java”, “…. more time spent on Java the better”.

Interestingly, with regard to the relationship between performance in Mathematics and the student’s ability in software development, neither group felt there was any link. Both groups also agreed that the most useful learning environment for them is the laboratories where they can spend time on solving practical exercises, worksheets etc, with the help of laboratory facilitators. This concurs with the constructivist approach in engaging the learner. The learning tool which was developed will be used online and in the laboratories. After conducting the focus group discussions, one overall observation from the authors’ viewpoint is the increasing amount of guidance and support needed to help 1st year students overcome the initial hurdles and convince them of the more enjoyable and rewarding aspects of software development as described by the 2nd year students.

3. Conclusion

The methods used for data gathering have been outlined and these include both quantitative, the scripts collation and the statistical analysis of same, and qualitative approaches such as the analysis of questionnaires and focus group discussion. A triangulation of both quantitative and qualitative measures was used to gather data. The measures used focused on gathering data that would be analysed in light of the research objectives. These objectives and their relationships to measures used are as outlined in Table 7:

Table 7: Relationship between research objectives and data gathering measures

Research Objective	Corresponding Data Gathering Measure
Determine students' main areas of weaknesses in semester 1 and semester 2 of software development	Analysis of students' past examination scripts.
To gain insight into the students' perceptions of their approach to problem solving and design	Survey responses/ Focus group discussion
Is Software Development the only area of difficulty in the undergraduate course	Statistical analysis of students' overall performance.

The data gathered pertaining to semester 1 topics in terms of the ranking of the areas of weaknesses across participating academic institutions, as determined by the user needs analysis, is outlined below in Table 8.

Table 8: Ranking of the different areas of weaknesses in participating institutions

Areas of Weakness Semester 1	ITT	ITB	DCU
Arrays	1		3
Looping	2	2	
Selection	3	3	
Methods		1	2
Problem-solving			1

The data gathered pertaining to semester 2 topics in terms of the ranking of the areas of weaknesses, as determined by the user needs analysis, is outlined below in Table 9.

Table 9: Ranking of the different areas of weaknesses in semester 2 in the participating institutions

Areas of Weakness Semester 2	ITT	ITB	DCU
Methods	1		1
Polymorphism	2		
Subclass object creation	3		
Arrays		1	3
GUI		2	
Object construction		3	2
Object creation			1

Having determined the main areas of weaknesses in the participating institutions, it was agreed to base the project's Learning Objects on some of the following areas:

- Arrays
- Looping
- Selection
- Methods
- Objects

The development of these Learning Objects is dependent on the time frame and scope of the project, and as such would be limited by those constraints. An initial prototype based on arrays was developed for review and discussion. An important feature of the project is to design and develop a meta-cognitive interface to develop the necessary reflective and self-analysing skills in novice programmers. As part of the meta-cognitive interface learners outline an initial approach to a concept/problem solution, reflect on what they knew initially and what they have learned and then review the approach initially adopted. By requiring the learner to articulate his/her approach, implicit knowledge becomes explicit, making the learning process more effective. The students' approaches and reflections will be stored and these metacognitive traces will be captured to provide the lecturer/tutor with a valuable insight into the approaches adopted by the students and any particular areas of weakness.

This tool is scheduled to be rolled out from Febuary 2007. An in-depth, cross-institutional evaluation of the educational impact of the RLOs with the supporting metacognitive interface, within the proposed learning environment, will be conducted. The evaluation will include an observational case study, quasi-experiment, student questionnaire and focus groups. An evaluation of a discussion forum will also be conducted as part of the evaluation phase. The data stored from the metacognitive interface will be retrieved and analysed. The data from the different evaluation methods will be triangulated in order to draw conclusions. The findings of the project will be disseminated to the wider educational community. By wrapping a meta-cognitive interface around learning objects which target the novice programmers' main areas of weaknesses, it is hoped to provide a comprehensive and innovative learning tool which addresses conceptual difficulties with programming constructs and at the same time promotes the learners' meta-cognitive skills which are essential for learning in this domain.

References

Blakey E., Spence S., 1990, "Developing Metacognition". ERIC Digest. ED327218.

Cohen, L., and Manion, L., 1997, "Research Methods in Education". London.

Fekete, A., Kay, J., Kingston, J., Wimalaratne, K., 2000, "Supporting reflection in Introductory Computer Science", ACM SIGCCSE Bulletin, Proceedings of the 31st SIGCSE technical symposium on Computer Science Education.

Gage N.L., Berliner, D.C.,1980, "Educational Psychology", Houghton Miffin, Boston.

Guzdial, M., 1994, "Software-realised scaffolding to facilitate programming for science learning." Interactive Learning Environments,4(1) 1-44.

Henderson, Peter B. 1986, Anatomy of an introductory Computer Science Course", Proceedings of the Seventeenth SIGCSE technical symposium on Computer Science Education, February 1986,257-263.

Linn Marcia C., Clancy Michael J, 1992,"The case for case studies of programming problems." Communications of the ACM, March 1992 v35 n3 p121(12).

Riley, D, 1981, "Teaching problem solving in an introductory computer science class", 12th SIGCSE Technical Symposium on Computer Science Education.

Shapiro, T. and Wolff, E. (eds.), 2001, Asset Building Policy and Programs for the Poor, Assets for the Poor: The Benefits of Spreading Asset Ownership, Russell Sage Foundation, New York.