Evaluation and proposed enhancements in learning undergraduate computer programming using *visual* Problem Based Learning (PbBL) *visual* Project Based Learning (PjBL) and *visual* computer programming.

Submitted for the Degree of Doctor of Philosophy

At the University of Northampton

2018

Hill, Gary J.

DEDICATION

TO MY FATHER

John Hill

For his inspiration, sense of adventure and approach to lifelong learning. May he rest in peace.

TO MY MOTHER

Jeanne Margaret Hill

For her continued support and belief in me.

TO MY WIFE

Carrie Jane Carruthers

For her encouragement, support and understanding with everything in life.

TO MY SON

Harrison John Hill

For his unknowing support and escapism.

DECLARATION

I hereby declare that the work described in this thesis is original work undertaken by me for the degree of Doctor of Philosophy, at the Faculty of Education and Humanities – University of Northampton, United Kingdom. No part of the material described in this thesis has been submitted for any award of any other degree or qualification in this or any other University or College of advanced education*.

ACKNOWLEDGMENTS

I will be ever thankful to Dr Cristina Devecchi and Dr Scott Turner for their encouragement and invaluable support throughout this process.

ABSTRACT

This research, as published in the papers being critically appraised here, evaluated existing teaching strategies adopted on computing degree programmes and proposed pedagogical innovation that impacts and improves first year undergraduate computer programming and, enhances student learning. This necessitated a departure from the didactic teaching strategy often adopted (of teaching code via isolated concepts and working through a few exercises) by introducing an improved and enhanced pedagogical approach.

The focus of the published work was the teaching of computer programming and problem solving to undergraduate first year computing students, using visual computer programming, together with robot's/robot simulators. The author is responsible for the original idea, design, development and introduction of a new first year undergraduate module at the University of Northampton in 2004. The new first year module called "CSY1020 Problem Solving and Programming" enabled proposed pedagogical approaches to be implemented and evaluated. The overarching new and innovative pedagogical approach introduced and implemented by the author, that improved and enhanced student learning was called 'Problems-first' and 'Graphics-first'. The author has used the term 'Graphics-first' to define the approach, whereby visual computer programming and graphical user interfaces (GUI) are developed from the outset and throughout the module.

The critical appraisal details the extent to which the author's published works provide a coherent demonstration of the new pedagogical innovation which focused on and emphasised the importance of the 'Problems-first' and 'Graphics-first' learning and teaching strategies, that were subdivided into **four** core pedagogical approaches:

- Problem Solving (PS).
- Problem-Solving-First (PSF) /Problems-First (PF).
- Problem Based Learning (PbBL)/Project Based Learning (PjBL).
- Physical to Visual/Visual Programming.

These **four** core pedagogical approaches have been applied by the author to the teaching of computer programming and problem solving to undergraduate first year computing students, using robots/robot simulators and visual programming to emulate the robot tasks. Students

were taught how to solve problems as part of a project/assignment that is related to a pseudo real-world problem which ultimately stimulates deeper learning and its transferability.

The author has been the first to define and differentiate between Problem Based Learning and Project Based Learning by using the abbreviations PbBL and PjBL and defining PjBL as where a project/challenge is set from the outset, such that one PbBL activity leads to another and the series of linked problems form the greater challenge or project. The author's innovative approach can be seen through adhering strictly to the above definition, throughout the delivery of the Programming module, which combined visual PbBL and visual PjBL.

The need to focus initial computer programming education on problem solving, prior to the teaching of computer programming syntax and software design methodology, is proposed by the author. The main vehicle for this approach is a robot/robot simulation programmed in the Java programming language, followed by the visual computer programming of a graphical representation/simulation to develop computer programming skills all delivered by the four pedagogical approaches.

The methodology used was action/practitioner research, where the proposed learning and teaching strategies were designed, developed, implemented, evaluated and reflected upon by the author to enable refined approaches to be further re-implemented. The findings and recommendations of this research are that the author's contribution and impact evidences to include the following learning and teaching strategies:

- Problem Solving (PS) throughout the teaching of computer programming (and allied subjects);
- Visual Problem Based Learning (PbBL) visual Project Based Learning (PjBL) and;
- Physical to visual/visual programming (VP).

The academic impact of the author's research shows that interest has been generated through its dissemination, resulting in citations and requests for papers, together with the impact of the citations, within the context of the pedagogical approaches discussed. This research led to impact and influence on many stakeholders whether: students (at all academic levels); lecturers within or outside the discipline; other Universities throughout the UK, EU and internationally; UK school education (primary and secondary) and even a commercial robot manufacturer.

The author's research is also shown to be timely, relevant and impactful following the introduction of the Teaching Excellence Framework (TEF) in 2016, due to its proposed pedagogical innovation. Also, with the introduction of the new Computing National Curriculum in September 2013, not only has computer programming become prominent within the School's curriculum, but both the introduction of computational thinking and the use of visual programming languages are directly relevant to the pedagogical approaches advocated by the author i.e. Problem Solving (PS) visual Problem Based Learning (PbBL) visual Project Based Learning (PjBL) and physical to visual/visual programming (VP).

KEY TERMS

Active blended learning, active learning; computational thinking; computer programming, experiential learning; graphical; graphical programming; graphical user interface; graphics-first; Problem Based Learning; Project Based Learning; problems-first; problem solving; robots; simulation; visual; visual programming, visual programming language.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

GLOSSARY

Term	Definition/Explanation	Section
Active Blended Learning (ABL)	"Student-centred activities that support the development of subject knowledge and understanding, independent learning and digital fluency. …..face-to-face teaching is facilitated in a practical and collaborative manner, clearly linked to learning activity outside the classroom. Opportunities are provided for students to develop autonomy, Changemaker attributes and employability skills" (UoN, 2017).	cf.6.3
Computational Thinking (CT)	Reformulating a seemingly difficult problem into one we know how to solve.	cf.2.1.4
CSY1020	A level 4 (first year) undergraduate module called Problem Solving and Programming module, taught at the University of Northampton since 2004.	
Experiential Learning	Learning by doing to emphasise and reinforce the relationship to other experiences and learning contexts, ultimately ensuring that the transfer of learning/experience to similar and dissimilar learning contexts.	cf. 2.4, 2.6.2
Graphical Programming	Sometimes considered synonymous with VP, but can also mean the drawing of graphical 2-Dimensional/3-Dimensional objects/shapes.	
Graphical User Interface (GUI)	A computer application that is a standalone application in its own window/frame, possibly with a menu, e.g. Microsoft Word.	cf.4.3.3
Java	A computer programming language publicly released in 1995.	
Problem Based Learning (PbBL)	A pedagogical approach that enables learning to take place by applying problem solving skills to a given activity	cf.2.1.2
Problem solving (PS)	The acquisition of problem-solving skills that are then used to solve Problem Based Learning (PbBL) activities.	cf.2.1.1
Problem Solving Learning (PSL)	The practice of solving problems to aid the learning.	cf.2.1.1

Term	Definition/Explanation	Section
Project Based Learning (PjBL)	Where a project/challenge is set from the outset, such that one PbBL activity leads to another and the series of linked problems form the greater challenge or project.	cf.2.1.3
Robot	The term robot has been used throughout this research as a generic term, particularly where the emulated/graphical representation of robots/objects have taken many incarnations e.g. robot, bomb disposal robot, a baby, car etc.	cf.1.1
Visual Programming (VP)	Everything you see in a computer application, not only the Graphical User Interface (GUI) but also all the other visual components and images contained within it, e.g. a standalone computer game application.	cf.4.5.3
Visual Programming Language (VPL)	A computer programming language that allows the development of computer programs by moving and arranging visual objects to form the logic and structure of the computer program.	cf.4.5.3

ACRONYMS AND ABBREVIATIONS

ABL	Active Blended Learning
AC	Abstract Conceptualization
ACM	Association for Computing Machinery
AE	Active Experimentation
BIE	Buck Institute for Education
CAS	Computing at Schools
CC	Code Club
CE	Concrete Experience
CSY1020	Problem solving and programming module
ESRC	Economic and Social Research Council
EU	European Union
FHEQ	Framework for Higher Education Qualifications
GUI	Graphical User Interface
HE	Higher Education
HEA-ICS	Higher Education Academy-Information and Computer Science
HEFCE	Higher Education Funding Council for England
IEEE	Institute of Electrical and Electronics Engineers
JICC	Java and the Internet in the Computing Curriculum
LMS	Lego Mindstorms Simulator
MIT	Massachusetts Institute of Technology
MRS	Microsoft Robotics Studio
NCC	Northamptonshire County Council
PbBL	Problem Based Learning
PBL	Problem Based Learning/Project Based Learning
PF	Problems-First
PGCE	Post Graduate Certificate
PjBL	Project Based Learning
PS	Problem Solving
PSF	Problem-Solving-First
PSL	Problem Solving Learning
QAA	Quality Assurance Agency
RCUK	Research Councils UK
REF	Research Excellence Framework

RO	Reflective Observation
TEF	Teaching Excellence Framework
UK	United Kingdom
UoN	University of Northampton
URB@N	Undergraduate Research Bursaries at Northampton
VARK	Visual, Aural/Auditory, Read, Kinaesthetic
VPL	Visual Programming Language
WCIT	Worshipful Company of Information Technologists

CHAPTER 1 INTRODUCTION.

1.1 Introduction: Critical Appraisal.

This critical appraisal explicitly demonstrates the contribution to knowledge evidenced through the pedagogical innovation and impact on student learning the author's published work has achieved. The focus of the published work is the teaching of computer programming and problem solving to undergraduate first year computing students, using visual computer programming (VP) together with robot's/robot simulators. In addition, all the published work relates to a module, CSY1020 problem solving and programming (cf. 2.6.1, 2.6.2) devised and introduced by the author at the University of Northampton (UoN) in 2004. The need to focus initial computer programming education on problem solving, prior to the teaching of computer programming syntax and software design methodology is also considered. The main vehicle for this approach was a robot/robot simulation programmed in Java (computer programming language) followed by the computer programming of a visual representation/simulation to develop computer programming skills. The author introduced a Graphics-first approach to computer programming from the outset, such that the visual computer programming (VP) and graphical user interfaces (GUIs) were developed throughout the module. The term robot has been used throughout this research as a generic term, particularly where the emulated/graphical representation of robots/objects have taken many incarnations e.g. robot, bomb disposal robot, a baby, car etc.

This critical appraisal details the extent to which the author's published works provide a coherent demonstration of the following:

> "1. The creation and interpretation of new knowledge, through original research or other advanced scholarship, of a quality to satisfy peer review, extend the forefront of the discipline, and merit publication*,
>
> 2. A systematic acquisition and understanding of a substantial body of knowledge which is at the forefront of an academic discipline or area of professional practice" (The University of Northampton, 2015, p6).

[* All the author's work considered here has been published].

Items 1) and 2) (above) are demonstrated and articulated in the following chapters (Figure 1.1):

- Introduction (Chapter 1).
- Critique of the literature, updated to include current literature developments and the definition of key terms (Chapter 2).
- Research methods/methodology (Chapter 3).
- An exploration of the interrelationships between publications to produce an overarching synopsis of the research as one study (Chapter 4).
- Impact and reception of the work (Chapter 5).
- Conclusions and recommendations for the future (Chapter 6).
- References and appendices (Figure 1.2).

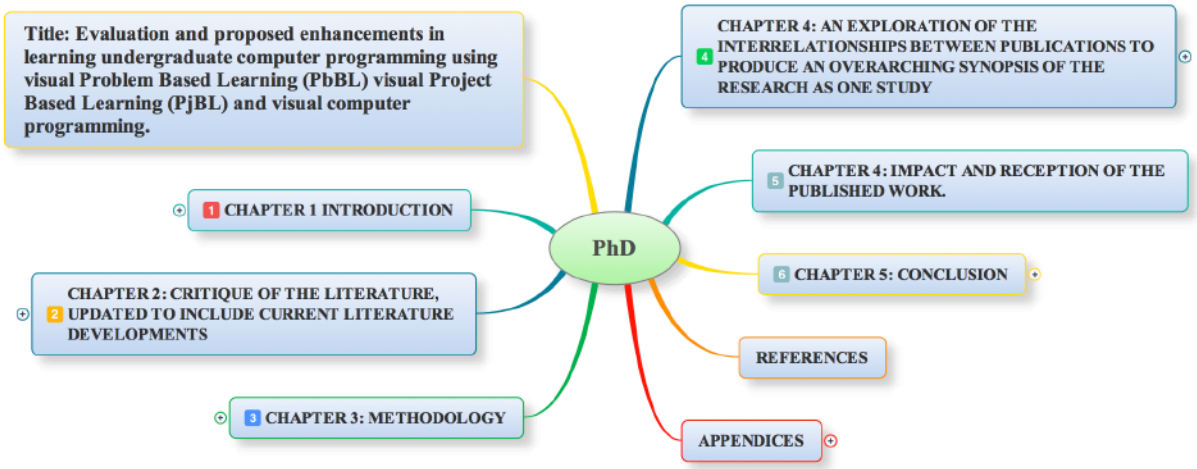


Figure 1.1: Critical Appraisal Structure.

Useful information for this critical appraisal is presented within the appendices (Figure 1.2). The first two appendices (Appendix 1, pp 102-103 and 2, p 104) include a summary, in the form of the abstracts, of the key published works (Appendix 1) to be considered as the body of research critically appraised here and any supplementary publications (Appendix 2, p104). written by the author. Appendix 3 (pp 105-106) contains the documentation required by the University for ethical considerations for engaging with participants for interviews and questionnaires. Appendix 4 (pp 107) contains a statement on the review processes of the journals (or equivalent) in which the works have been published, with Appendix 5 (pp 108-109) including statements from co-authors on the extent of the applicant's contribution to the

key published research papers. Latterly, Appendices 6 (pp 108-195) and 7(pp 196-232) give the 'full text' versions of the published papers from Appendix 1 and 2 respectively. Finally, Appendix 8 (pp 233-236) contains the module specification/guide for the CSY1020 problem solving and programming module devised and introduced by the author (cf. 2.6.1, 2.6.2).

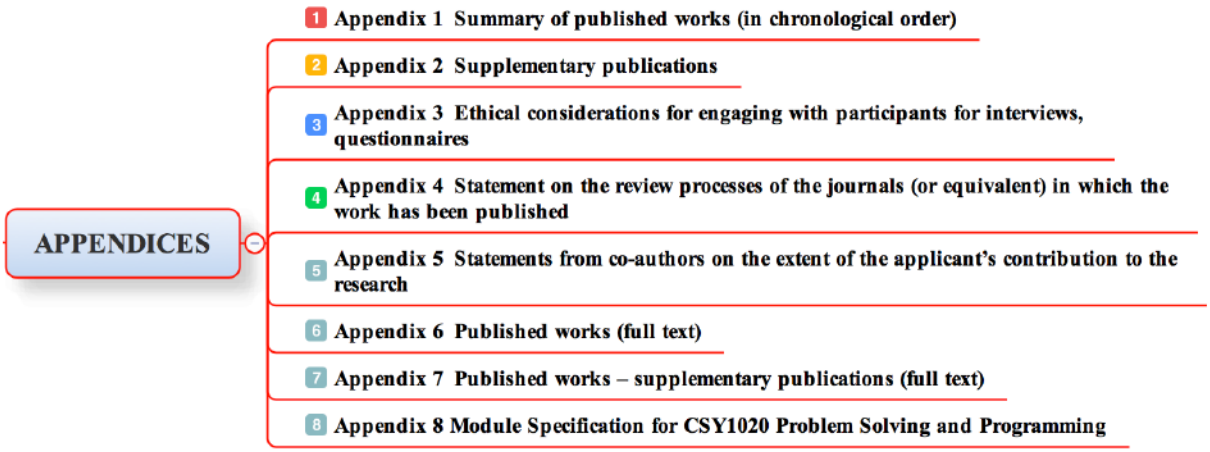


Figure 1.2: Appendices Structure.

The author has, since 1999, taught graphical computer programming to undergraduate and postgraduate students at the University of Northampton and was convinced of the benefits of visual computer programming from the outset and throughout. The author, through this research, professional expertise and practice set out to explore this conviction. The author has used the term 'Graphics-first' to define the approach, whereby visual computer programming and graphical user interfaces (GUI) are developed from the outset and throughout the module. The subject/topic is the teaching of computer programming and problem solving to undergraduate first year computing students, using visual computer programming (Graphics-first) Problem Based Learning (PbBL) Project Based Learning (PjBL) together with robot's/robot simulators.

Traditionally, computer programming had often been taught by adopting a didactic teaching strategy (of teaching a computer programming language via the introduction of isolated concepts, then working through a few exercises by writing computer code) (cf. 1.3). The overarching aim of the author as evidenced within this critical appraisal, was to evaluate and propose improvements to enhance student learning using a *visual* Problem Based Learning (PbBL) and *visual* Project Based Learning (PjBL) pedagogical approach, demonstrated and reinforced via visual programming. Initially the author considered and evaluated the

importance of problem solving first before computer programming but this evolved into what the author has called 'Problems first, second and third' (Hill and Turner, 2014a) to reflect the introduction of problems-first (problem solving) and then the progression onto PbBL and PjBL. Together with the idea of Problems first, second and third, PbBL (cf. 2.1.2) and PjBL (cf. 2.1.3) the importance of computer programming always requiring problem solving skills (cf. 2.1.1) and computational thinking (cf. 2.1.4) are also explained.

In addition, the author's (cf. 1.2) earlier research (since 2001) on a problems-first approach to computer programming (e.g. Hill and Turner, 2011; 2014a) emphasises the importance of problem solving, Problem Based Learning (PbBL)/Project Based Learning (PjBL) and the benefits of both physical and visual solutions.

1.2 Contribution to the Research.

The published works that form the substance of this critical appraisal include 10 key papers (Table 1.1) and 7 supplementary papers (Table 1.2). Table 1.3 illustrates the author's contribution, whilst Appendix 5 includes statements from co-authors on the extent of the author's contribution to the key (and supplementary) published research papers.

From the 10 key papers the author is sole author of 2, lead author of 2 and the second co-author of 4 and a co-author of three for the two remaining papers. In all co-authored cases the authors' contribution is equal. One co-author (Turner) appears on a number (8) of the published works. The distinction between the work of the two authors (Hill and Turner) is clear. Whilst, the work is closely related and integrated, the separation area occurs with 'physical' and 'simulated' Robots (Turner) against the 'visual' / graphical emulation of the robot activities via computer programming PbBL and PjBL (Hill).

In all the co-authored papers, there is an admission that all the authors took an equal responsibility for the writing of the papers/publications. However, the concept of the CSY1020 Problem Solving & Programming module and the definition and implementation of PbBL, PjBL with the emphasis on visual PbBL, visual PjBL, Visual Programming (VP) and Graphical User Interfaces (GUI) were the sole contribution to new knowledge by the author.

1	Turner, S., Hill, G. J. (2006) ***The Inclusion of Robots Within The Teaching OF Problem Solving: Preliminary Results***, 7th Annual Conference of the ICS HE Academy, Trinity College, Dublin, 29th - 31st August 2006, Proceedings pg 241-242 ISBN 0-9552005-3-9 (Poster).
2	Turner, S., **Hill, G. J.** (2007) *Robots in Problem-Solving and Programming,* 8th Annual Conference of the Subject Centre for Information and Computer Sciences, University of Southampton, 28th - 30th August 2007, pp 82-85 ISBN 0-978-0-9552005-7-1
3	Turner, S., **Hill, G. J**. (2008) *Robots within the teaching of Problem-Solving*, ITALICS, HEA-ICS, Volume 7 Issue 1, June 2008, pp. 108-119, ISSN: 1473-7507.
4	Turner, S., **Hill, G.,** Adams, J. (2009*) Problem Solving and Creativity for Undergraduate Computing and Engineering students***,** 9th 1-day Teaching of Programming Workshop, University of Bath, 6th April 2009.
5	Turner, S., **Hill, G. J.** (2010) *Innovative Use of Robots and Graphical Programming in Software Education*, Computer Education, Volume 9, May 2010, pp. 54-6, ISSN: 1672-5913.
6	**Hill, G. J.**, Turner, S. (2011) *Chapter 7: Problems First, Software Industry-Oriented* Education Practices and Curriculum Development: Experiences and Lessons, M Hussey, X Xu and B Wu (Eds.) IGI Global, USA, pp 110-126, ISBN: 978-1-60960-797-5.
7	Kariyawasam, K., A., Turner, S., **Hill, G**. (2012) *Is it Visual? The importance of a Problem Solving Module within a Computing course*, Computer Education, Volume 10, Issue 166, May 2012, pp. 5-7, ISSN: 1672-5913.
8	**Hill, G. J.,** Turner, S. (2014a) *Problems First, Second and Third*, International Journal of Quality Assurance in Engineering and Technology Education (IJQAETE) IGI Global, USA, Volume 3 Issue 3, pp. 88-109, DOI: 10.4018/ijqaete.2014070104, ISSN: 2155-496X, EISSN: 2155-4978.
9	**Hill, G. J.** (2015) *Review of a problems-first approach to first year undergraduate programming,* 11th China – Europe International Symposium on Software Industry Orientated Education: 29-30th April 2015, Zwickau, Germany.
10	**Hill, G. J.** (2016) *Review of a problems-first approach to first year undergraduate programming*, Software Engineering Education Going Agile: 11th China–Europe International Symposium on Software Engineering Education (CEISEE 2015): Kassel S., Wu B (Eds.) Springer, pp. 73-80, ISBN 978-3-319-29165-9.

Table 1.1: Selected Published Works (In chronological order).

12	Zhao, F., Turner, S., **Hill, G.**, Dravid, R., Zhang, Y. (2010) "*A Virtual Environment Training System for Haptic Laparoscopic Surgery*", 16th International Conference on Automation and Computing, University of Birmingham, Birmingham, UK, 11 September 2010.
13	**Hill, G.**, Svennevik, E., Turner, S. (2011) *"Green Computer Science Courses! We're going mobile!"*, 7th China - Europe International Symposium on Software Industry Oriented Education, 23-24 May 2011, University of Northampton, Northampton, UK. .
14	**Hill, G**., Turner, S. (2012) "*Referencing within Code in Software Engineering Education!*", Computer Education, Volume 10, Issue 166, May 2012, pp. 1-4, ISSN: 1672-5913.
15	**Hill, G.**, Turner S. (2014b) *"Chapter 5: Electronic Online Marking of Software Assignments"*, Progress in IS: Software Engineering Education for a Global E-service Economy, Motta, Gianmario; Bing, Wu (Eds.) Springer, ISBN 978-3-319-04216-9, DOI 10.1007/978-3-319-04217-6_5, pp 41-48.
16	**Hill, G.**, Svennevik, E., Turner, S. (2014) "*Computer Science Courses Using Laptops*", Innovation in Teaching and Learning in Information and Computer Science, ITALICS, HEA-ICS, Volume 13 Issue 1, June 2014, pp 1-7, DOI: 10.11120/ital.2014.00011.

Table 1.2: Supplementary Published Works (In chronological order).

Publication Number	Author/s	Nature and extent of candidate's contribution
1, 2, 3, 5	ST/**GH**	ST lead author, but equal contribution.
4	ST/**GH**/JA	ST lead author, but equal contribution.
6, 8, 14, 15	**GH**/ST	GH lead author, but equal contribution
7	KK/ST/**GH**	KK lead author, but equal contribution
11	AM/ST/**GH**	AM lead author, but equal contribution
13, 16	**GH**/ES/ST	GH lead author, but equal contribution
9, 10	**GH**	GH sole author
12	FZ/ST/**GH**/RD/YZ	Minor contribution, but secured Leverhulme Trust funding.

Table 1.3: Author's published works and contribution (See also Appendix 5).

Figure 1.3 indicates the keyword density from the published work (using VOSviewer (2016)) indicating the importance of not only robots, but other associated topics of student experience, problem solving, java, emulation, software education, problems first, graphical programming, visual and programming. Figure 1.3 helps with demonstrating the separation from the physical robots and the author's contribution.

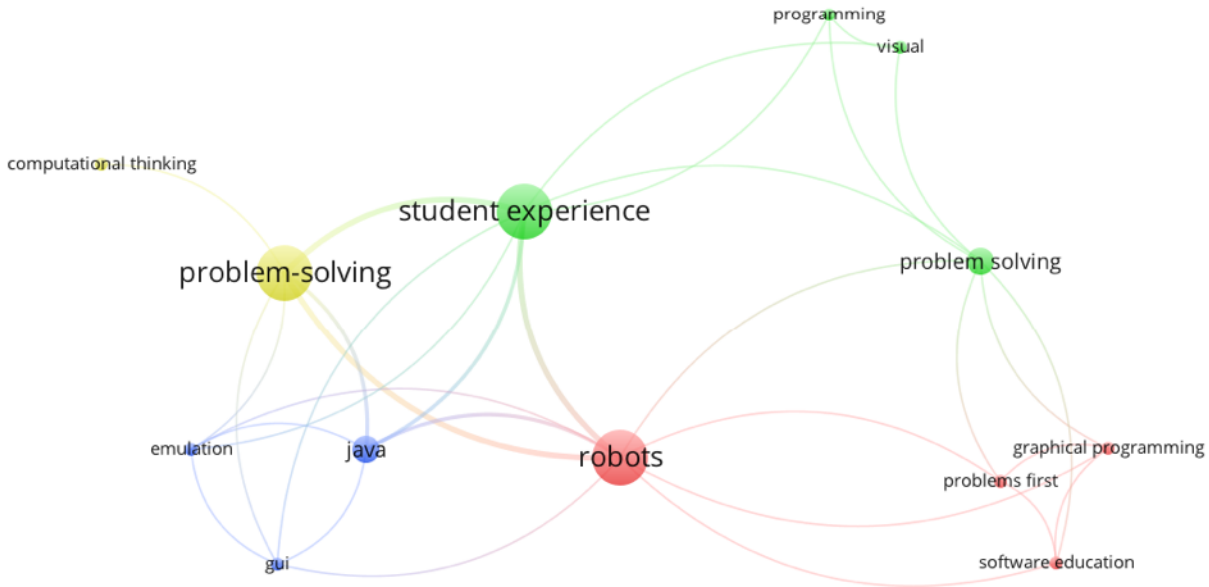


Figure 1.3: Bibliographic Research (Limited Keyword Density) taken from NECTAR (2016) using VOSviewer (2016).

The author has, since 2004, lectured two-thirds of the module CSY1020 Problem Solving and Programming (Appendix 8) at the University of Northampton, which is the subject of all 10 key papers, where the author has taught, reflected and disseminated on the module in practice. Turner has lectured the remaining third part of the module. The author is solely responsible for the:

- CSY1020 Problem Solving and Programming module idea, design and development (cf. 2.6.1, 2.6.2, 4.2 and Appendix 8);
- Problem Solving (PS) and Problem Based Learning (PbBL) approach;
- Incorporation of Problem Based Learning (PbBL) to a wider Project Based Learning (PjBL) situation;
- Graphical/visual computer programming (cf. 4.5.3);
- Linking of the Problem Solving assignment directly to the Programming assignment (cf. 2.3 and 2.6.2).

1.3 Aim and Objectives of the Research.

To improve first year undergraduate computer programming, it is important to consider computational thinking as it underpins all the research of the author. Wing (2006; 2008a; 2008b) highlighted the importance of thinking like a computer scientist. By this it is meant that the thinking processes involved in being a computer scientist are more complicated than just being able to program since computational thinking (cf. 2.1.4) is the reformulating of a seemingly difficult **problem** into one we know how to **solve**. This reformulation emphasised the importance of problem solving and led to the author developing and implementing learning and teaching strategies that evolved into **four** core pedagogical approaches:

- Problem Solving (PS) (Section 2.1.1).
- Problem-Solving-First (PSF)/Problems-First (PF) (Section 2.1.1).
- Problem Based Learning (PbBL)/Project Based Learning (PjBL) (Section 2.1.2 and 2.1.3).
- Physical to visual/visual programming (Section 4.5).

The aim of the research, as published in the papers being critically appraised here, was to evaluate existing teaching strategies adopted on computing degree programmes and propose pedagogical innovation that would impact and improve first year undergraduate computer programming and, to enhance student learning. The author also advocated that this would necessitate a departure from the didactic teaching strategy often adopted (of teaching code via isolated concepts and working through a few exercises) with a desire to introduce an improved and enhanced pedagogical strategy.

Therefore, the objectives of this research were to:

- Evaluate and reflect on existing teaching and learning strategies;
- Develop new, improved and innovative teaching and learning strategies;
- Implement new teaching and learning strategies;
- Critically evaluate and review these strategies and;
- Enhance and evolve the teaching and learning strategies based on findings.

The teaching and learning strategies focused on and emphasised the importance of 'Problems-first' and 'Graphics-first' which explicitly related to the introduction and enhancement of the four core pedagogical approaches: Problem Solving (PS) Problem Solving Learning (PSL); Problem-Solving-First (PSF)/Problems-first (PF); Problem Based Learning (PbBL)/Project Based Learning (PjBL) and; Physical to visual/Visual Programming. These are covered in sections2.1.1, 2.1.2, 2.1.3 and 2.1.4, respectively.

CHAPTER 2 CRITIQUE OF THE LITERATURE, UPDATED TO INCLUDE CURRENT LITERATURE DEVELOPMENTS.

2.1 Definition of terms.

The aim of this chapter is to provide a critique of the literature surrounding the author's published research, updated to include current literature developments. The first section introduces the terms and concepts used throughout this research (cf. 2.1) as these have subtle nuances, such that within the sub-sections (cf.2.1.1, 2.1.2, 2.1.3 and 2.1.4) these terms are defined and, in some cases, redefined. The next section (cf.2.2) clarifies the specific published research that has been reviewed within this critical appraisal, before chronologically explaining the interrelationship, evolution and development of the research. Section 2.3 covers the first 11 years (1999-2009) followed by the next 8 years (2010-2017) (cf.2.4) with a summary (cf.2.5) of the proposed improvements to pedagogy resulting from the evolution and evaluation of the teaching strategies implemented. The final section gives the background context to the research (cf.2.6) introducing first (cf.2.6.1) the rationale for a new computer programming module, followed by a discussion of the pedagogy associated with the domain of *Visual* Problem Based Learning (PbBL) and *Visual* Project Based Learning (PbBL).

The aim of this section is to clarify the terms and concepts used in the published works and critical appraisal of these publications. The key terms are defined below, but will be discussed in more detail within the critical review (Figure 2.1):

- Problem Solving (PS) and Problem Solving Learning (PSL) (Section 2.1.1)
- Problem Based Learning (PbBL) (Section 2.1.2)
- Project Based Learning (PjBL) (Section 2.1.3)
- Computational Thinking (CT) (Section 2.1.4)

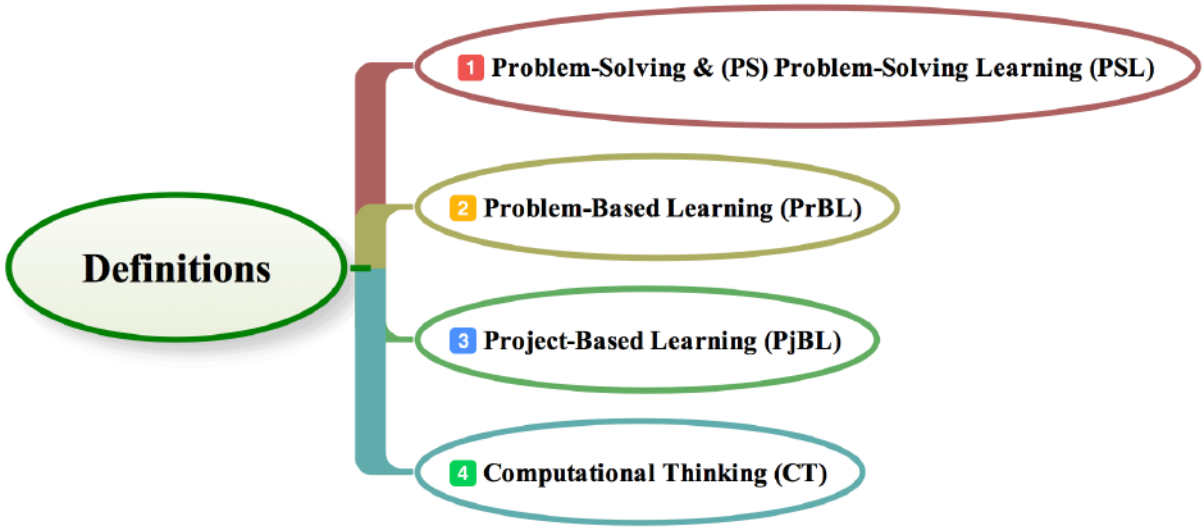


Figure 2.1: Definitions.

2.1.1 Problem Solving (PS) and Problem Solving Learning (PSL).

It is suggested that PSL, is where:

> "the focus …. on learning is largely on acquiring the answers expected by the lecturer, answers that are rooted in the information supplied in some way to the students" (Savin-Baden, 2000, p2).

In this critical appraisal, the term problem solving refers to the acquisition of problem-solving skills that are then used to solve Problem Based Learning (PbBL) activities, and not solely the practice of solving problems to aid the learning (PSL). This may sound a little obscure, but there is a subtle difference between just solving problems as part of the learning process e.g. various mathematical exercises, in comparison with learning how to solve problems that can be applied and transferred to other applications. Later (cf. 4.4) a commonly used computer software development method, called the 'waterfall' method, is given as an example of a structured approach to solving a problem.

2.1.2 Problem Based Learning (PbBL).

The characteristics, which help to define Problem Based Learning are said to be:

> '"Just in time" learning: Learn information/skills/attitudes to "solve a problem"
>
> Problem posed first (before the students have learned anything)
>
> Students empowered with selecting learning goals, resources, assessment.
>
> Work cooperatively in small groups (with or without a tutor present in each group).
>
> Teacher? maintain standards, "Guide on the side not sage on the stage," monitors the process, (like a design project).
>
> Students actively engaged in the learning process; students teach each other'". (Woods, 2006, p1-1)

Unfortunately, the abbreviation of PBL within education confusingly appears to be used interchangeably for both Problem Based Learning (PbBL) and Project Based Learning (PjBL) (Project Based Learning (PjBL)is defined later (cf. 2.1.3). An innovative aspect of this research is clearly differentiating Problem Based Learning (PbBL) from PjBL).

The author feels that the two terms are not synonymous and therefore it is important to differentiate between the two, and the abbreviations PbBL for Problem Based Learning and PjBL to indicate Project Based Learning are used throughout this critical appraisal. Therefore, PbBL can be defined as a pedagogical approach that enables learning to take place by applying problem solving skills to a given activity. In addition, there:

> "remains diversity of opinion about what does and does not count as problem-based learning" (Savin-Baden and Wilkie, 2004, p2).

PbBL is said to have been first developed within medical education in the 1950s (Hung, Jonassen, Liu, 2008, p486). Although, the term 'Problem Based Learning' has been credited to McMasters University, where they have used 'PBL' in their medical education curriculum since the late 1960's (Lee and Kwan, 1997, p149).

2.1.3 Project Based Learning (PjBL).

The term 'Project Based Learning' (PjBL) or 'project pedagogy' as it was originally termed, is said to have originated when:

> "a tradition of project pedagogy in engineering education emerged in Denmark" (de Graaff, Kolmos, 2007, p3)

….in the 1970's to include learning by doing and experiential learning.

Woods (2006, p1-0) suggests that:

> "with the success of"

PBL

> "has come a spin-off of all kinds of variations on the theme".

One such 'spin-off' mentioned is that:

> "Some use problems to synthesize previous knowledge. This is characteristic of engineering design projects and has led to the term 'project based learning'" (Woods, 2006, p1-0).

Whilst, the author would agree with the synthesis of previous knowledge, the author's definition concurs with what Savin-Baden and Wilkie (2004, p2) term the *"evangelist's"* view suggesting that for true PbBL to occur, you must have an:

> "integrated curriculum where one problem builds upon another…" (Savin-Baden and Wilkie, 2004, p2).

Therefore, the author defines PjBL as where a project/challenge is set from the outset, such that one PbBL activity leads to another and the series of linked problems form the greater challenge or project. However, the "evangelist's" view quoted by Savin-Baden and Wilkie (2004, p2) suggests that this integrated curriculum and the problems and potential projects continue, not only through one semester or year, but from year to year, possibly even integrated across the whole curriculum – not just one silo module - that has progression from year to year.

More recently, PbBL has been widely promoted and introduced in North America with the Buck Institute for Education (BIE) which exclusively:

> "creates, gathers, and shares high-quality PBL instructional practices and products and provides highly effective services…." (BIE, 2017) to educators.

A couple of BIE publications illustrate the focus of PbBL e.g. *Preparing Students for a Project-Based World* (Lathram, Lenz, Vander Ark, 2016) and *PBL for 21st Century Success* with the publication cover strapline of '*Teaching Critical Thinking, Collaboration, Communication, and Creativity*' (Larmer, Mergendoller, Boss, 2013) explicitly emphasising that PbBL can include critical thinking, collaboration, communication, and creativity. The BIE (2017) website even presents the Forbes (2017) quote from George Lucas, the creator of the Star Wars and Indiana Jones franchises, advocating that:

> "With project-based learning, students learn by designing and constructing actual solutions to real life problems" (Forbes, 2017).

More importantly, Larmer, Mergendoller, Boss (2013, p5) define PbBL as:

> "Project Based Learning is a systematic teaching method that engages students in learning important knowledge and developing 21st century competencies through an extended, student-influenced inquiry process structured around complex, authentic questions and carefully designed products and learning tasks".

2.1.4 Computational Thinking.

In Hill and Turner (2011) the authors suggested that Wing (2006; 2008a; 2008b) has raised the profile of problem-solving within computing, by highlighting the importance of thinking like a computer scientist. By this it is meant that the thinking processes involved in being a computer scientist are more complicated than just being able to program. Computational thinking is reformulating a seemingly difficult problem into one we know how to solve, perhaps by reduction, embedding, transformation, or simulation. With the introduction of the new Computing National Curriculum (Department for Education, 2012) in September 2013, computational thinking (CAS, 2015a, b) has been promoted as including the following concepts:

- Logical reasoning: predicting and analysing
- Algorithms: making steps and rules
- Decomposition: breaking down into parts
- Abstraction: removing unnecessary detail
- Patterns and generalisation: spotting and using similarities
- Evaluation: making judgements

The following Figure 2.2 further illustrates the concepts and approaches used by a 'computational thinker':

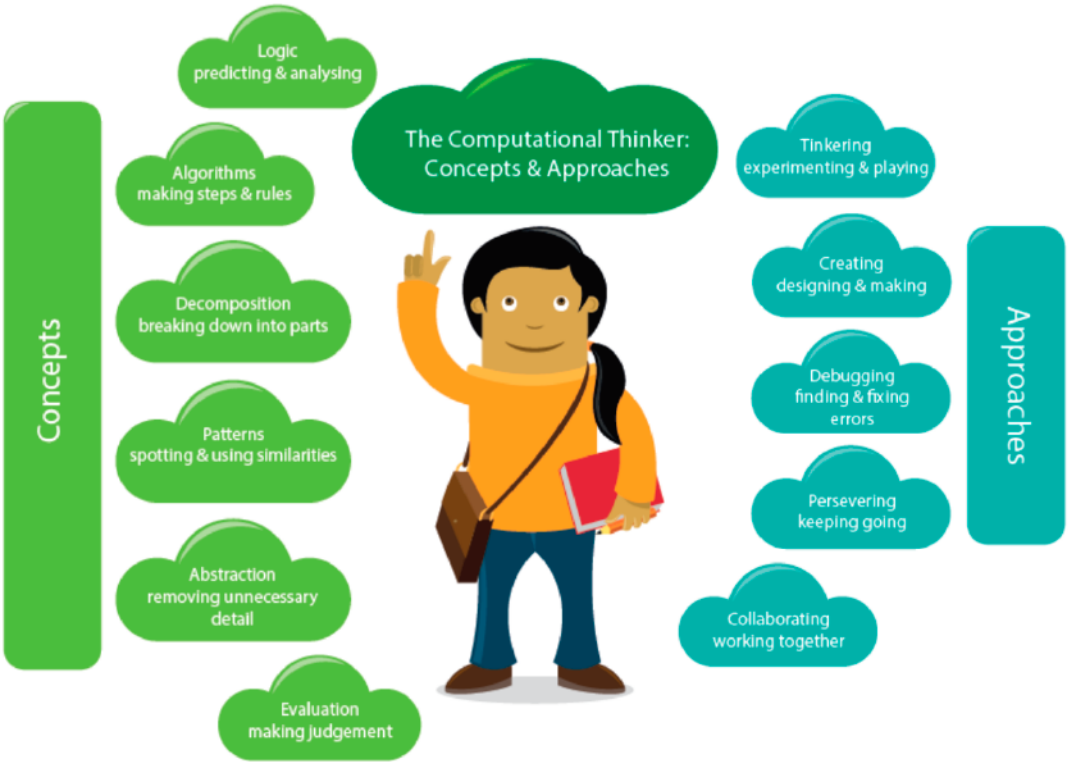


Figure 2.2: The Computational Thinker: Concepts and Approaches (Source: CAS, 2015a,b).

The concepts and approaches of computational thinking discussed here relate explicitly to the computing curriculum, but these concepts and approaches are later (cf. 2.6.2) discussed in a broader educational context related to traditional learning theories.

Computing at Schools (CAS) have also released a dedicated Computational Thinking guide for teachers in late 2015 (CAS, 2015b) which emphasises the importance of Computational Thinking.

In summary, computational thinking should be considered an applied form of analytical thinking to include critical thinking. Computational thinking is more directed and structured to the concepts and approaches used within computing. The authors suggest (Hill, Turner, Childs, 2017) computational thinking to be the approach best suited for problem solving within the undergraduate and postgraduate computing curriculum. The following sections (cf. 2.2 to 2.6) will justify the importance and relevance of computational thinking to the approaches adopted and advocated here to improve first year undergraduate computer programming.

This section (cf. 2.1.1 to 2.1.4) has sought to define and clarify the main terms and concepts (Figure 2.1) used in the published works and this critical appraisal. The following chapters (cf. Ch.2 to 5) will justify the importance and relevance of these terms to the approaches adopted/advocated here to improve first year undergraduate computer programming.

2.2 Review of Published Works.

The following sections (cf. 2.2-2.5) introduce and discuss, chronologically, the published works (selected and supplementary published works, Tables 1.1 and 1.2 respectively) that coherently demonstrate the evolution and evaluation of the proposed improvements to first year undergraduate computer programming, to enhance student learning. This is achieved by the following discussions:

- A critical summary of the publications selected (Table 1.1);
- Contextualizing the selected publications;
- Demonstrating the coherence of the selected publications;

The discussion of the papers will demonstrate that the author's initial publications advocated that a problem solving or problems-first approach was felt by the author to be an enhancement

to teaching computer programming. This approach was enhanced and evolved such that it was felt by the author to be important to continue the 'problems' approach through the whole teaching of this subject, hence the term used as the title of a later publication (Hill and Turner, 2014a) 'Problems first, second and third' which reflects the view that problems-first and progression to PbBL and PjBL approaches were advocated by the author.

In addition, the author also advocates that computer programming always requires problem solving skills and computational thinking.

The discussion of the papers also demonstrates the importance of the visual approach adopted throughout i.e. Graphics-first.

Figure 2.3 graphically illustrates the chronology/evolution of the published works in relation to the keywords/concepts introduced. The colour temperatures relate to the timeline (illustrated in the bottom right-hand corner) of the published works. The use/density of the keyword/concept is emphasised by the size of the circle next to the keyword, with the thickness of the interconnecting lines emphasising the strength of the relationship. A glance at Figure 2.3 will clearly show the interrelated concepts with a resulting emphasis on 'problem', 'programming' and 'teaching' which further justifies the first sentence above (cf. 1.1) where the author states that: '*This critical appraisal, predominantly discusses the* ***teaching*** *of computer* ***programming*** *and* ***problem*** *solving to undergraduate first year computing students…*' (cf. 1.1).

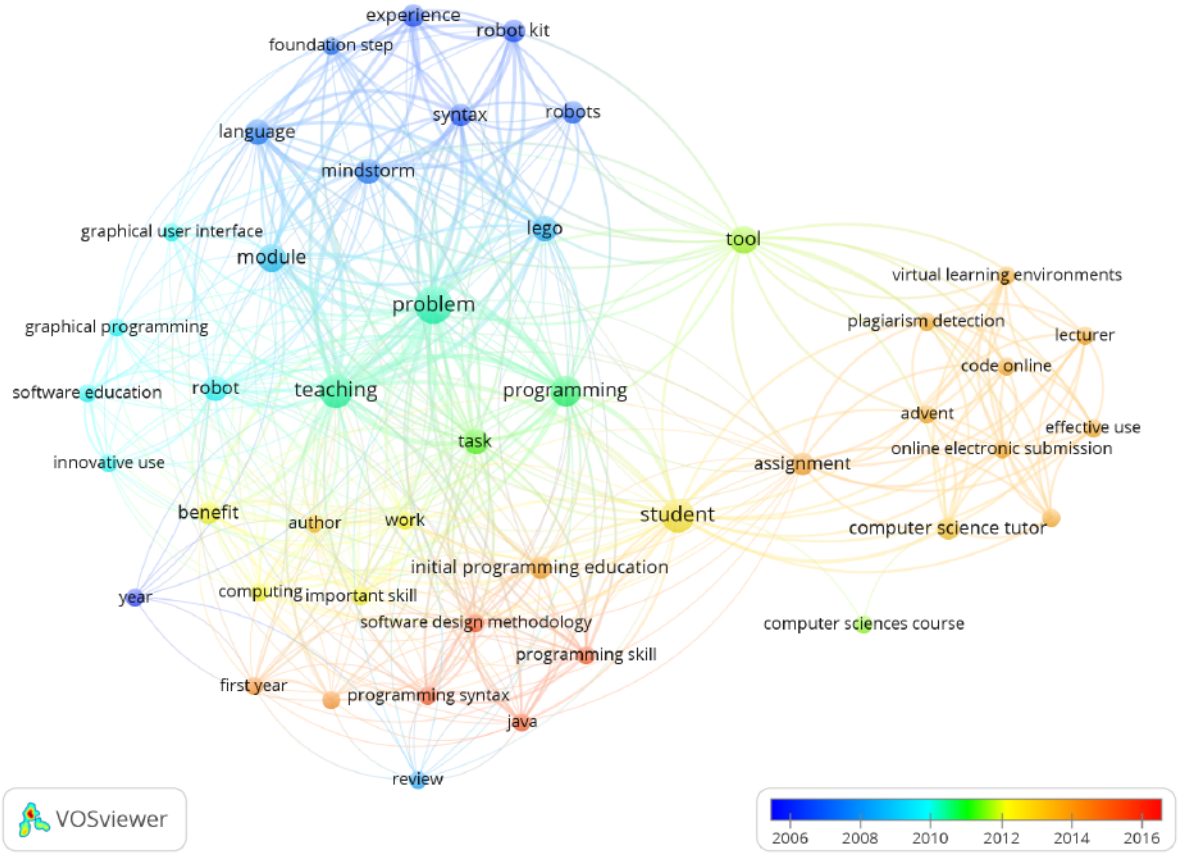


Figure 2.3: Bibliographic Research (Timeline) extracted from NECTAR (2016) using VOSviewer (2016).

The selected 10 key/core published works that form the body of this critical appraisal are items 1 to 10 (Table 1.1) together with the 7 supplementary published works 11 to 17 (Table 1.2). See also, Appendix 6 and 7 for the full text for each of the selected and supplementary published works.

The next two sections (2.3 and 2.4) provide an account of the published papers (Table 1.1 and 1.2) interrelationship and chronology in the development of the author's teaching of *Visual* PbBL and PjBL. The first section (2.3) covers the first eleven years (1999-2009) and the second (2.4) covering the next eight years (2010-2017) respectively.

2.3 Visual Problem Based Learning (PbBL) and Visual Project Based Learning (PbBL) – The first eleven years (1999-2009).

Chapter 3, discusses research methodologies and the choice of action research for this study, but recognises that action research can be 'messy' as, whilst action research should be sequential and cyclical in development and progression, the various strands of an action research study are likely to progress at varying rates of development and progression. This (cf. 2.3) and the following section (cf. 2.4) offer a critique of the chronological overview of the progression and evolution of the authors published works, whilst attempting to illustrate the progression as an action research study. Figure 2.4 attempts to diagrammatically illustrate a simplistic progression and evolution of the action research study by:

- highlighting the progression, due to the implementation and initiated change, based on the previous publication/s and experience, and;
- the evaluation and reflection of the previous publication/s and experience.

The figure (Figure 2.4) also identifies any requests received from publishers requiring an update on the previous publication and experiences gained. In addition, for clarification, Figure 2.5 attempts to illustrate the thematic progression, development and evolution of the published work (as a direct result of action research). Each study/publication always fed into the next, such that each subsequent paper was informed by the previous, but the figure (Figure 2.5) shows the themes that have developed chronologically, mainly by optimising and deepening the initial teaching and learning pedagogical approaches, through a variety of data collection and analysis methods (cf. 3.3).

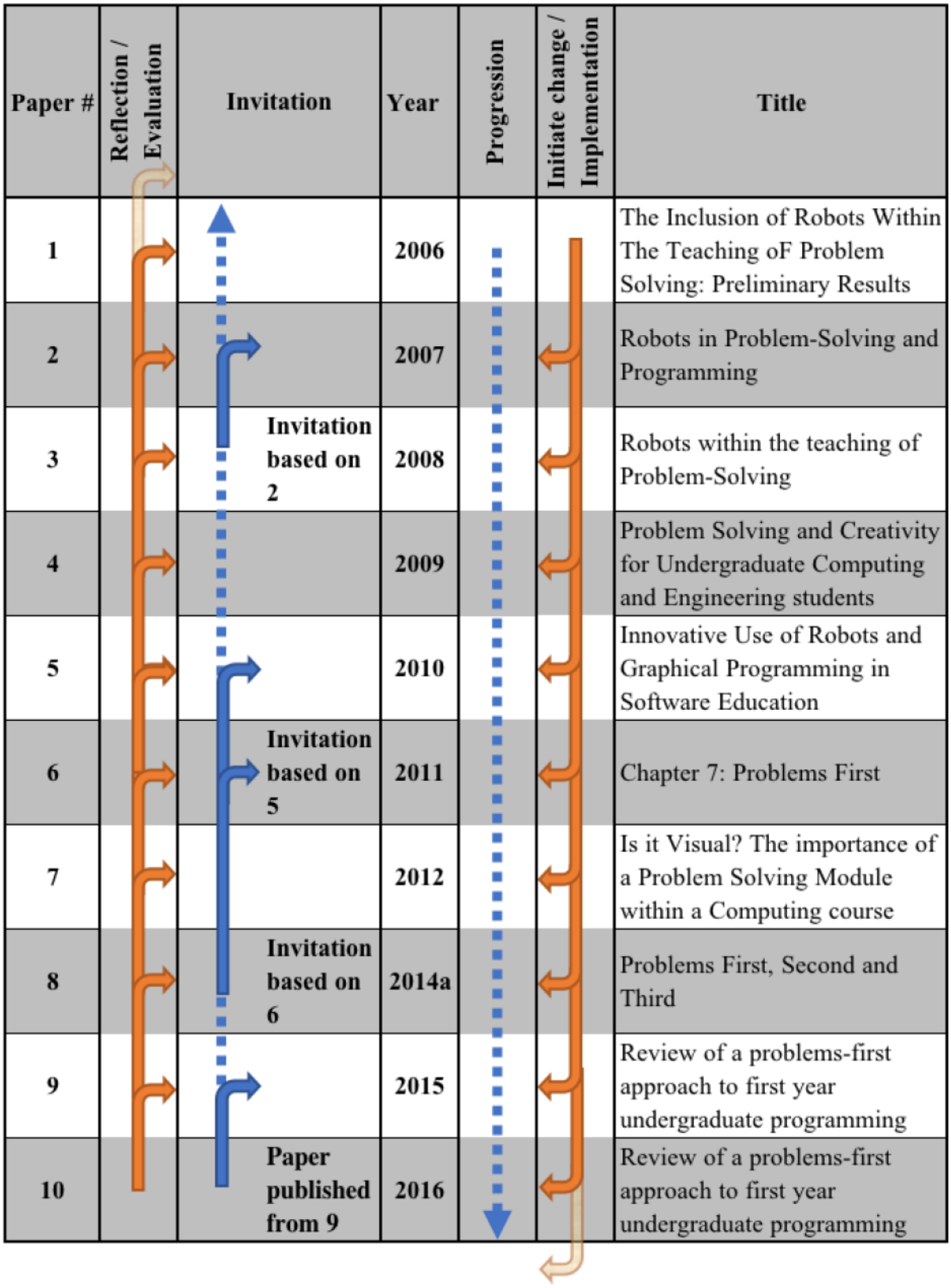


Figure 2.4: Chronological evolution and progression of the authors published works, as an action research study.

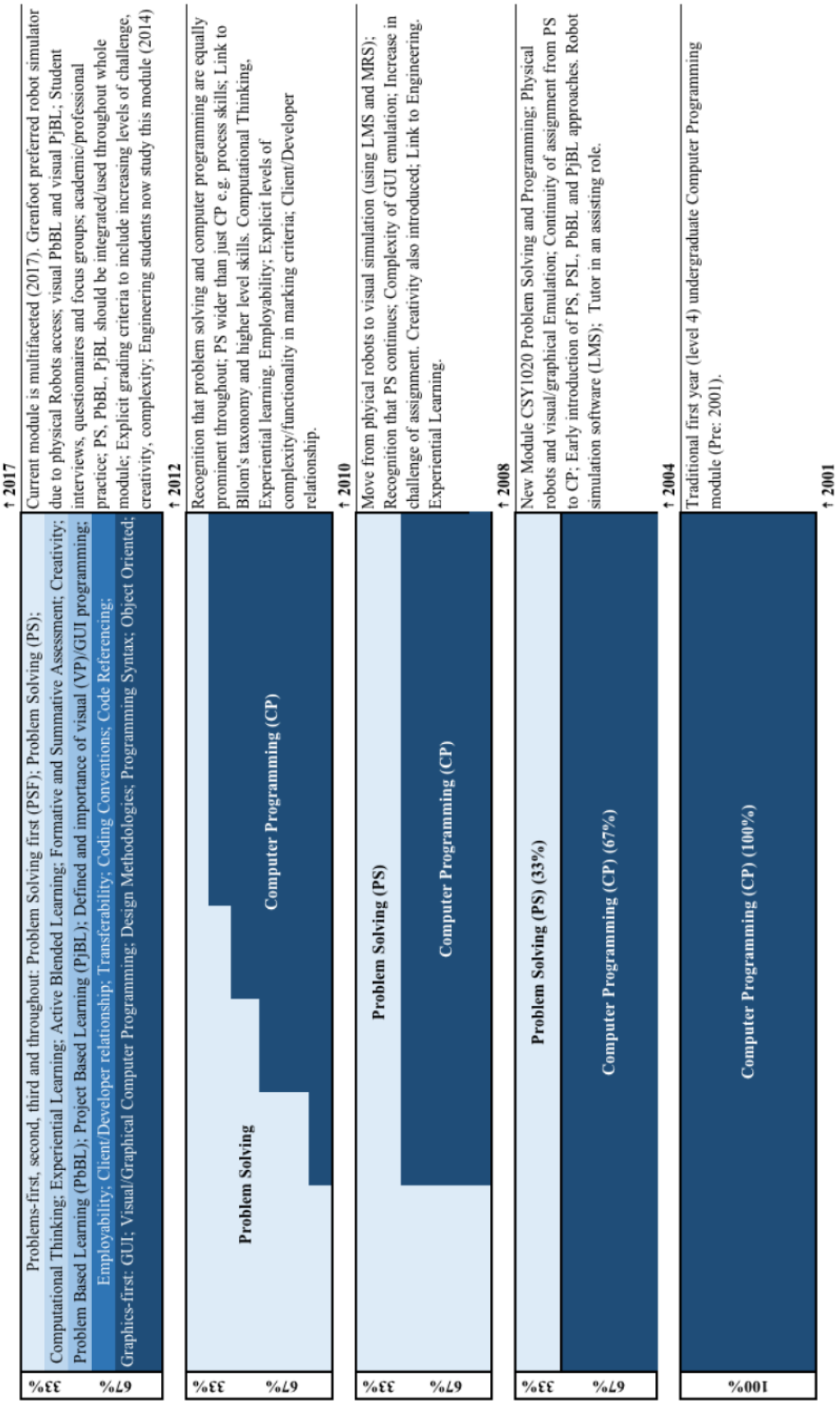


Figure 2.5: Chronological evolution and progression of 'themes'.

Based on a Java and the Internet in the Computing Curriculum (JICC, 2001) conference discussion (cf. 4.2.1) in 2001. The author suggested introducing problem solving techniques prior to computer programming. In 2004 a new level 4, 20 credits module was devised (Appendix 8) and introduced by the author to introduce problem solving prior to computer programming. The module was called "CSY1020 Problem Solving and Programming" and included graphical/visual computer programming from the outset (graphics-first) based on the authors' experience since 1999 of non-visual and visual programming (cf. 1.1). This section covers 1999-2009 and section 2.4 covers 2010 to 2017.

Two years after the introduction of the CSY1020 Problem Solving and Programming module, in 2006, the authors presented the preliminary findings of the CSY1020 module, through a conference poster (Turner and Hill, 2006). Highlighting, predominantly, students' feedback from using Mindstorm (LEGO) robots (Figure 2.6) for problem solving as a precursor to the computer programming element of the module and the linking of the two parts of the module. One student commented on their module feedback stating there was "Good progression from problem solving to java".

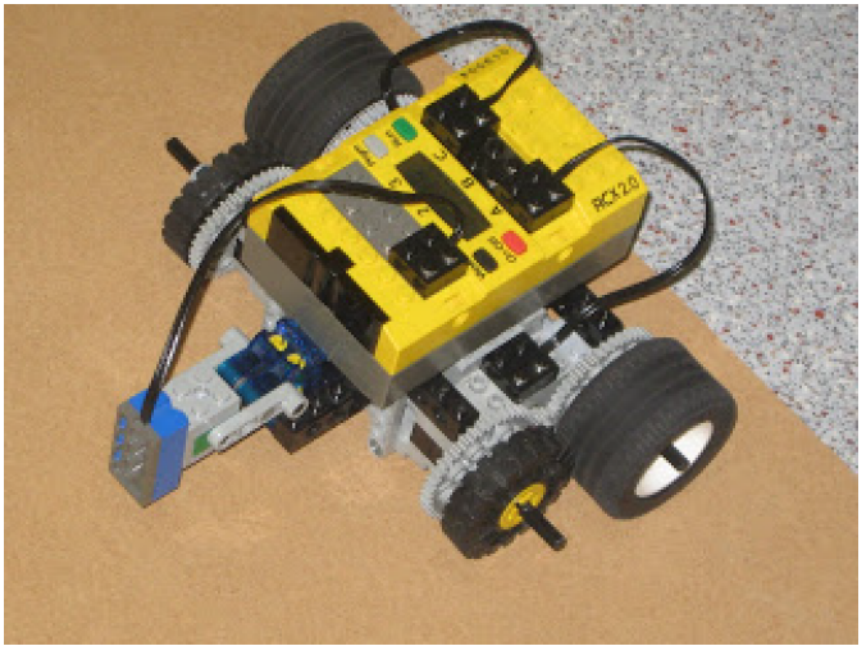


Figure 2.6: LEGO Robot used to navigate an 'M' shaped maze as in Figure 2.7 (Source: Turner).

In summary, this presentation (Turner and Hill, 2006) briefly discussed the findings from the first academic year using programmable robots and the linking of the two assignments (assignment 1: problem solving and, assignment 2: programming). An overview of the assignments was only briefly mentioned and that student feedback, including a formal questionnaire, suggested they liked the way the assignments linked from the problem solving to the programming part of the module. The conclusion was that the preliminary findings suggested the approach was worthy of further investigation, in relation to:

- the physical and visual approach to the module being liked by the student;
- the link and progression from the problem solving to programming module and assignment, and;
- consider increasing the complexity and potential challenge of assignment problem/scenario.

In 2007, the Turner and Hill (2007) paper gave the opportunity to not only evaluate the continued development of the CSY1020 module, problem solving and programming (cf. 3.1 for a more detailed explanation of the methodology used) after two academic years, but to address the issues highlighted as requiring further consideration and investigation in the previous paper (Turner and Hill, 2006). This paper (Turner and Hill, 2007) articulated in more detail, particularly the specifics of, the module operation and the students' views. The paper (Turner and Hill, 2007) explained the operation of the module in terms of: hours per week; number of weeks; the specifics of the problem solving exercises used, and; that the assignments were linked.

Figure 2.6 illustrates the use of a LEGO Robot being programmed to navigate an 'M' shaped maze. Figure 2.7 illustrates the use of a visual/graphical programme where the black squares show the path that the 'visual' robot has followed, in this case, leaving the 'M' shape. The two key observations were that the students liked the way that the:

- robots aided with a visual or physical representation of the problems, and;
- problem solving assignment provided good progression to the computer programming assignment – as the second assignment repeated the earlier problem but required a graphical computer programming solution (Figure 2.7).

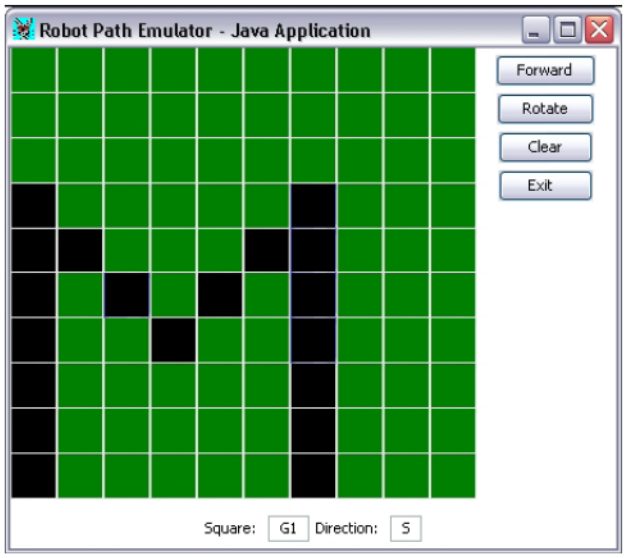


Figure 2.7: Prototype Graphical User Interface from the computer programming assignment (Source: Turner and Hill, 2007, p84).

The paper (Turner and Hill, 2007) emphasised that problem solving is not trivial (Beaumont and Fox, 2003) and is an important soft and transversal skill, central to computing and engineering.

The computer programming element of the CSY1020 module explained the introduction of Graphical User Interfaces (GUIs) as early as possible (graphics-first) and that the author felt the visual programming aided engagement, enjoyment and learning. Further evidence that students liked the transferability of skills was also highlighted where:

> "One student made the explicit comment that they felt there was a 'good progression from problem-solving to programming'. In addition, the students commented that they could take the ideas developed in one part of the module to the second part, thus evidencing clear transferability of skills" (Turner and Hill, 2007, p84).

In addition, the paper concluded that:

- the physical and visual nature of the module was liked;
- the term PSL was used to describe the approach adopted in the module and;
- a problem-based learning approach should be considered as a further development of the module.

Therefore, the three themes of physical and visual, PSL and PbBL were felt by the author to warrant further exploration and implementation into the CSY1020 module. It was also the first paper to mention that each small problem solved progressed the assignment in functionality of the GUI application and the term PSL was used to describe this. Access to physical robots was identified as a potential limitation and that robot emulation software should be considered. The need to include further increased challenge within the assignments/problems was also suggested and that students still liked the visual and physical outcomes. The paper also concluded, with the first mention, that the author should adopt or consider a PbBL approach with the module tutor taking an assisting role.

Whilst not explicitly stated in the paper (Turner and Hill, 2007) it could be seen that another evolving innovation introduced by the authors was to take a more flexible/open approach to PbBL such that, for the first time, the module tutor role could be seen as assisting/supporting the students to find, explore and solve the problems, instead of just being responsible for setting the problems. This was not articulated explicitly until later (Hill and Turner, 2011) where the role of the tutor as the 'client' and the student as the 'developer' were explained. This paper (Turner and Hill, 2007) was also the start of a developing argument for pedagogical innovation relating, initially, to PbBL, later published in explicit detail in Hill and Turner (2014a) and Hill (2015, 2016).

Based on the first Higher Education Academy-Information and Computer Science (HEA-ICS) paper (Turner and Hill, 2007) the authors were invited to write another paper for the HEA-ICS 'Italics' journal. This paper (Turner and Hill, 2008) citied by Gold (2010, p11) explained in yet more detail the evolution, mechanics and subtleties of the operation of the CSY1020 Problem Solving and Programming module.

A conclusion of the previous paper (Turner and Hill, 2007) was that the PbBL approach should be considered as a further development of the module. The 2008 paper (Turner and Hill, 2008) implemented and articulated for the first time, the PbBL/PjBL nature of the module, where for each small problem solved, the students were rewarded with the increasing functionality of their GUI application. In addition, whilst the assignment/project was complex, the students could appreciate that with the revealing and introduction of each new

aspect of the computer programming language, this could be directly applied to their assignment.

This new approach provided the opportunity to offer more advanced students greater flexibility/openness in their problems/assignments, with suggested options of extra functionality and complexity that they could incorporate into their assignment solution to gain a higher grade e.g. Figure 2.8a) shows the 'model' default solution, with Figure 2.8b) illustrating an 'enhanced' student submission which includes additional functionality and complexity. It was found that approximately 15% of students (out of 60) availed themselves of this opportunity and the annual module evaluation reported high levels of satisfaction (90%+ satisfaction).

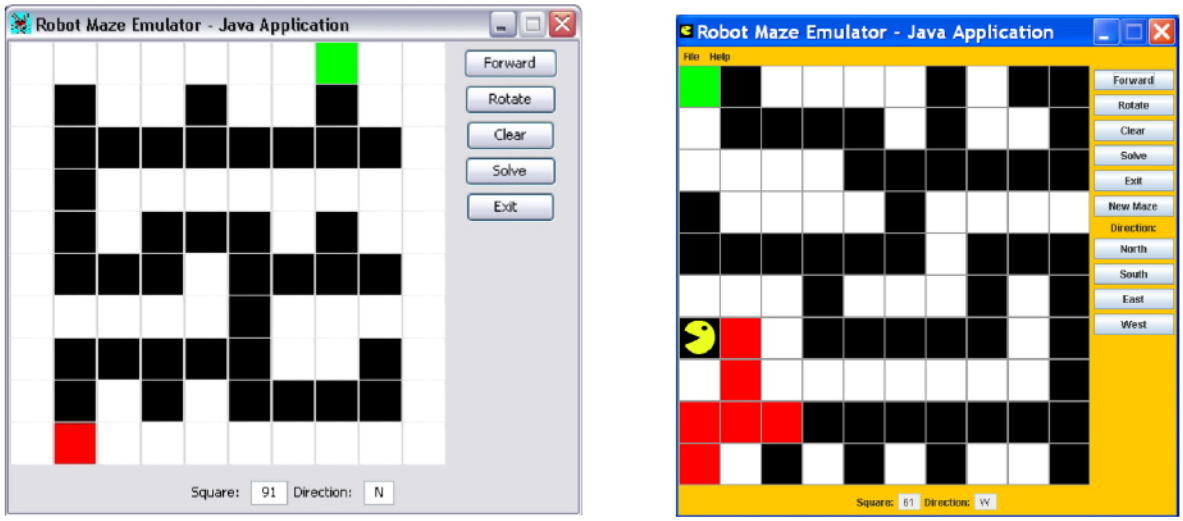


Figure 2.8: a) Model (left) and b) Enhanced (right) Solutions (Source: Turner and Hill, 2008, p112-3).

The importance of the visually-oriented methods employed in the module were also seen, by the author's professional assessment, as beneficial (e.g. learning styles/modalities cf. 2.6) to the student learning.

> "The 'eureka' moment is evident with each small problem solved and the increasing functionality of their GUI application" (Turner and Hill, 2008, p112).

This is due to:

a) the instant visual feedback received by the student from the GUI/visual computer application, e.g. the computer application opens with the added visual components, and,
b) the visual interaction achieved with the GUI/visual computer application e.g. by pressing the 'Forward' button (Figure 2.8a) the robot (represented by the red square) should move forward along the predefined path – in this case upwards/north onto the next black square).

This links to both visual styles of learners e.g. Visual, Aural/Auditory, Read, Kinaesthetic (VARK, cf. 2.6.2) by Fleming and Mills (1992, p138) and the perceived benefits of visual representations of a solution using visual programming (cf. 4.5 Physical to visual/visual programming).

This paper (Turner and Hill, 2008) gave the opportunity to emphasise the novel Java programming teaching approach, with GUI and visual programming introduced at the outset (graphics-first) and throughout. The complexity of GUI emulation, expected from the assignment, was explained, indicating the clear evidence of transferability that was not only liked by the students, but an important academic/professional skill. The link between the concepts learnt, about the programming language, to the assignment were discussed i.e. the emphasis and application of GUI specific aspects of the programming language (Java: GUI layout managers) and the impact on the assignment of some of the simpler concepts (Java: repetition using 'for' loops).

Due to issues of limited access to physical robots, two robot software emulators/simulators (Microsoft Robotics Studio (MRS) and Lego Mindstorms Simulator (LMS) were used and discussed for first time. However, students found one of the simulators (LMS) a little harder to use and the computer programming code used by the simulator required additional modification. The use of the simulator (LMS) diverged from using a defined programming language throughout the CSY1020 module (as the Lego robots used Java) which had previously been an important prerequisite of the module.

In summary, the author concluded that further exploration and implementation of a visual problem solving and visual PbBL approach should continue to include:

- the opportunity for the assignment/solution to incorporate additional functionality and complexity;
- encouraging the use of creativity;
- robot simulation software should be investigated further.

The evolving PbBL/PjBL nature of the module had been described, although the terms PbBL/PjBL had not yet been explicitly used or defined (cf. Hill and Turner (2014a) and Hill (2015, 2016)).

In 2009, the Computing module was continuing to evolve, but the opportunity arose to share and test the research across two related disciplines: computing and engineering (Turner, Hill and Adams, 2009*)*. The previous two publications (Turner and Hill 2007, 2008) had mentioned the link of using robots for teaching problem solving and programming to computing and engineering students (Lawhead et al,2003; Price et al,2003; Williams et, 2003 and Fagin 2003) and encouraging creativity in the assignment solutions. This earlier work (Turner and Hill 2007, 2008) with computing undergraduates showed the potential for developing computer programming through problem solving skills, PbBL and also encouraging a creative approach. Adams *et al*. (Adams and Turner, 2008, Adam *et al*., 2008, 2010 and Turner, Hill and Adams, 2009) included assistance from the author to develop a dedicated problem solving and creativity module for engineering undergraduates.

The workshop presentation (Turner, Hill and Adams, 2009) gave a brief comparative side-by-side review of the delivery of a two-year cycle (academic years 2007-2008 and 2008-2009) of the two modules i.e. the engineering level 4 module using problem solving and creativity and the corresponding computing module (CSY1020 problem solving and programming).

The paper (Turner, Hill and Adams, 2009) concluded that the engineering module was considered difficult by the students, as they were not as familiar with computer programming. Therefore, the engineering module initially diverged from the approach of computing and used robot simulation software, that was not Java computer programming based. Adams published three papers (Adam *et al*., 2008, 2009 and 2010) on the problem solving and

creativity approach. In 2014 it was felt by engineering programme team at the UoN that it would be more beneficial for the engineering students to learn computer programming and therefore they studied the CSY1020 Problem Solving and Programming module with the computing students. This collaboration between computing and engineering explicitly indicated at an early stage that this research was innovative and had an impact on teaching in another allied discipline (cf. 5.1).

The 2009 paper (Turner, Hill and Adams, 2009*)* again outlined the new approach used within the Computing module of using a similar assignment scenario to test different skills. Firstly, problem solving and secondly programming, enabling students to progress through the module encouraging creativity through increased freedom/flexibility. The students could choose to use Basic; Moderate and; Advanced functionality/features/complexity in their solutions. The presentation concluded that with each year there was an increase in challenge available within the assessed work. The transferability of work from one assignment to the other, highlighting that explicit re-use and adaptation of previous robot routines from one assignment, was encouraged. It was also evident that physical robot access was becoming a limitation due to a rise in student numbers (in excess of 100 students in 2008, rising to 180+ in 2016) and the flexibility of using an appropriate robot simulation software should be further investigated. Two types of robot simulation software (LMS/MRS) had already been evaluated (Turner and Hill 2007, 2008) and the later publications of Kariyawasam, Turner and Hill (2012) suggesting 'Greenfoot' as an alternative, with Hill and Turner (2014a) discussing its implementation.

Finally, the Turner, Hill and Adams (2009) paper posed a question of "Has it improved students programming ability?" which took a further 3 years to attempt to answer (Kariyawasam, Turner and Hill, 2012) (cf. 2.4) as in 2011 UoN funding was obtained for a student researcher to consider the views of the students over the 3 years of their studies. This study surveyed level 4, 5 and 6 students to consider their perception of the benefit of the CSY1020 module (cf. 2.4). The results of this study and how they were used to inform future changes to this study will be discussed in section 2.4, but it is worth mentioning that the original aim of the author was to evaluate and propose improvements to enhance student learning of computer programming and this paper (Kariyawasam, Turner and Hill, 2012) gave the opportunity to evaluate the views of a full 3 years of student opinion.

In summary, the first eleven years (1999-2009) explored Visual Problem Based Learning (PbBL) and Project Based Learning (PbBL) via:

1. Design and implementation of a new problem solving and programming module (CSY1020).
2. Initial problem solving without computer.
3. Initial problem solving using robots.
4. Visual and graphical programming introduced as the medium for computer programming from the outset and throughout.
5. Problem solving to aid the teaching of computer programming.
6. Continuity of assignment between the Problem Solving and the problem solving and programming assessments.
7. Use of PbBL to the assignment.
8. Dual role as the module tutor setting the problem, but also assisting with the solution.
9. The link between the PbBL aspect of the module, building to solve a larger problem or project and PjBL.
10. Increase in complexity/functionality of the computer programming assignment, whilst providing clearer feedback from the application to help with debugging.
11. Opportunity to incorporate creativity.
12. Collaboration with the Engineering discipline indicating impact in an allied subject.

The 12 items above summarise the key achievement from the early publications. The following section (cf. 2.4) continues this discussion for the latter years (2010-2017) explicitly demonstrating the continued evolution and evaluation of the proposed improvements using action/practitioner research as the methodology for this research (cf. Chapter 3).

2.4 Visual Problem Based Learning (PbBL) and Visual Project Based Learning (PjBL) – The last eight years (2010-2017).

In 2010 the Turner and Hill (2010) publication further emphasised that the assignment problems/projects scenarios and their complexity, flexibility and creativity was still evolving and that excellent student feedback (90%+ satisfaction) via formal annual module feedback, as well as anecdotally, continued. The author still felt that from all the previous experience and publications (Turner and Hill, 2006, 2007, 2008 and Turner, Hill and Adams, 2009) that the graphical/visual approach to computer programming adopted from the outset (graphics-first) and transferability of the problem solving to the graphical computer programming task (Figure 2.9) aided the student engagement, enjoyment and learning. The paper also, for the first time, expressed that problem solving had a wider benefit than just a precursor to computer programming, due to the link to process skills. Therefore, it was evident that the approach being adopted, and the surrounding concepts shared a close affinity with Computational Thinking (cf. 2.1.4) and the work of Wing (2006; 2008a; 2008b) (cf. 1.3, 2.1.4). Computational Thinking was discussed further in the Turner and Hill (2011) paper. The author also advocated the effectiveness and importance of introducing and developing problem-solving concepts as a precursor to technical computer programming in Computer Science education. Highlighting that, initially, any teaching of computer programming begins with simple problem solving with the minimum of commands. The students learn almost unconsciously, with computer programming concepts and objects unknowingly used. These are then later introduced/learned during the more formal element of the computer programming stage of the computing module (CSY1020 Problem Solving and Programming).

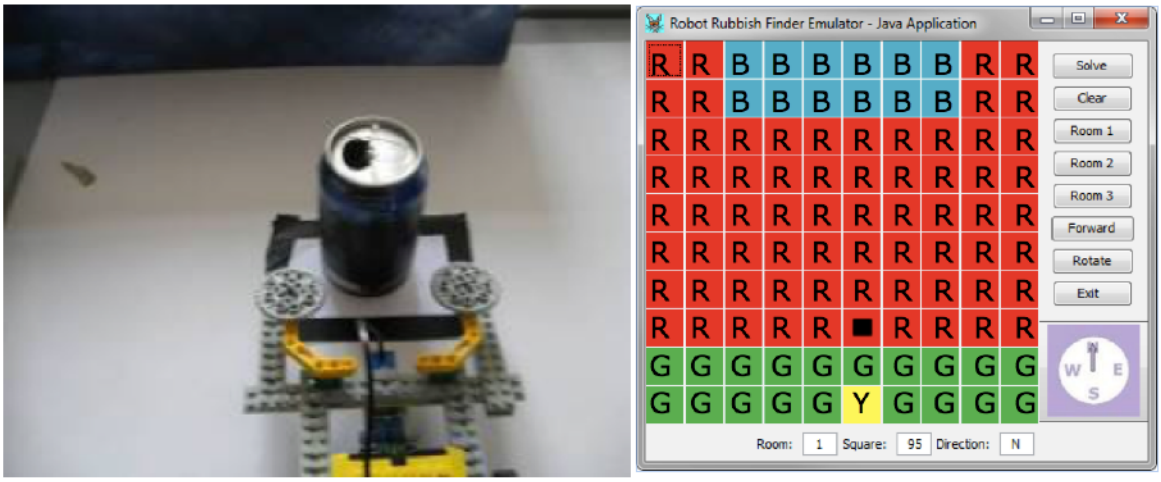


Figure 2.9: Physical and Visual Examples (Source: Turner and Hill, 2010, p55).

In 2011, a direct approach to the authors was made from publisher IGI-Global to write a chapter (Hill and Turner, 2011) about the problem solving and programming - 'Problems-first' - approach and to summarise the experience and findings. The chapter was published in a special edition entitled *'Software Industry-Oriented Education Practices and Curriculum Development: Experiences and Lessons'* (Hussey, Wu and Xu, Eds., 2011). This was the first time the term Problems-first' was used as an abbreviation to problem solving and PSL, in a similar way that the term graphics-first was used to define the use of graphics/visual programming from the outset and throughout. It was felt that the term 'problems-first' was appropriate to define the use of problem solving and PSL first (later (cf. 4.1) defined as Problem Solving First (PSF)) from the outset and also throughout. The chapter highlighted synergies and supporting evidence of similar approaches used elsewhere within the computing discipline e.g. the synergy of teaching programming using robots and teaching programming using not just problem solving, but visual programming and the simulation of robots. The chapter offered, for the first time, a detailed technical computing overview of the PbBL/PjBL nature such that problems (new computer programming concepts) were introduced, taught, practiced and used to incrementally solve another element of the project/assignment. The chapter explicitly, for the first time, explained in detail the evolution of the tutor's role, first eluded to in Turner and Hill (2007). The 'real-world' professional practice roles of the lecturer as the 'client' and the student as the 'developer' was introduced and related to their relevance in the computing/software-oriented industry (i.e. employability/graduate skills). This approach aimed to not only assist with the growth and development of the student through experiential learning, but to emphasise and reinforce the

relationship to other experiences and learning contexts, ultimately ensuring that the transfer of learning/experience to similar and dissimilar learning contexts took place (cf. 2.1.5). In particular this included the emphasis of key employability skills and context.

The author still maintained the supportive 'facilitator' role of introducing, assisting and reinforcing the relevant tools/concepts to enable the solution of the problems/project as discussed earlier (cf. 2.3, and Turner and Hill, 2007). The facilitator's role, including scaffolding of learning, relates to the constructivist learning theory (cf. 2.6.2) where the lecturer:

> "instead of being the 'sage on the stage' functions as a 'guide on the side' facilitating learning in less directive ways" (King, 1993, p30, cf. 2.1.2).

The 'developer' role of the student within a 'development team', reinforced in terms of coding conventions, led to the supplementary Hill & Turner 2012 & 2013 papers on referencing and academic scholarship. The authors identified the need and importance of a clear coding convention (readability, referencing, code comments, file headers, debugging) when writing computer code in software engineering, which became the focus of a supplementary published paper the following year (Hill and Turner, 2012). The need for clear referencing and commenting of computer code is an important vocational skill to possess and was included in the CSY1020 module, but in addition to the vocational need there had been issues with plagiarism/academic integrity where the similarity of some student work required further investigation. The author researched this issue and found that there appeared:

> "to be no explicit method, recommendation or advice available to computer science tutors and students on a referencing approach!" (Hill and Turner, 2012, p1).

The other supplementary paper (Hill and Turner, 2013) was also written to explain these issues and suggested solutions related to, and in determining, the similarity of student computer code when electronically submitted through a Virtual Learning Environment (VLE) and subsequently the use and evaluation of the data obtained from the plagiarism/similarity tools available.

The chapter (Hill and Turner, 2011) maintained that there had been a continued indicative increase in the grades and positive responses/feedback from the students. The roles of the

'developer' and fictitious 'development' team reinforced the transfer of learning/experience to similar context of employment e.g. the 'developer' needed to be aware of the importance and influence of their role within a 'development team'. That what they were being asked to do was not just an academic exercise but had important 'real-world' transferability. Alexander and Irons (2004, p12) support this view that:

> "Computing programmes must develop the skills required to become a computing professional and transferable skills are an important component of this skills set".

Hill and Turner (2011) suggested that problem solving and programming should be considered to be non-trivial and linked to Bloom's taxonomy (Bloom, 1956) and higher-level skills like: synthesis; evaluation; analysis and; application (cf. 2.6.2). Computational thinking (previously mentioned in Turner and Hill, 2010) and thinking like a computer was emphasised as being more complicated than just computer programming. Emphasis was given to the importance of Computational Thinking within the module (CSY1020) of reformulating a difficult problem by reduction, embedding, transformation and simulation (cf. 2.1.4). Approaches of software design methodology were introduced throughout the module to support the professional and transferable skills, to include analysis, design, implementation, evaluation, testing and reflection. The software design methodologies were related to the students in terms of the higher-level skills expected within the module, the assessment and its accompanying technical report. The paper also gave an update on the newer problem/project scenarios that increased explicit complexity, with the assessment criteria indicating basic, moderate and advanced functionality and complexity. It was also noted that there had been an increase to 25% of students opting for the more advanced, challenging, complex and creative aspects of assignment. The paper raised the issues still being experienced by the robot simulator software and that this would be investigated further (a possible alternative was proposed in Kariyawasam, Turner and Hill (2012) and its implementation discussed in Hill and Turner (2014a)).

In 2012, the Kariyawasam, Turner and Hill (2012) paper (cf. 3.2) considered the students' perspective from research collected/collated by a student researcher (Kariyawasam) under an initiative within the University (URB@N - Undergraduate Research Bursaries at Northampton). Entitled 'Is it Visual? The importance of a Problem Solving Module within a Computing Course', the aim of the study was to consider the students' view (from interviews,

questionnaires and focus groups) relating to the usefulness of a problem solving and programming module in the first year of a 3-year undergraduate programme. This was an opportunity to consider formal evaluation of the module in addition to the previous publications where student views were sought through formal annual module evaluation questionnaires and the anecdotal evaluation and observations from the author whilst delivering the module. The study also investigated whether, after seven years of the teaching of problem solving, a more **visual** approach had benefits (the visual computer programming included both two-dimensional graphical computer programming and GUI design and development). All students interviewed either had completed the module within the two years of the survey or were completing the problem-solving module in their first year. The paper concluded that there appeared:

> "to be a perceived increase in the positive impact of using robots reported by second and third year students, which seems to indicate that the visual and physical nature of using robots is liked and ultimately appreciated by the students" (Kariyawasam, Turner and Hill, 2012, p7).

The study also obtained some supportive comments as follows:

> "The visual nature helped to identify errors in the programming logic and made it easier to rectify any errors made."

> "It helps me think through a problem if I can visualise it.

> "It's a gentle introduction to the problem-solving concept and how it applies to the course and modules later…" (Kariyawasam, Turner and Hill, 2012, p7).

These first two comments reinforce that the visual nature of the CSY1020 module is seen by the student as helpful/positive. These two comments are also mentioned later (cf. 4.5.2) when reinforcing the perceived benefits from two respondents to the problems solving teaching and learning approaches adopted.

The third comment helps to reinforce the proposition that the students appreciate the importance of problem solving and programming skills not only from the outset (Year 1) but appear to appreciate the 'visual' nature as they advance in academic level (Figure 2.10). As if

in hindsight they appreciated the benefit of how the module had prepared them for their future years of study, on their academic programme. For clarification, Figure 2.10 shows the breakdown of results by academic year as:

- Year 1 – level 4 - (Yes = 59%, No = 41%)
- Year 2 – level 5 - (Yes = 67%, No = 26%)
- Year 3 – level 6 - (Yes = 77%, No = 23%)

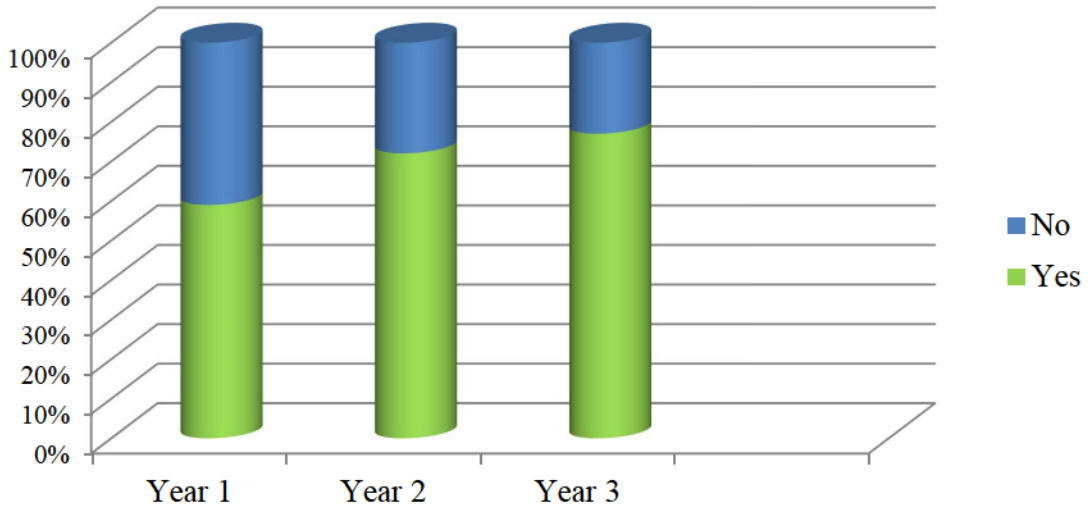


Figure 2.10: Questionnaire: Did the visual nature help you to develop problem solving skills? (Kariyawasam, Turner and Hill, 2012, Fig 2, p6)

The 2012 paper (Kariyawasam, Turner and Hill, 2012) is discussed in more detail later (cf. 4.5.1) to emphasise the perceived benefits of using robots, expressed by two respondents:

> "…it takes the concept of problem solving and places it into a physical and tangible domain…"

and

> "It's easier to understand something if you can touch it and try it in real life, rather than seeing it on a screen" (Kariyawasam, Turner and Hill, 2012, p7)

The paper also concluded with a recurring theme of limited physical resource of the robots as a limitation and criticism of the module. One second year (level 5) student commented that there were: "Not enough resources to go round – e.g. robots".

Two types of robot simulation software (LMS/MRS) had already been evaluated (Turner and Hill 2007, 2008) and this later publication (Kariyawasam, Turner and Hill, 2012) suggested investigating Greenfoot (2015) as an alternative, with Hill and Turner (2014a) discussing its implementation.

Three years after their first approach in 2011, IGI-Global approached the authors, to write a further update on their experience (Hill and Turner, 2014a). This paper primarily focussed on the 2011 paper (Hill and Turner, 2011) but also the Kariyawasam, Turner and Hill (2012) and supplementary papers Hill and Turner, 2013 and 2014b). In addition, wider research findings from other authors e.g. Gold (2010) and group work was discussed. At this stage, it was still thought that, whilst the problems-first approach was felt by the author to be an enhancement to teaching computer programming, it was important to continue the 'problems' approach through the whole teaching of this subject, hence the title of 'Problems first, second and third' to reflect the problems-first and progression to problem-based (PbBL and PjBL) approaches advocated by the author. The author, also, advocated that computer programming always requires problem solving skills and computational thinking.

This paper (Hill and Turner, 2014a) reiterated the Problem/PjBL 'real-world' professional practice roles of the lecturer as the 'client' and the student as the 'developer'. Together with the experience of 'developers' and the importance of clear referencing of computer code (Hill and Turner, 2012). The issue relating to the assessment of PbBL/PjBL was also raised, as the author identified that the adoption of electronic online marking and similarity checking meant that the electronic online marking of project/problem based software assignments, led to some high similarity percentages. [Note: In 2010, the University of Northampton introduced electronic online submission and marking of assignments. Assignments were submitted via a Virtual Learning Environment (VLE) Blackboard (2017) using 'turnitin' (Turnitin, 2016) as the plagiarism detection software. To conduct the marking a GradeMark (2017) plugin tool is used alongside (Hill and Turner 2013)]. With discussions leading to the use of similarity checking using turnitin (Turnitin, 2016) another paper explaining the reason and acceptance of high similarity in this type of assessment was published (Hill and Turner, 2014b). Whilst the unique PbBL/PjBL assignments chosen by the author ensured that the solution was unique to the cohort, this meant that higher percentages of similarity of solution/code, within the cohort, were submitted.

The paper (Hill and Turner, 2014a) discussed, for the first time, the implementation of Greenfoot (2015) as suggested previously (Kariyawasam, Turner and Hill, 2012). Greenfoot was introduced in the academic year 2011/2012 and discussed within this paper after 2 academic years' worth of implementation. Compared with the previous two types of robot simulation software (LMS/MRS) that had been evaluated (Turner and Hill 2007, 2008) Greenfoot was found to be a significant improvement, as Java was used as the programming language within Greenfoot. The Greenfoot application could also be simulated in the programming assignment, leading, not only, to an increased level of complexity, but the emulation of a commercially available application (previously the main application was a pseudo-real application designed by the author). See Figure 2.11 and 2.12 which show screenshots of the official Greenfoot application and scenario and the simulated student assignment version, respectively.

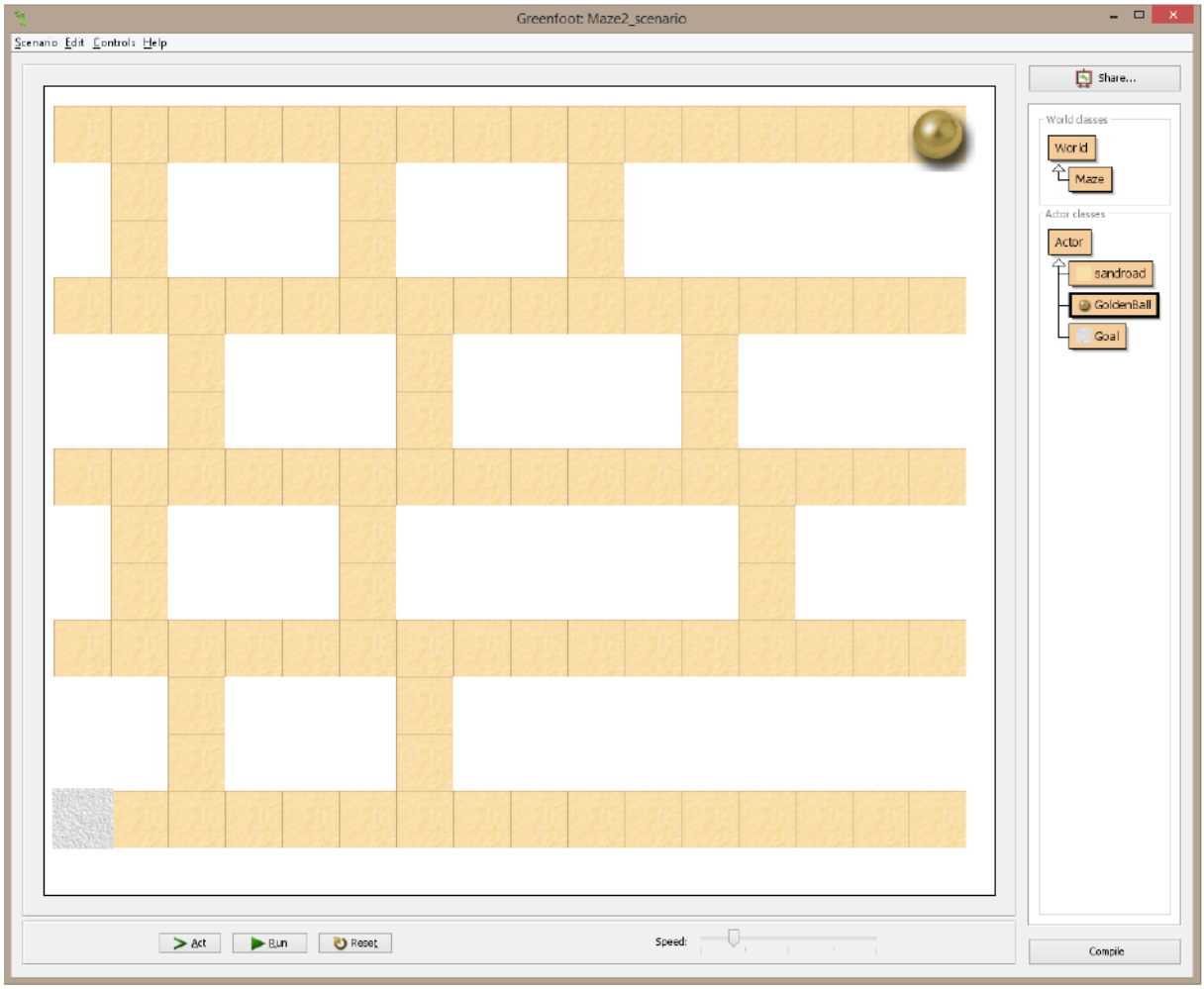


Figure 2.11: Greenfoot Application with Golden Ball Maze Scenario.

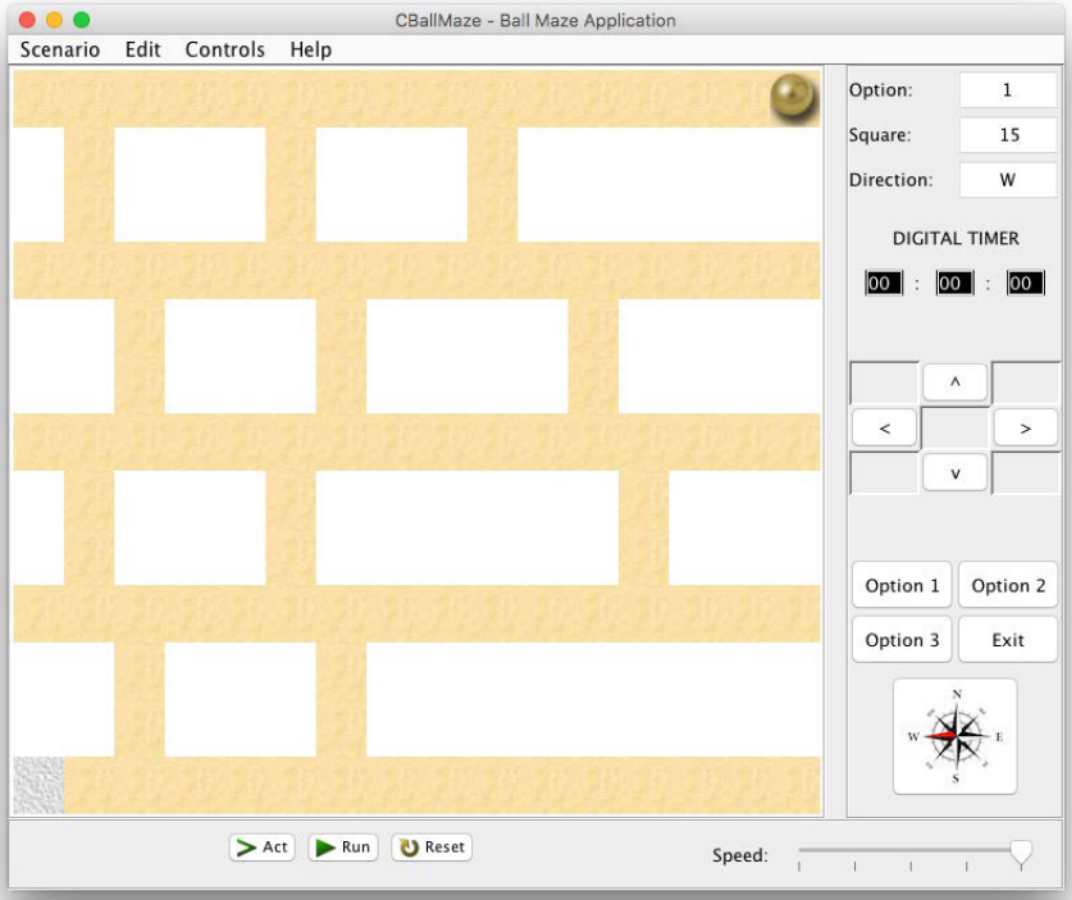


Figure 2.12: Student Emulated Greenfoot Application with Golden Ball Maze Scenario (Assignment 2: Academic Year 2017/18)

The increase and improvement in complexity was due to the additional functionality introduced by Greenfoot, where there were 'Reset', 'Act' and 'Run' buttons and resulting functionality already built-in which were to be emulated. Greenfoot has continued to be successfully used right up until academic year 2017/18. There was also the ability to share some of the application images (assets) for the applications user interface design and even some of the code between the two assignments could be repurposed and adapted. The successful use of the simulation software (Greenfoot) solved the issue of the limited physical robots (Kariyawasam, Turner and Hill, 2012) although physical robots continued to be used, but not explicitly with the assignments. Finally, the author noted that whilst the students had covered problem solving and PSL within the first section of the CSY1020 module, then progressed to the visual PbBL and visual PjBL approaches the students seemed reluctant to sit

back and sketch out or think through any of the problems they faced, instead they attempted to find the solution on line, or by typing in every permutation of computer code into their computer programme (cf. 4.4). This became the subject of a later paper (Hill, Turner, Childs, 2017) entitled *'The answer's not on the screen'*.

In 2015, the opportunity arose to submit a peer reviewed paper to offer a review of a problems-first approach to first year undergraduate computer programming (Hill, 2016). The short paper attempted to review the approach adopted by the author since 2001, of problems-first and visual computer programming to first year undergraduates. This paper included the first explicit definition of PjBL, from the author, as 'where a project/challenge is set from the outset, such that one PbBL activity leads to another and the series of linked problems form the greater challenge or project' and also linked this to Savin-Baden (2004) and their suggested 'integrated curriculum' where one problem builds upon another. The paper also suggested that active learning took place within the module. The paper also highlighted that the Institute of Electrical and Electronics Engineers (IEEE) and Association for Computing Machinery (ACM) (IEEE/ACM, 2001) recognised the 'weakness of programming first approaches'. IEEE suggested another approach could be 'algorithms and hardware first', as opposed to the 'Problems-first' and "Graphics-first' as recommended by the author (cf. 5.2.3).

The suggested/integrated approach adopted (Hill, 2016) was explained and summarised for the first time as being subdivided into **four** core pedagogical approaches in Figure 2.13.

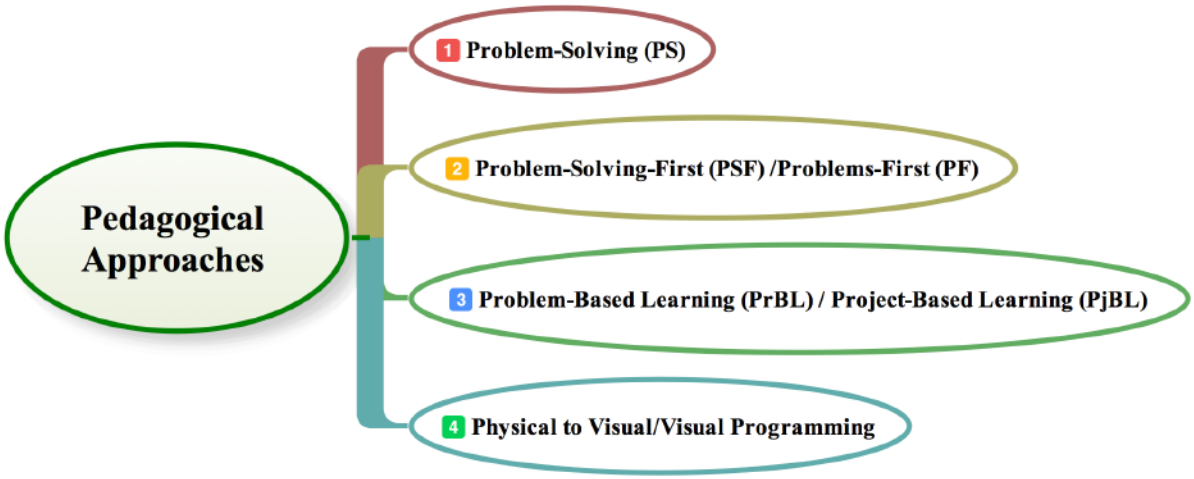


Figure 2.13: Four core pedagogical approaches.

This paper (Hill, 2016) in addition to the three requests for the authors findings (Turner and Hill, 2008, Hill and Turner, 2011, Hill and Turner, 2014a) demonstrates the interest this work has stimulated. The 2016 paper additionally referenced several authors that have cited this work, confirming its impact and broad reach.

2.5 Conclusion of Visual Problem Based Learning (PbBL) and Visual Project Based Learning (PjBL).

The previous two sections (cf.2.3 and 2.4) explicitly introduced and discussed chronologically the published works that coherently demonstrated the evolution and evaluation of the proposed improvements to first year undergraduate visual computer programming, to enhance student learning. Section 2.3.and 2.4 both discussed the chronological development of the papers demonstrating that the author's initial publications advocated a problem solving or problems-first approach was felt by the author to be an enhancement to teaching computer programming. This approach was then articulated, enhanced and evolved such that it was felt by the author to be important to continue the 'problems' approach through the whole teaching of this subject i.e. 'Problems first, second and third' which reflected the view that problems-first and progression to PbBL and PjBL approaches were advocated by the author. In addition, the author also advocated that computer programming always requires problem solving skills and computational thinking. The discussion of the papers also demonstrated the importance of the visual approach to both problem solving and the computer programming adopted throughout.

Previously (cf. 2.2) Figure 2.1 graphically illustrated the chronology/evolution of the published works in relation to the keywords/concepts introduced. The following Figures (Figure 2.14 and 2.15) re-emphasise the same keywords and concepts by using a 'keywords' density view of all the published works. Figure 2.14 emphasises the keyword/concepts by the size of the circle next to the keyword, with the thickness of the interconnecting lines emphasising the strength of the relationship. Figure 2.15 using the same relationships but illustrates the outcome as a heat map. The colours used are the familiar thermal imaging rainbow palette, where red represents the highest density (hottest) in terms of brightness and

spread, with dark blue representing no relationship/density (coldest). The colour temperatures relate to the timeline (illustrated in the bottom right-hand corner) of the published works. The Figures (2.14 and 2.15) again confirms as in Figure 2.1 (cf. 2.2) the first sentence above (cf. 1.1) that: 'This critical appraisal, predominantly discusses the **teaching** of computer **programming** and **problem** solving to undergraduate first year computing students…' (cf. 1.1).

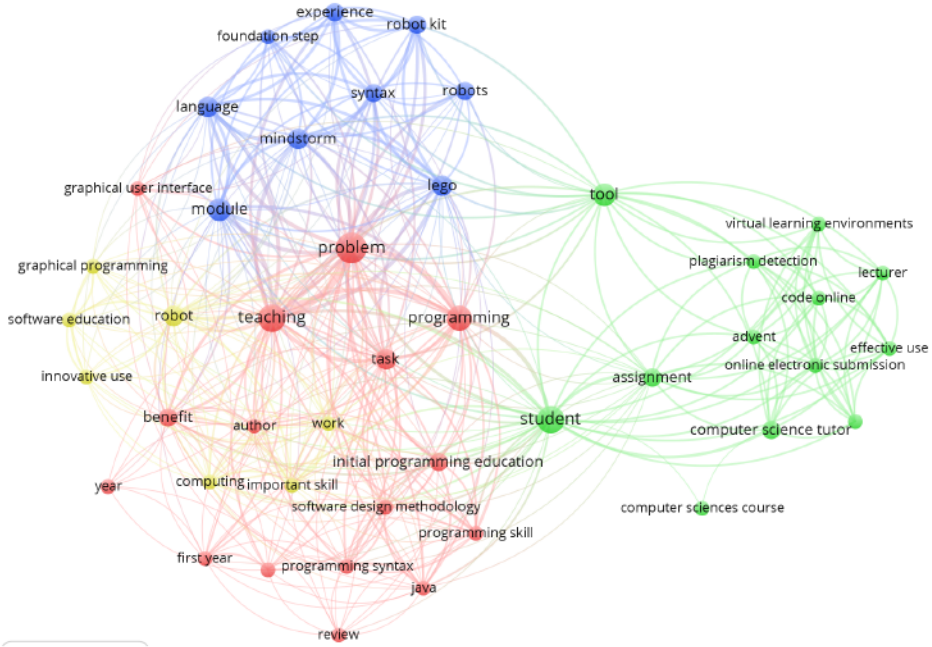


Figure 2.14: Bibliographic Research (Keyword Density) taken from NECTAR (2016) using VOSviewer (2016).

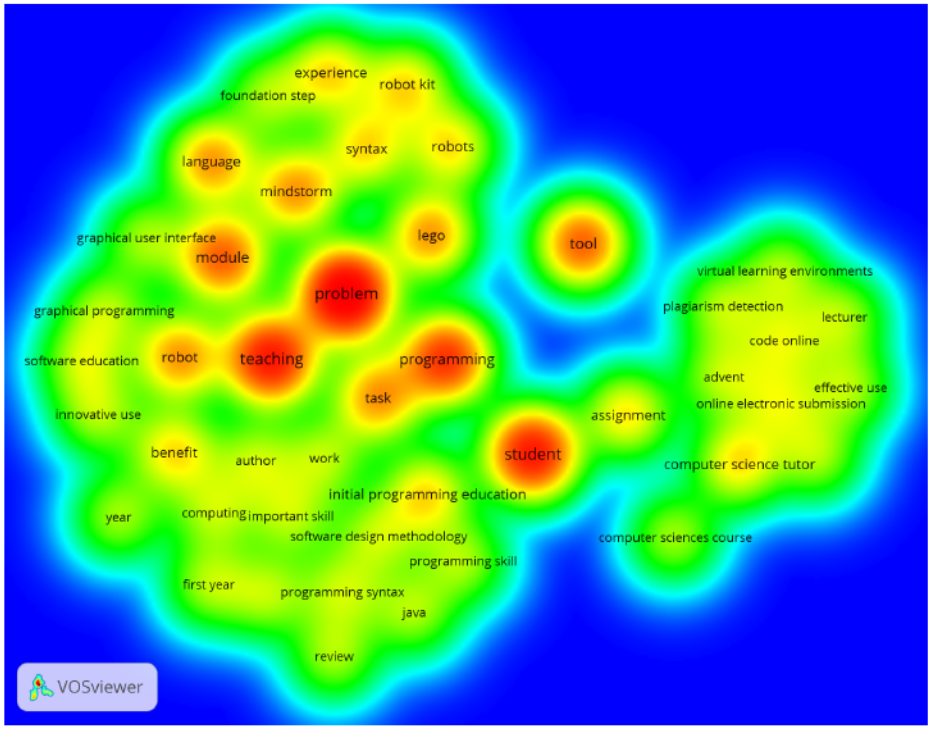


Figure 2.15: Bibliographic Research (Keyword Density – Heat map) taken from NECTAR (2016) using VOSviewer (2016).

2.6 Background to the research.

2.6.1 Introduction to the rationale for a new computer programming module.

Previously (cf. 2.1) the key terms and concepts were defined: Problem Solving (PS) and Problem Solving Learning (PSL) (cf. 2.1.1) Problem Based Learning (PbBL) (cf. 2.1.2), Project Based Learning (PjBL) (cf. 2.1.3) and Computational Thinking (cf. 2.1.4). The aim here and in the next section (cf. 2.6.2) is to locate this research in a broader educational context beyond the explicit course and subject discipline where this research has been applied.

The following section (cf. 2.6.2) also highlights the two-way transferability of the approach adopted, between various levels of educational learning and subject disciplines e.g. Higher Education to/from Schools; Computing to/from Engineering (cf. 2.1.3).

It is felt by the author that, prior to 2000, University Computing departments had numerous discussions about which computer programming language should be taught first (cf. 7.3) instead of how best to teach a computer programming language. This concurs with a practice advocated by Papert (1993, p156) of not divorcing the study of how a subject is learned e.g. computer programming, from the study of computer programming itself. In 2001, based on a conference discussion (JICC5, 2001) (cf. 4.2.1) and the suggestions from a joint IEEE and ACM task force (IEEE/ACM, 2001, p28) it was suggested that it may be prudent to introduce problem solving techniques prior to and in isolation from computer programming, in an attempt to improve the learning of computer programming.

The solution and improvements to the teaching and learning of computer programming could be both a '*top-down-design'* and '*bottom-up-design'* approach. The high level '*top-down-design'* could be viewed as being the recognition/admission that there was an issue with computer programming in University education and that an early introduction of problem solving (learning to think) could be a solution. Therefore, a course should be designed considering the attributes a computing graduate should possess, then reverse engineer the level 6 (year 3) attributes to ensure the necessary attributes are introduced and nurtured from the outset (year 1 – level 4). *Bottom-up-design* could be considered where a module is designed and introduced with the aim of improving the learning experience of a computing graduate's first exposure to computer programming. The improvement in acquisition of computer programming skills early on a course could then ensure students reach their final year possessing the appropriate attributes e.g. skills, competencies, abilities etc. Generically, the aim would be for graduating students to be better placed to meet one of the 3 key abilities within the Framework for Higher Education Qualifications (FHEQ) level 6 descriptors:

> "critically evaluate arguments, assumptions, abstract concepts and data (that may be incomplete), to make judgements, and to frame appropriate questions to achieve a solution - or identify a range of solutions - to a problem" (Quality Assurance Agency (QAA) 2014, p26).

Fellow faculty academics within the author's University had identified issues with their perception of the level of ability of the final year students compared to, as a lecturer, their expectations of what a students' prerequisite abilities should be. This led to a view that there were concerns with the 3-year degree course design and curriculum development.

Once the author had agreed that the overarching need/aim was to ensure students reached the final year and ultimately graduated with the expected level of computer programming abilities, a '*bottom-up'* solution was designed. The aim was therefore to prepare computing students each year for the computer programming tasks/subjects ahead and ensure graduating students possessed the necessary skills expected by employers (Weisfeld, 2013). The aim was then to integrate this into the curriculum/course design i.e. from the '*bottom-up*'. The vehicle chosen was to be a '*foundation/cornerstone'* 20 credit level 4 module – subsequently called 'CSY1020 Problem Solving and Programming' (cf. 2.3) - which constituted 1/6th of the first year (level 4) studies. The main consideration was 'how' to design the module to meet the aim above.

2.6.2 Pedagogy.

In order to successfully design a new module, consideration of existing educational theory and 'how students learn' was necessary, although the author recognised that educational theory appeared to predominantly relate to school children. The author had completed a Post Graduate Certificate (PGCE) in Further and Higher Education (1986) and the critical appraisal served the purpose of reacquainting the author with the educational theory literature.

Despite a lack of literature specifically aimed at the teaching of computer programming to adults, literature on how to teach children can still be applied e.g. how Dewey's concepts about experiential learning (learning by doing) and reflection…

> "are appropriated, developed and used within adult education theory". (Miettinen, 2000, p1).

In addition, adult and professional learning via action/practitioner research was used as the methodology for this research (cf. Chapter 3).

Dewey is often quoted with the statement:

> "…. give the pupils something to do, not something to learn; and the doing is of such a nature as to demand thinking, or the intentional noting of connections; learning naturally results." (Dewey, 1916, Chapter 12).

This statement emphasises that giving students something interesting to do, should stimulate thinking and natural reflection. Dewey suggests that to 'arouse' thinking and reflection in learners, examples that occur in ordinary life should be employed, as these stimulate interest and engage activity. This applies directly to the problem and project based approach advocated within the author's work and this critical review, where students were encouraged to solve problems as part of a project that is related to a real-world problem, that ultimately aimed to stimulate deeper learning and its transferability. It was also acknowledged by the author and other lecturing colleagues within the University of Northampton (and elsewhere (JICC5, 2001 and IEEE/ACM, 2001) cf. 4.1, 4.2 and 5.2.3) that the type of students recruited to the University Undergraduate Computing courses struggled with abstraction and preferred kinaesthetic learning (i.e. physical activities and active learning). One of the few authors to explicitly mention learning computing programming, Jenkins (2002) suggested even in 2002 that:

> "At the moment the way in which programming is taught and learned is fundamentally broken" (Jenkins, 2002, p53).

In the cognitive (knowledge based) domain of Bloom's Taxonomy (Bloom, 1956) an attempt is made to classify educational objectives in increasing levels of complexity. These are often used to express progression of curriculum learning, activities and assessment. The usual six cognitive domain objectives, listed in increasing levels of complexity, are:

- Remembering
- Comprehension
- Application
- Analysis
- Synthesis
- Evaluation

The author suggests that the need for continuous and varied problem solving within computer programming involves the higher-level skills of evaluation, synthesis, analysis and applications and so it is, perhaps, not surprising that students often struggle in this area. So, whilst students prefer learning by doing, a graduate degree requires them to achieve higher-level thinking skills when learning computer programming. The changes made to the author's

teaching can be seen as supporting students to move out of their comfort zone and learn how to abstract and apply their learning. Therefore, learning computer programming can be seen as covering a variety of skill levels or learning styles. Jenkins (2002) reinforces this view when he suggests:

> "It is not sensible to memorise the rules of the syntax of some programming language and then to move on to apply it. This puts programming beyond the educational experience of most students; it requires a mixture of learning styles that most, if not all, of them have not had to apply before" (Jenkins, 2002).

To support this view, Beaumont and Fox (2003) continue by suggesting that:

> "problem solving is not trivial"

and

> "In order to write programs, students need to analyse problem statements, synthesise solutions and evaluate whether they meet the specification" (Beaumont and Fox, 2003, p90).

In addition, the approach adopted by the author aims to ensure students gain deeper learning through designing, creating, experimenting and exploring rather than just delivering and transmitting information (Resnick, 2013). Students are expected to think critically to not only aid deeper learning but to consider the reflection and review of their learning to be able to transfer this to real-world problems. Hence the vehicle of Problem Solving (PS), Problem Based Learning (PbBL) eventually contained within Project Based Learning (PjBL).

Piaget - writing with Inhelder (Inhelder, Piaget, 1958) is known as an epistemologist for his theory of the cognitive development (of children through to adulthood) and introduced two learning processes: assimilation (using the old to deal with the new) and; accommodation (changing the old to absorb the new). Piaget (Inhelder, Piaget, 1958) also divided the process of the cognitive development process into four stages: sensorimotor; preoperational; concrete operational and; formal operational. For each of these stages an approximate age range of a child's cognitive development was suggested with the 'formal operational' stage relating to mid-teens through to adulthood – covering students of University age. This later stage was particularly relevant to their capabilities in the development of "hypothetico-deductive reasoning"(Inhelder, Piaget, 1958) and understanding of abstract concepts. It is acknowledged, by the author, that whilst these stages can be considered as guiding templates

for how thinking develops at different stages of life, they do not necessarily reflect the only type of thinking human beings can achieve. Due to the perceived difficulties with abstraction or 'formal' learning, Papert (1993, p21) reinterprets Piaget to 'concretize' the 'formal'/abstract, which helps emphasise the importance of kinaesthetic learning and its transition to formal or abstract learning. In addition, as stated in the first sentence of this section, whilst the learning theories discussed here predominantly consider children, these theories have been used as a starting position to investigate the approaches (Problem solving; Problem-solving-first/Problems-first; Problem Based Learning (PbBL)/Project Based Learning (PjBL) and Physical to visual/visual programming) discussed, implemented and evaluated here.

Papert (1993) has succinctly defined 'Piagetian learning' as:

> "the natural, spontaneous learning of people in interaction with their environment, and contrasted it with the curriculum-driven learning characteristic of traditional schools" (Papert, p156, 1993)

and further attributed a link to Piaget for 'active learning' and 'learning to think', which further relates to Problem Solving (PS), Problem Based (PbBL) and Project Based Learning (PjBL) applications advocated within this critical review and the research contained within it.

Piaget introduced the processes and stages of learning above, but other authors (Barbe, *et al*., 1979) introduced the concept of the three learning styles/modalities, namely: Visual, Auditory and Kinaesthetic/Tactile (VAK). This concept was further extended to VARK by Fleming and Mills (1992, p138) with the introduction of 'R' for reading/writing:

- Visual (seeing)
- Aural/Auditory (hearing)
- Read/write
- Kinaesthetic/Tactile (moving/touching)

In theory, "the more senses or modalities we can activate, the more learning will take place" (Powell, 2005, p. 62). These learning styles/modalities are also inherent in the problem solving, problem based learning (PbBL) and project based learning (PjBL) applications

advocated within this research. The terms of problem solving, problem based learning (PbBL) and project based learning (PjBL) and others, were defined earlier in context of this research/critical appraisal (cf. 2.1). In addition, Piaget (1973) in the book 'To Understand is to Invent' was responsible for emphasising the importance of inventing to learn, where:

> "students are frequently found who, though mediocre in lessons of arithmetic, prove to have a comprehensive or even **inventive** spirit when the problems are posed in relation to any activity that interests them"

and continues that students:

> "show an entirely different attitude when the problem comes from a concrete situation and is related to other interests"

warning that:

> "They remain passive and often even blocked in the school situation that consists of resolving problems in the abstract" (Piaget, 1973, p98).

Martinez and Stager, as recently as 2013, recognised Piaget's impact on shaping "how teachers taught and children learned" (Martinez and Stager, 2013, p1) through the mid-eighties, in their book "Invent to learn: Making, tinkering, and engineering in the classroom".

Their book has been reviewed as being:

> "the most important book of the 21st century for anyone interested in children and learning"

and restating that

> "Children learn best by making things whether physical or virtual" (Solomon, 2013, pv).

Whilst children are, again, used as the subject of above observations, the author feels these are equally applicable to University students, such that this research and critical appraisal will emphasise and reinforce the importance of physical, virtual/visual application to student learning for computer programming (cf. 5).

Kolb (1984) having evaluated the learning models of Lewin *et al.* (1939, 1951) Dewey (1897, 1933, 1938) and Piaget (1978) succinctly defined experiential learning as:

> "the process whereby knowledge is created through the transformation of experience. Knowledge results from the combination of grasping and transforming experience" (Kolb, 1984, p41). Kolb

also proposed a learning cycle (Figure 2.16) which considered:

> "two dialectically related modes of grasping experience—Concrete Experience (CE) and Abstract Conceptualization (AC) -- and two dialectically related modes of transforming experience—Reflective Observation (RO) and Active Experimentation (AE)" (Kolb and Kolb, 2009, p5).

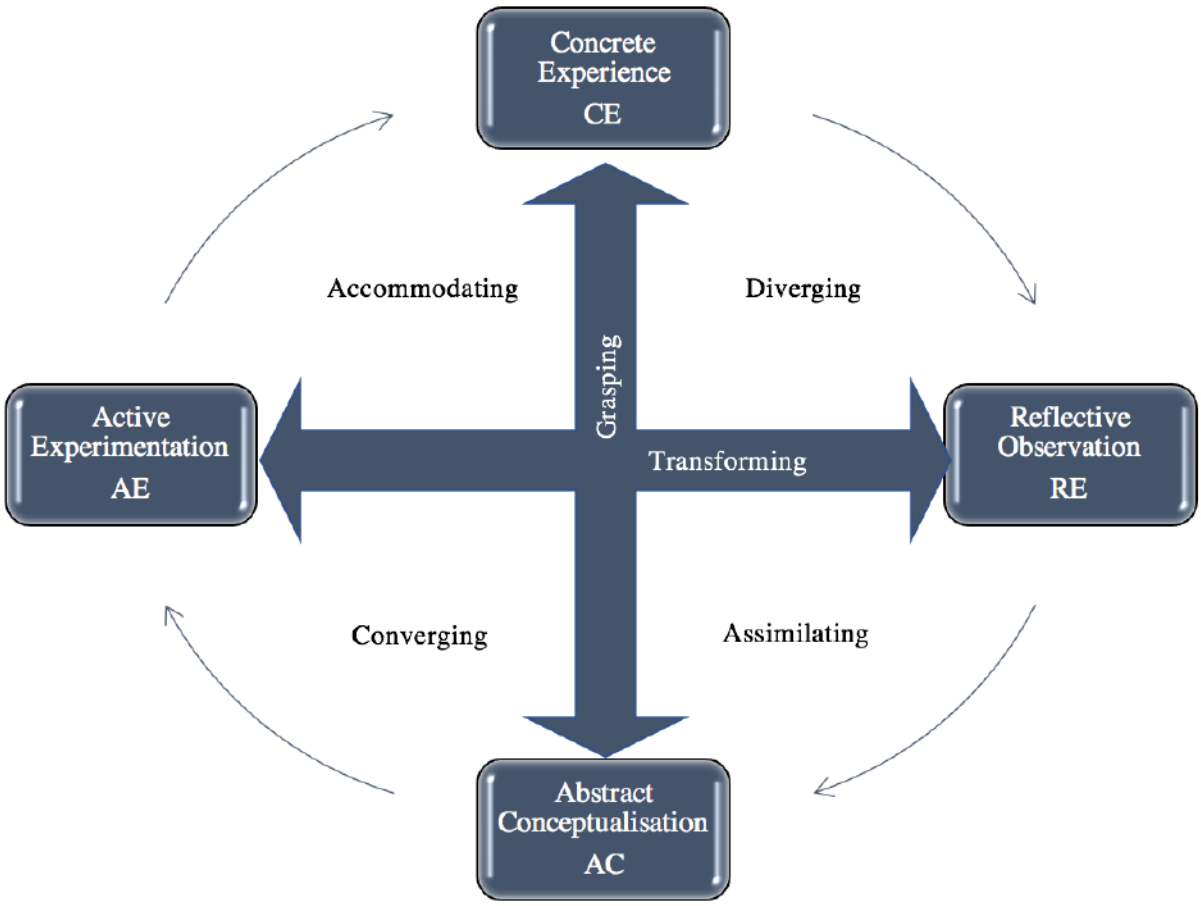


Figure 2.16: Experiential Learning Cycle (Based on: Kolb and Kolb, 2009, p6).

Kolb (1984) also suggested a model that defined the stages a learner could progress through when development and deep learning takes place due to the experiential learning cycle (Figure 2.17). It depicts the increased integration or converging of the modes of experiential learning i.e. CE, AC, RO, AE. Kolb and Kolb (2009, p5) had, in addition to the term 'cycle', used the term 'spiral' which helps to visualise this growth and development.

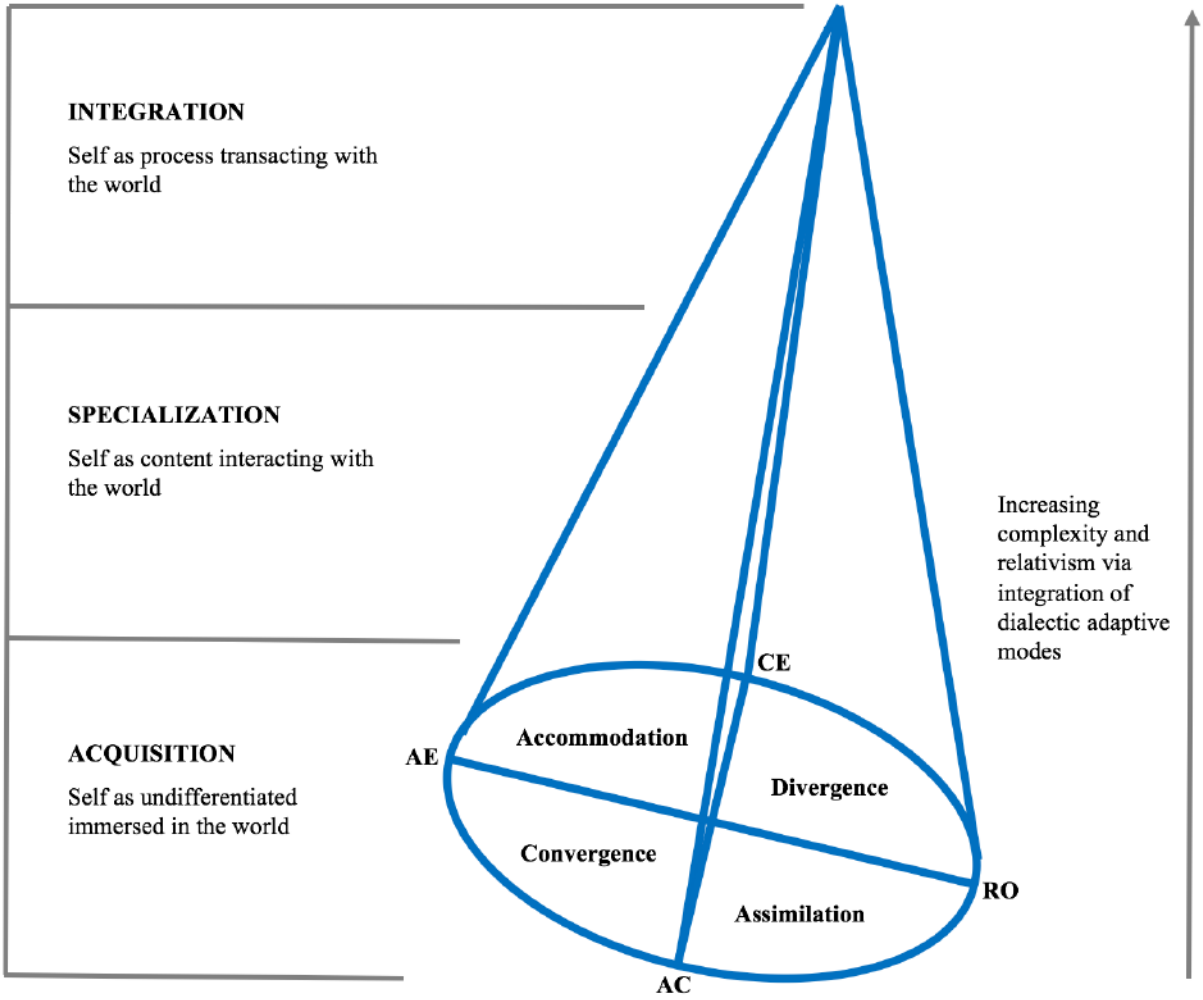


Figure 2.17: The Experiential Learning Theory of Growth and Development (Based on: Kolb and Kolb, 2009, p16).

An 'inverse' diagrammatic representation of the experiential growth and development cycle has been defined (Smith, 2015, New Zealand Ministry of Education, 2015) as the 'spiral of experiential learning' (Figure 2.18). This diagram helps to not only illustrate the growth and development of experiential learning, but its relationship to other experiences and learning contexts, ensuring that the transfer of learning/experience to similar and dissimilar learning contexts takes place.

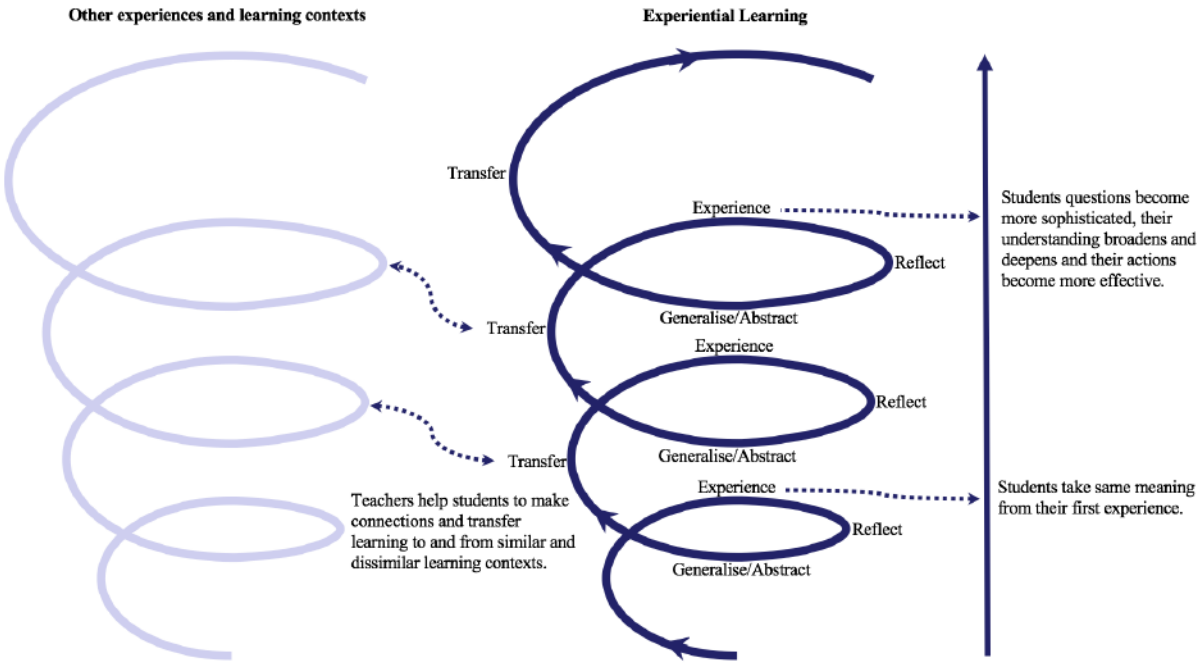


Figure 2.18: Relationship of Experiential Learning to other experiences (Based on: New Zealand Ministry of Education, 2015).

The key pedagogical approaches to learning adopted within the author's research critically reviewed within this document were defined earlier and will be discussed later i.e. Problem Solving and Problem Solving Learning (PSL) (Section 2.1.1, 4.3, 4.4) Problem Based Learning (PbBL) (Section 2.1.2, 4.6) and Project Based Learning (PjBL) (Section 2.1.3, 4.6). The pedagogical approach adopted within this research can be considered social-constructivist:

> "supporting both Papert's constructionism and Piaget's constructivism" (Caldwell and Smith, 2016, p4).

Where learning happens within a community of practice, as learners:

> "construct their own understanding of the topic, assisted by suitable scaffolding from the teacher" (Caldwell and Smith, 2016, p4)

and the lecturers

> "instead of being the 'sage on the stage' functions as a 'guide on the side' facilitating learning in less directive ways" (King, 1993, p30, cf. 2.1.2).

Whilst, Piaget has been attributed with exploring:

> "children's abilities to form concepts entirely as a result of their own thinking, i.e. in the absence of social interaction ('spontaneous concept' formation)" (NTRP, 2003, p6)….

Vygotsky (1978) advocated concepts which developed in the child as a result of social constructivism.

A couple of key elements associated with Vygotsky and linked to the research presented in this critical appraisal are:

1 the idea of social constructivism and the 'zone of potential (or proximal) development' (zpd) where: "learning could go beyond what the child showed in tests into their zpd, involving a higher stage of cognitive development which a learner could reach in collaboration with a more knowledgeable person, such as a teacher" (NTRP, 2003, p2) and;

2 the "transferability of thinking processes from one context to another" (NTRP, 2003, p2)

Element 1 above, relates closely to the aim of Project Based Learning (PjBL) (cf. 2.1.3) where students are given an advanced and complex project, with support from the lecturer providing the 'scaffolding'. Scaffolding is fundamental to the research described in this critical appraisal, as students are supported with the typical scaffolding terms (Leat, 1998) using explanations, demonstrations and analogies to begin to assimilate appropriate concepts and skills, which in turn aids with the building of confidence and reinforcement of the key learning required. In addition, mutual support amongst the students is encouraged. This idea is further emphasised where:

> "Vygotsky and his colleagues often set children challenging tasks or introduced obstacles for them to overcome and then provided them with support to scaffold their learning" (NTRP, 2003, p11).

Element 2 above, relates closely to the aim of Problem Based Learning (PbBL) (cf. 2.1.2) where students are given tuition and support in not only developing problem solving skills, but also skills of computational thinking (cf. 2.1.4) with the aim of initially transferring and

reinforcing these skills in the computer programming part of the module (discussed in this critical analysis) but also emphasising their transferability to all other modules/subjects'. As for element 1 above,

> "Vygotsky proposed that the development of abstract thinking and other higher order skills forms a unified process, which runs through all subject areas" (NTRP, 2003, p8).

Donaldson (2014) supports the pedagogic views of those above by suggesting that:

> "Constructionism brings creativity, tinkering, exploring, building, and presentation to the forefront of the learning process" (Donaldson, 2014).

The author shares the same experience of Donaldson (2014) when he states:

> "Imagine my surprise and joy when I realized that I had arrived at constructionism prior to knowing that such a theory even existed. I believe that thousands of other educators are unknowingly working within the constructionist paradigm as well" Donaldson (2014).

Figure 2.19 succinctly illustrates the main pedagogic approaches and their advocates which closely relate to this critical appraisal and the published works. Figure 2.19 illustrates the Project Based Learning (PjBL) and places Problem Based Learning as a subset, together with learning by doing, constructivism and experiential learning. Key educational theorists (as discussed above cf. 2.6.2) of Bloom, Dewey, Papert, Piaget, Vygotsky are also highlighted (as above cf. 2.6.2).

In addition to the pedagogical background and emphasis on the approach taken within this research, the importance and relevance of Problem Solving Learning (PSL) (cf. 2.1.1) Problem Based Learning (PbBL) (cf. 2.1.2) and Project Based Learning (PjBL) (cf. 2.1.3) Computational Thinking (cf. 2.1.4) are discussed later (cf. 4.3, 4.4. 4.6).

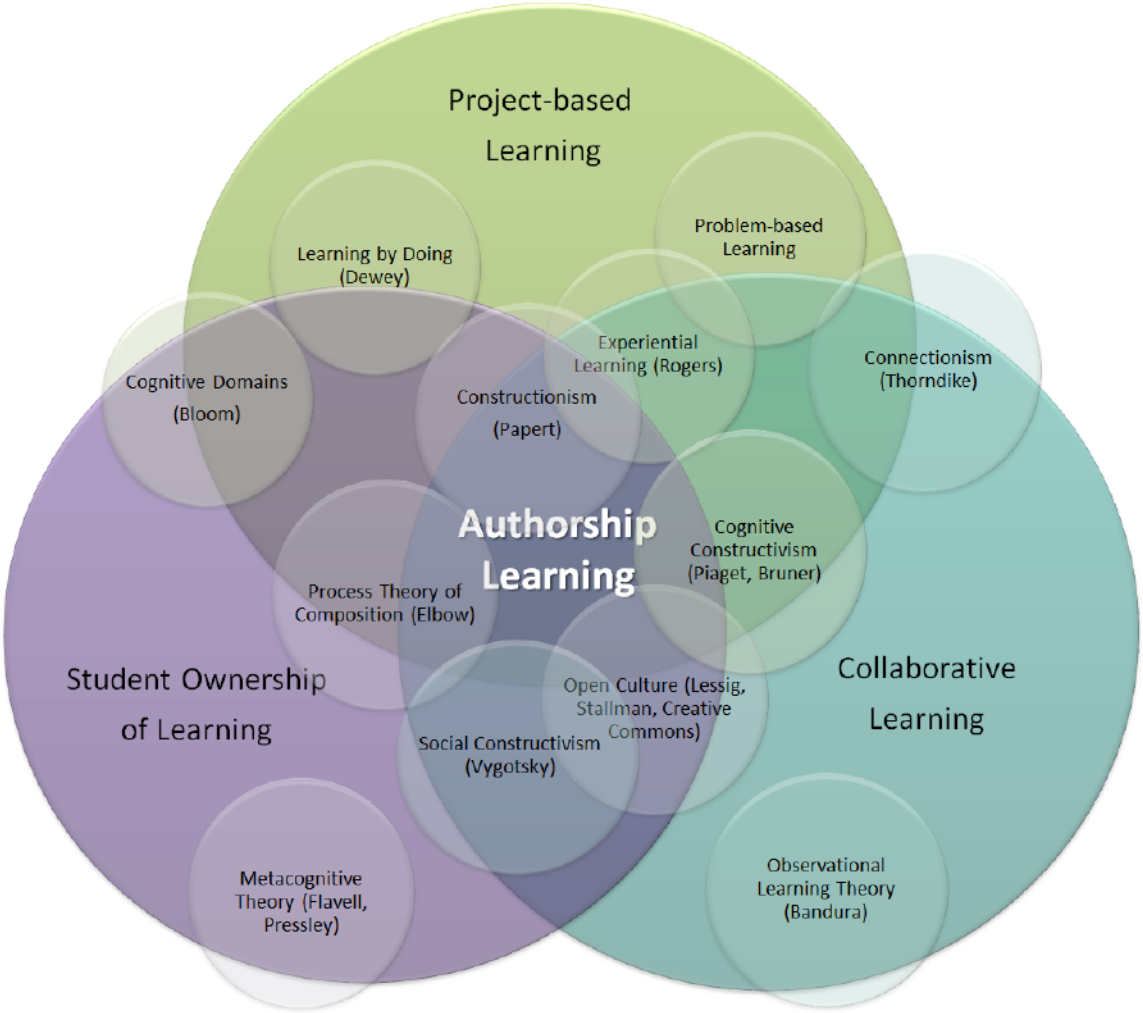


Figure 2.19: Authorship Learning (Source: Donaldson, 2014).

CHAPTER 3 METHODOLOGY.

3.1 Methods and methodology.

This chapter provides an in-depth discussion and justification of the choice of research methodology, a brief consideration of other possible methodologies, any ethics and health and safety issues and the data collection and analysis methods used.

In justifying the research methodology used, it is important to clarify that due to the nature of a PhD by means of published work, the "critical appraisal details the extent to which the author's published works provide a coherent demonstration of the following:

> "1. The creation and interpretation of new knowledge, through original research or other advanced scholarship, of a quality to satisfy peer review, extend the forefront of the discipline, and merit publication,
>
> 2. A systematic acquisition and understanding of a substantial body of knowledge which is at the forefront of an academic discipline or area of professional practice" (The University of Northampton, 2015, p6)" (cf. 1.1).

Therefore, there are two research methodologies that could be considered. One for the published work itself and the other for the critical appraisal.

There are a number of sources that discuss research methods in education (e.g. Burton and Bartlett (2009) Cohen, Manion and Morrison (2011)) as well as 'real world' research (e.g. Robson and McCartan, 2015). Cohen et al., (2011, p217-374) describe nine different styles of educational research methodologies/approaches:

- Naturalistic, qualitative and ethnographic research
- Historical and Documentary research in education
- Surveys, longitudinal, cross-sectional and trend studies
- Case studies
- Ex post facto research
- Experiments, quasi-experiments, single-case research and internet-based experiments
- Meta-analysis, research syntheses and systematic reviews
- Action research
- Virtual methods in educational research

In considering the nine styles, it is action research that is considered, by the author, as the most appropriate due to the suggested areas of:

- *"teaching methods*: replacing a traditional method by a discovery method;
- *learning strategies*: adopting an integrated approach to learning in preference to a single-subject style of teaching and learning;
- *evaluative procedures*: improving one's methods of continuous assessment;
- *attitudes and values*: encouraging more positive attitudes to work, or modifying pupils' value systems with regard to some aspect of life;
- *continuing professional development of teachers*: improving teaching skills, developing new methods of learning, increasing powers of analysis, of heightening self-awareness;
- *management and control*: the gradual introduction of the techniques of behaviour modification" (Cohen et al., 2011, p344).

The author feels that five of the six attributes suggested above articulate explicitly the areas that are covered by the authors published work and therefore the research methodology applied.

Burton and Bartlett (2009, p4) succinctly define the types of research as:

> "Pure or basic research
>
> Concerned with the development of theory and the discovery of fundamental facts to extend the boundaries of knowledge.
>
> Applied or field research
>
> The application of new knowledge to everyday problems. Though more practical it usually employs the same rigorous methodology as pure research.
>
> Action research
>
> Research into specific practical situations carried out by practitioners to solve clearly identified problems in order to improve them. As such it is continuous and cyclical.
>
> Evaluation research
>
> This is carried out to assess the effectiveness of specific projects to see if the original aims have been achieved. Many government-funded projects will allocate a proportion of their budgets for evaluation". Burton and Bartlett, 2009, p4).

In defining the methods and methodology used in this research and the published works, the author considers, in retrospect, that the methodology used in the development of the published works was *'action/practitioner research'*. Action/practitioner research, as defined by Burton and Bartlett (2009, p4) best describes the methodology used as the authors research related to a specific practical situation i.e. the enhancements to learning undergraduate computer programming, and was carried out by the author as a practitioner i.e. University lecturer/module leader to solve the clearly identified problems (see Table 3.1 for the authors academic and professional development experience as a practitioner, to illustrate how the authors professional development has informed the published works). The proposed solution or pedagogical approach proposed and adopted was the use of visual Problem Based Learning and visual Project Based Learning. In addition, due to the annual cycle of the CSY1020 Problem Solving and Programming module, delivered at the University of Northampton (UoN) the research was continuously evaluated, since 2004, throughout the annual cycle.

1985	B.Sc. (Hons) Civil Engineering (Numerical Analysis) Swansea University.
1986	P.G.C.E., University of Greenwich (Garnett College) London.
1988	Qualified Teacher Status DES #85 53197.
1998	M.Phil., Computer Assisted Learning - Structural Analysis & Design, Loughborough University.
2002	M.Sc., Computer Science, Anglia Ruskin University, Cambridge.
2006	Member of the Council of Professors and Heads of Computing (CPHC).
2008	MBCS Member of the British Computer Society #990244659.
2009	Northampton Branch Committee Member of the British Computer Society (BCS) Education Liaison Officer.
2009	CITP Chartered IT Professional #990244659.
2010	FBCS Fellow of the British Computer Society #990244659.
2015	SFHEA Senior Fellow of the Higher Education Academy #PR082148.

Table 3.1: Authors academic and professional experience as a practitioner.

Burton and Bartlett (2009) suggest that the term action research, often attributed to Lewin (1946) became less popular in "1980s and early 90s" as the term 'Practitioner research' became more accepted. Figure 3.1 shows a modified version of the action/practitioner research cycle promoted by Burton and Bartlett (2009, p9) indicates a move from:

1. Identify an issue,
2. Collect data on an issue,
3. Plan change in light of data,
4. Initiate change and finally,
5. Evaluate change.

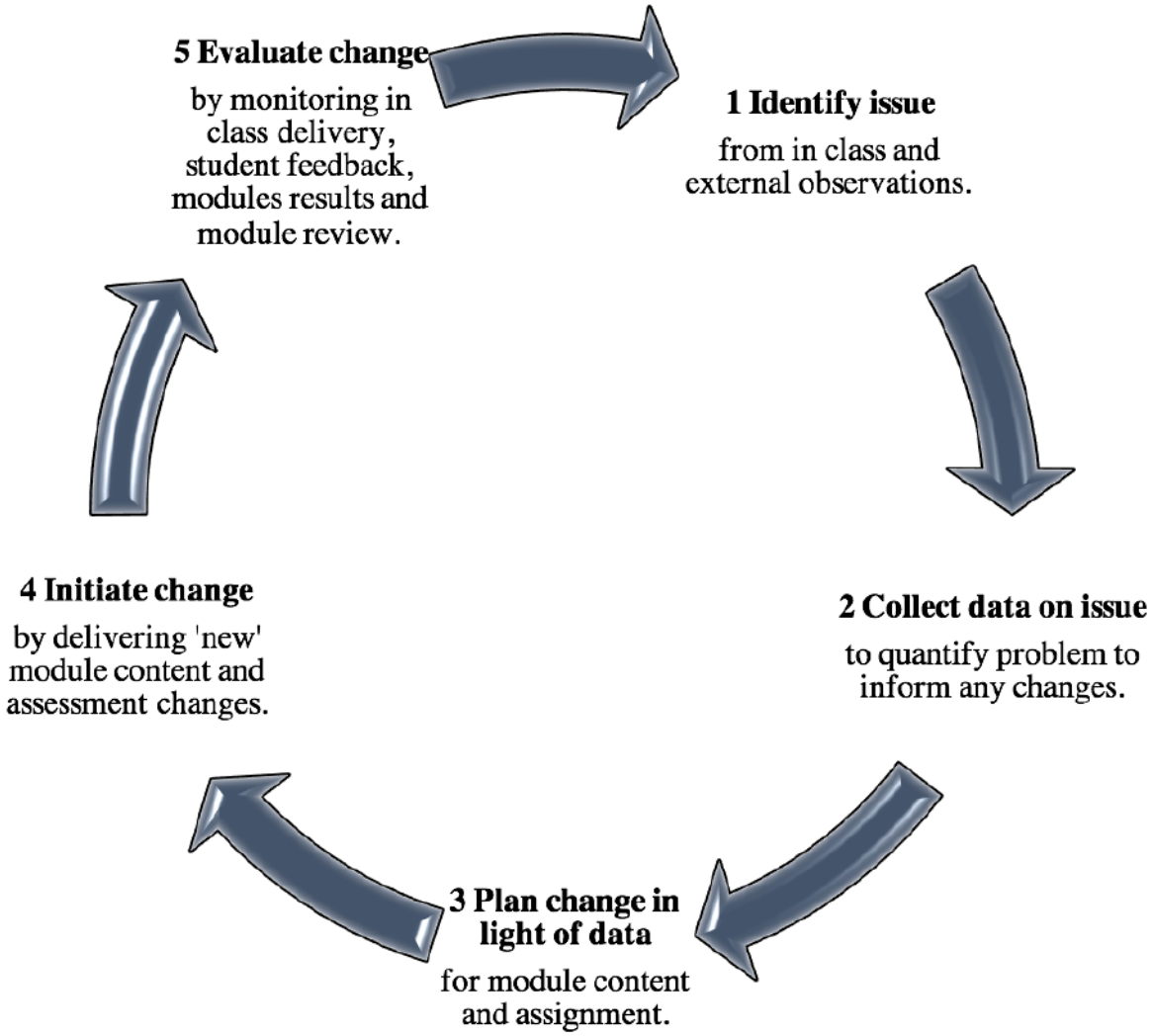


Figure 3.1: Action/practitioner research cycle for module evaluation (modified from: Burton and Bartlett, 2009, p9).

Each of these phases will be discussed further below, but these phases are in summary reflected in the original aim of this research of evaluating and proposing improvements to the teaching of first year undergraduate computer programming, to enhance student learning. This led to the author introducing new pedagogical approaches, that were introduced in 2004. These approaches were evaluated and reflected upon to enable the approaches to be further

refined, such that the approach used in 2017 has evolved through the module evaluation cycle (cf. Figure 3.1) and the journey of the author through a continuous spiral of experiential learning advocated by Kolb and Kolb (2009, p16) (cf. Figure 2.17) and hence the action/practitioner research. Therefore, the cycle of module evaluation (cf. Figure 3.1) is a continuous cycle where the iterations lead to incremental changes, small or large, introduced into the module (CSY1020) on an annual basis. This cycle has occurred annually since 2004 to 2017.

The application of the action/practitioner research to the teaching of first year undergraduate computer programming is explained as follows:

1. Identifying an issue from in-class and external observations. Initially, the identification of the issues with computer programming came from both internal and external sources (cf.2.6.1 and 4.1) e.g. University colleagues/lecturers, professional bodies, conferences and research papers. In subsequent passes around the cycle, new issues would be identified from similar internal and external sources, including the author's published works.

2. Collect data on an issue to quantify issue and inform any changes. Once the initial problem was identified, research into the teaching of computer programming, PbBL, PjBL was conducted to ascertain possible insights into good practice and issues identified by others in 'like' disciplines. Consideration of existing and new teaching strategies/theory was also researched.

3. Plan change in light of data for module content and assignment: Initially, from the first two stages, a new module was developed introducing Problem Solving and Programming and appropriate teaching strategies using visual PbBL and PjBL. In subsequent cycles these changes were made to incrementally enhance the module content and assessment.

4. Initiate change by delivering 'new' module content and assessment changes: All changes identified were implemented within the CSY1020 Problem Solving and Programming module and its associated assessments. Each cycle provided new enhancements to the assignments and module delivery.

5. Evaluate change by monitoring in-class delivery, student feedback, module results and module review. There were various methods for evaluating the module. These were via normal classroom observation by the author, whilst delivering the module and seeking informal student feedback. The module sought formal student feedback annually, that was evaluated and reviewed, together with the module and assignment results that were collected annually. There was also a requirement to conduct an annual formal module review reflecting on the operation and outcomes of the module. After subsequent cycles of evaluating the impact of the implemented changes the author disseminated the findings via the published works. The evaluation phase sought to identify positive outcomes, but also any issues that could be extracted to pass on to phase 1 above to enable the cycle to continue.

The author has referred to this 'optimisation cycle' later (cf. 6.1) by stating that the proposed learning and teaching strategies were designed, developed, implemented, evaluated and reflected upon to enable refined approaches to be further re-implemented. Figure 3.2, shows how the 5 phases of the action/practitioner research cycle (Burton and Bartlett, 2009, p9) correlate to this statement.

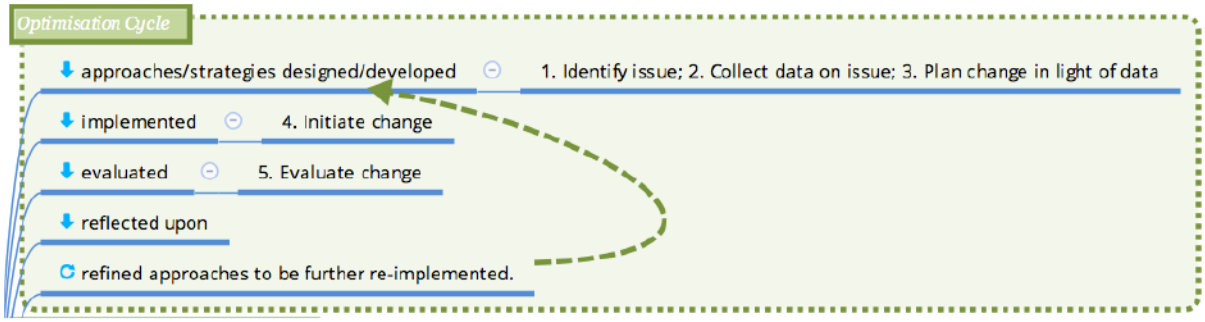


Figure 3.2: Action/practitioner research optimisation cycle.

In addition, Figure 3.3 illustrates some of the changes to the module CSY1020 Problem Solving and Programming that were considered during the action/practitioner cycle, which have been highlighted previously (cf. 2.3).

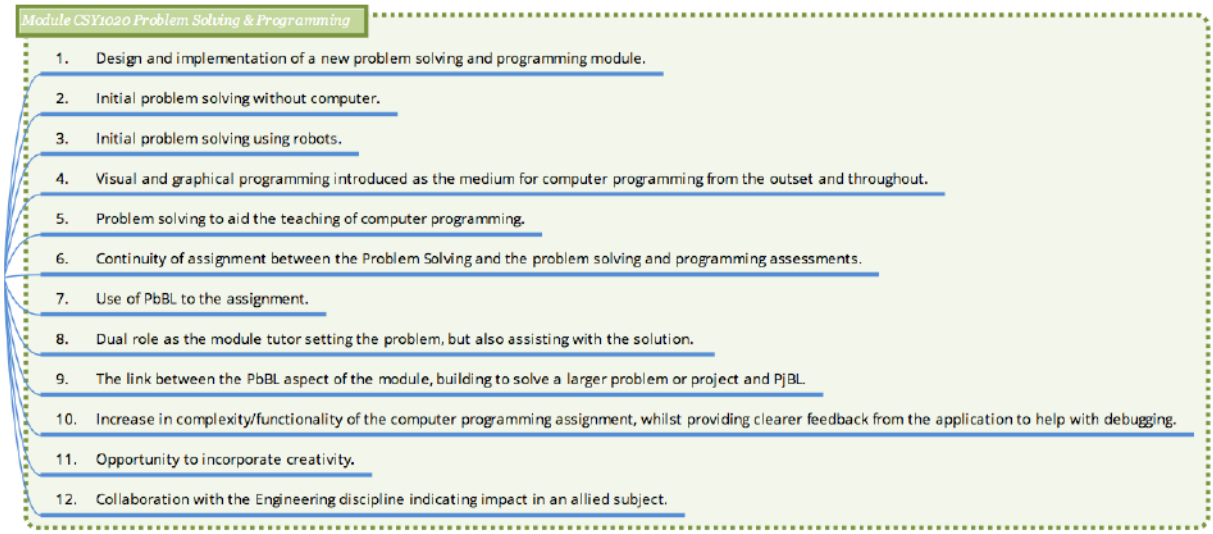


Figure 3.3: Action/practitioner research optimisation cycle.

In summary, the experience discussed within this critical appraisal, has been gained through implementing, evaluating and reviewing several evolving strategies, together with researching and publishing on this topic. Also, the evaluation of the approaches was through both peer and student feedback. The original intention was to enhance student learning of computer programming but evolved into the research and dissemination of the experiential journey via action/practitioner research.

The author considers that the practitioner research initiated and reported within this critical appraisal and the published work relates to a particular interest and concern as an experienced lecturer/practitioner, that is of value to, not only, the author, but other computing colleagues and peers, initially, within computing at the University of Northampton, but also across the UK, European and International community within computing and allied subjects. The impact of this practitioner research is made explicit within Chapter 5.

3.2 Ethical or health and safety issues.

The Faculty and University of Northampton policies and procedures relating to ethics and health and safety were adhered too throughout this research. Individuals have not been identified.

The experience has been gained through implementing, evaluating and reviewing a number of evolving strategies to enhance the student learning experience of first year undergraduate computer programming.

Predominantly, this experience has run concurrently with the normal professional conduct of a University lecturer. Feedback from the students has been sought through the student voice formally using University standard feedback questionnaires. Informal anecdotal feedback has been sought throughout as part of the normal day to day role of a module leader.

The author considered the students' perspective from research collected/collated by a student researcher under an initiative within the University (**URB@N** - Undergraduate Research Bursaries at Northampton) which was reflected in a co-authored paper (Kariyawasam, Turner and Hill, 2012, cf. 2.3, 2.4, 4.5.1 and 4.5.2). It is important to emphasise that all the student interview, questionnaire and focus group data was collected independently by the student researcher (Kariyawasam) and not the module lecturers (Hill and Turner). The aim of the study was to consider the students' view of the usefulness of a problem solving and programming module in the first year of a 3-year undergraduate program. In further investigating, after seven years, the teaching of problem solving, the study considered whether a more visual approach has benefits (the visual computer programming includes both 2-dimensional objects and graphical user interfaces (GUIs)). All students interviewed either had completed the module within the two years of the survey or were completing the problem-solving module in their first year. The methodologies adopted included interviews, questionnaires and focus groups.

3.3 Data Collection and Analysis

Data collection and analysis were primarily necessary within two of the five phases of the action/practitioner research cycle, promoted by Burton and Bartlett (2009, p9) (cf. 3.1 and Figure 3.1):

> Phase 2. **Collect data on an issue** to quantify issue and inform any changes, and;
> Phase 5. **Evaluate change** by monitoring in-class delivery, student feedback, module results and module review.

Further clarification of the data collection and analysis methods used are discussed here and that it is important to reiterate the need for the continuous cyclical nature of the collection and analysis.

The 'collect data' phase 2 was generically defined once the initial problem was identified:

i. research into the teaching of computer programming, PbBL, PjBL was conducted to ascertain possible insights into good practice and issues identified by others in 'like' disciplines (cf. 3.1). This was initially evidenced within chapter 2 where the author provided a critique of the literature surrounding the author's published research. Additionally, chapter 4 highlighted the interrelationship of the published works within the context of computing, any allied disciplines, with chapter 5 in particular (cf. 5.2) discussing the impact and confirmation that other researchers in computing concurred with aspects of PbBL and PjBL introduced by the author.
ii. Consideration of existing and new teaching strategies/theory was also researched (cf. 3.1). This was a continuation of (i) above, with the addition of 2.6.2 specifically considering existing and developing educational theory.

The 'evaluate change' phase 5 was generically defined as monitoring in-class delivery, student feedback, module results and module review. There were various methods for evaluating the module:

i) **classroom observation** by the author, whilst delivering the module and seeking informal student feedback (3.1). Due to the practical nature of computer programming, the author was always able to talk informally to students individually, whilst moving around the computer laboratory in the practical sessions. The author would not only ask questions of the students' understanding, but could see evidence of the students' progression. The students were always forthcoming with anecdotal feedback. In addition, for 2 academic years 2015-2016 and 2016-2017 a graduate teaching assistant was employed to assist with student support during timetabled practical computer laboratories. Prior to 2015 a volunteer support student from the level 5 or 6 cohort supported the CSY1020 module and since 2017 (i.e. academic year 2017-2018) a paid classroom assistant/student support student has continued to support the students. The student support since 2015, in addition to supporting the author in the practical laboratory sessions, has run a separate 1 hour per week support session. The use of the additional support has led to a positive rapport between the author/lecturer, with the

author meeting and requesting regular (weekly) continuous feedback from the supporting person and enabling a valuable insight and reassurance into the delivery of the module.

ii) **formal student feedback annually**, that was evaluated and reviewed (3.1). The University of Northampton has always sought and monitored formal student module feedback, although the form of feedback has evolved over the years to be more consistent in the type of questions asked about the module and programme across the University, as the evaluation collection method was originally programme/Faculty/School specific. This module feedback was always collated and formally included in the annual quality monitoring procedures within the University. This formal feedback was both quantitative and qualitative. Prior to the academic year 2012/2013 the module evaluation questionnaire simply asked how satisfied students were with the module (amongst all other modules on their programme – normally six modules). Responses available were Excellent, Good, Satisfactory, Below Average, Poor, N/A (I don't take this module). In addition, there was an opportunity to give qualitative feedback on: a) What positive comments would you make about the course/modules? b) In what ways could the course/modules be improved?

Since academic year 2012/13 the University of Northampton has adopted a standard module evaluation form that reflects the National Student Survey (NSS) categories of: satisfied with academic support; satisfied with feedback; satisfied with teaching and; satisfied overall. Responses are made by ticking one of five boxes with a range of options between Definitely agree to Definitely disagree. Students are also given the opportunity to offer qualitative comments about the specific module with the question: Looking back at the experience so far, are there any particular aspects you would like to highlight? With the option to complete a Positive or Areas for improvement section (see figure 3.3 below for 2 examples).

iii) **module and assignment results** that were collected annually (3.1). The module and assignment results are published at the end of the year with the statistics for the previous 3 years. The information available is the percentage of passes, percentage of passes at A&B, average pass grade and percentage non-submission.

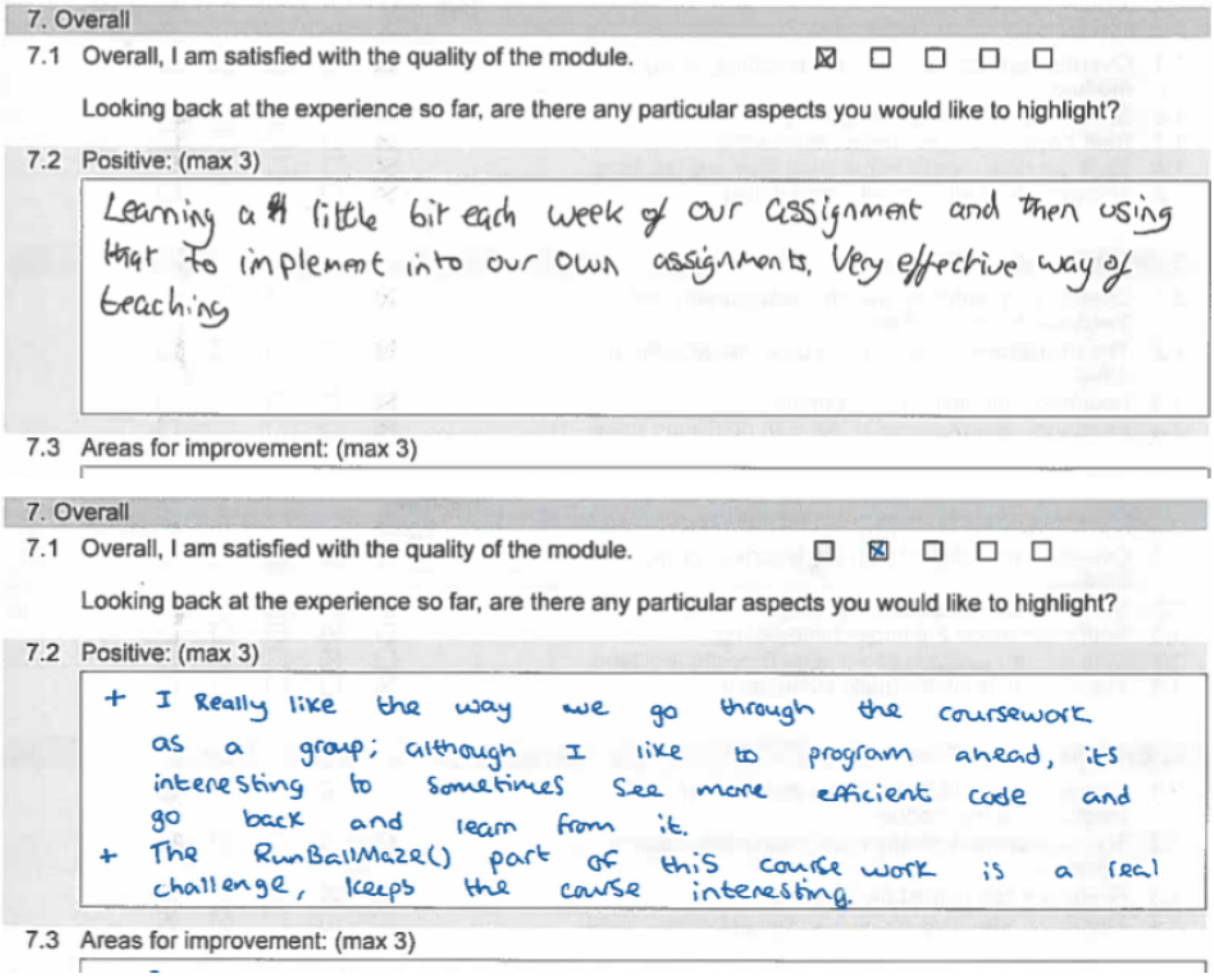
iv) **annual formal module review** reflecting on the operation and outcomes of the module" (3.1). The University of Northampton require annual module review forms to

be completed by every module leader. The annual module review provides a reflective trend analysis of the module under ten headings.

- D1. Student enrolment
- D2. Student achievement (including pass rates, non-submissions and spread of assessment grades – A's & B's and average grades)
- D3. External Examiner commentary (module specific points only) discussions, ratings and communications with students and staff
- D4. Student feedback and evaluations and student complaints
- D5. Internal and external reviews
- D6. Learning and Teaching and Assessment Strategy
- D7. Equality and Diversity
- D8 Public Information
- D9. Other issues, successes or good practice not covered above which have had an impact (e.g. staffing matters)
- D10. Commentary on matters relating to Education with Others.

v) **published works**. After subsequent cycles of evaluating the impact of the implemented changes the author disseminated the findings via the published works (3.1). The process of publishing the work provided a number of opportunities for feedback, initially from peer reviewers when submitting a paper, then peer feedback at conference presentations and finally papers that acknowledge and cite the authors published work.

The evaluation phase 5 sought to identify positive outcomes, but also any issues that could be extracted to pass on to phase 1 above to enable the cycle to continue" (3.1).


7. Overall

7.1 Overall, I am satisfied with the quality of the module. ☒ ☐ ☐ ☐ ☐

Looking back at the experience so far, are there any particular aspects you would like to highlight?

7.2 Positive: (max 3)

Learning a little bit each week of our assignment and then using that to implement into our own assignments. Very effective way of teaching

7.3 Areas for improvement: (max 3)

7. Overall

7.1 Overall, I am satisfied with the quality of the module. ☐ ☒ ☐ ☐ ☐

Looking back at the experience so far, are there any particular aspects you would like to highlight?

7.2 Positive: (max 3)

+ I Really like the way we go through the coursework as a group; although I like to program ahead, its interesting to sometimes see more efficient code and go back and learn from it.
+ The RunBallMaze() part of this course work is a real challenge, keeps the course interesting.

7.3 Areas for improvement: (max 3)

Figure 3.4: Student Module Feedback (anonymous from Academic year 2014-15.

In conclusion, this chapter provided an in-depth discussion and justification of why action/practitioner research was used, together with a brief consideration of other possible methodologies available. The author is aware of some of the known issues with action/practitioner research, in that it can be said to be a little messy! Mellor (2001) writing about educational action research, in a paper entitled 'Messy Method: the unfolding story' attempted to take a "a frank look at the untidy realities of research" (Mellor, 2001, p465) and articulate a research "approach that could accommodate those 'disorderly "messy" features of the research process' (Fine & Deegan, 1996, p. 437)" (Mellor, 2001, p466).

In 2009 Cook (2009) suggested that:

> "Engaging in action research – research that can disturb both individual and communally held notions of knowledge for practice – will be messy. Investigations into the 'messy area', the interface between the known and the nearly known, between knowledge in use and tacit knowledge as yet to be useful, reveal the messy area' as a vital element for seeing, disrupting, analysing, learning, knowing and changing. It is the place where long - held views shaped by

> professional knowledge, practical judgement, experience and intuition are seen through other lenses. It is here that reframing takes place and new knowing, which has both theoretical and practical significance, arises: a 'messy turn' takes place". (Cook, 2009, p277).

The author has found the action/practitioner research to be primarily 'messy' as whilst the methodology is designed and articulated as being sequential and cyclical, this is not always the case. The research and resulting published works have evidenced the cyclical progression, but progress is not always consistently sequential. There were various strands of the pedagogical approaches that developed in parallel/concurrent to each other, but the rate of progression was rarely consistent from published work to published work or from year to year, such was the ebb and flow of the pedagogical approaches considered. Whilst, it is recognised that progression has been at a subtle or even imperceptible rate in some areas, the pedagogical approaches have been developed and progressed within the action/practitioner research cycle. Chapter 2 sought to offer a critique of the chronological overview of the progression and evolution of the authors published works, which reflect the ebb and flow as well as the introduction of new themes, whereas Chapter 5 considers the chronological and sequential overview of the progression and evolution of the pedagogical approaches themselves.

CHAPTER 4 AN EXPLORATION OF THE INTERRELATIONSHIPS BETWEEN PUBLICATIONS TO PRODUCE AN OVERARCHING SYNOPSIS OF THE RESEARCH AS ONE STUDY.

4.1 Introduction.

This chapter will illustrate the interrelationships between the published works to produce an overarching synopsis of the research as one coherent study.

As shown in Chapter 2, since 2004 the author evolved, refined and evaluated a problem-solving-first approach for first year undergraduate students (Framework for Higher Education Qualifications (FHEQ) level 4 computer programming).

The hypothesis that, 'Problem solving should be taught first as a fundamental skill and then latterly the 'visual' computer programming language can be used/taught as a tool for solving problems', originated during discussion at a Java and the Internet in the Computing Curriculum (JICC) Higher Education Academy (HEA) conference in 2001 (JICC5, 2001).

The **four** core pedagogical approaches, stated earlier in section 2.4 (cf. Figure 2.13 and 4.1) that can be drawn from the author's experience when aiming to improve first year undergraduate computer programming are:

- Problem Solving (PS) (cf.4.3).
- Problem-Solving-First (PSF) /Problems-First (PF) (cf.4.4).
- Physical to Visual/Visual Programming (cf.4.5)
- Problem Based Learning (PbBL)/Project Based Learning (PjBL) (cf.4.6).

The author's experience has been gained using a practitioner research methodology (cf.3.1) by implementing several evolving strategies (discussed within this review) together with researching and publishing on this topic.

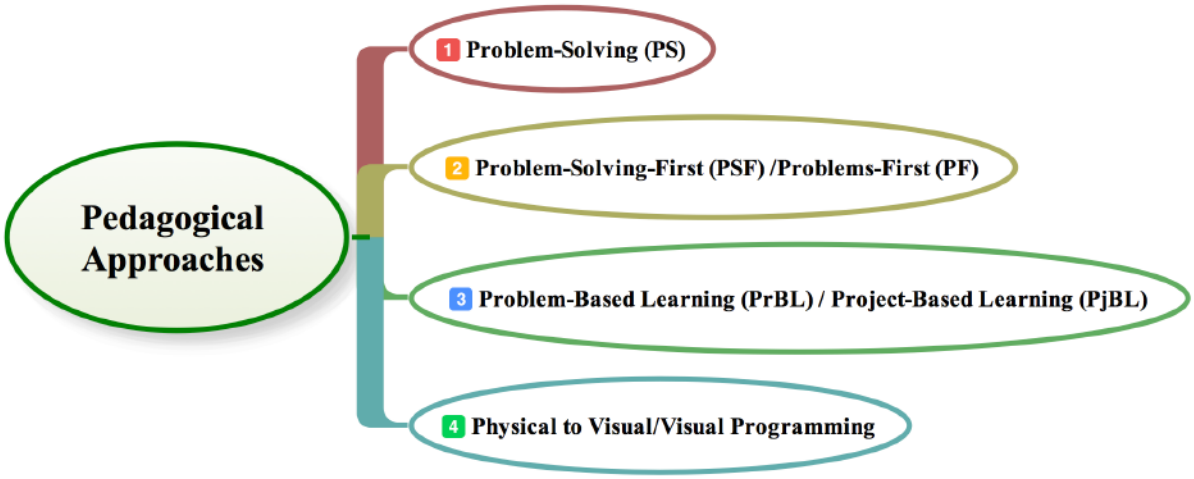


Figure 4.1: Four core pedagogical approaches (cf. Figure 2.13).

What is evidenced in the research activities below, is how the teaching of problems-first has evolved through the design of a new module with assessments designed around a professional practice project/problem based approach with a pseudo client brief (with functionality and requirements) together with clear and explicit marking criteria. The concept of the pseudo client brief was first discussed in 2011 (Hill and Turner, 2011, p121) with the assignment being set with the lecturer-as-the-client and the student-as-a-developer.

> "The module tutors adopt a client-developer relationship when discussing the project brief and executing the assignment. Throughout the assignment the relationship between the students and the lecturer is reinforced as a developer-client relationship. Specific time is allocated to the assignment discussions in an attempt to prevent any potential conflict/confusion surrounding the lecturer being the facilitator and the client" (Hill and Turner, 2014a, p99).

Explicit guidance was given to the students, including examples of feedback from previous students on where their work could have been improved.

4.2 Brief Background of the CSY1020 Problem Solving and Programming Module.

The module CSY1020 discussed (cf. 2.6.1, 2.6.2, and Appendix 8) was designed, developed and introduced in 2001. The problems first, PbBL and PjBL nature of the module has been adopted throughout the undergraduate and postgraduate curriculum, not only in Computing, but also within Engineering (Adams and Turner, 2008; Adams, Turner, Kaczmarczyk, Picton and Demian, 2008). The module is structured into two parts, eight weeks (16 hours face to face) spent on problem-solving followed by sixteen weeks (32 hours face to face) of visual computer programming in Java. The underlying approach is that as the module develops, the focus evolves from general concepts of problem solving (e.g. brain-storming, functional decomposition) to solving problems based around robots, which increase in difficulty (but not necessarily in complexity). The module is still first assessed by a robot-based project (50%) which then leads into developing the same problem via a graphical user interface (GUI) in Java, which is finally assessed (remaining 50% of module assessment). The visual nature of the subject being taught and the linkage of the assignments aid the development of the necessary skills.

4.3 Problem Solving (PS).

The following sections (cf.4.3, 4.4, 4.5 and 4.6) will illustrate the interrelationships between the published works and each of the **four** core pedagogical approaches, stated in the previous section (cf. 4.1) when related to the module (CSY1020). Whilst the key pedagogical approaches have previously been defined i.e. Problem Solving (PS) Problem Solving Learning (PSL) (Section 2.1.1) Problem Based Learning (PbBL) (Section 2.1.2) Project Based Learning (PjBL) (Section 2.1.3) it is the incorporation of these into the module that is discussed here. The approach used for the delivery of the module is multifaceted and the author continues to advocate that computational thinking (cf.2.1.4) should be integrated within these pedagogical approaches and that experiential learning takes place, whilst the theme of 'problems' and 'graphics' continues throughout the module.

The emphasis of the theme of problems or problem solving throughout, as suggested earlier by the authors (Turner and Hill, 2007) is that probably the most important skills a computer scientist or engineer must possess are those of problem solving, to the point where the role of computers as problem-solving tools is seen as increasingly useful (Gallopoulos, Houstis and Rice, 1994). These skills are highlighted in numerous benchmark and guideline statements for engineering and computing (Adams and Turner, 2008; Adams, Turner, Kaczmarczyk, Picton, and Demian, 2008; Houghton, 2004). Problem solving is not trivial (Beaumont and Fox, 2003). In fact, if one considers the cognitive domain within Bloom's Taxonomy (Bloom, 1956, cf. 2.6.2) problem solving involves the high-level skills of synthesis, evaluation, analysis and application and so it is, perhaps, not surprising that students often struggle in this area. Whilst the author has suggested that problem solving is an important skill, the author also considers that it is important that these skills are introduced as early as possible.

4.4 Problem-Solving-First (PSF)/Problems-First (PF).

The approach discussed here focuses upon the development of problem-solving skills first (Hill and Turner, 2011, p112) and not on learning a new computer programming language from the outset. Therefore, initially, any computer programming is kept simple with the minimum of commands, with *objects* unknowingly used, as these are later introduced/learned during the computer programming stage of the computing module. Having mentioned (cf. 4.3) that problem solving is not trivial and that problem solving is a major element of computer programming, it was felt by the author that students would benefit from being taught problem-solving skills first and in isolation to computer programming (IEEE/ACM, 2001, p28). In the paper Hill, Turner and Childs (2017) the authors, from the experiences gained of delivering the CSY1020 problem solving and programming module, contributed the suggestion that students too readily attempted to solve computer programming problems by blindly typing in every possible permutation of code they could think of.

Alternatively, they attempt to access appropriate solutions or information, which it can be argued is difficult for students even with the Internet, but maybe even harder because of it

(Colyer, 2013, p24) as there can be too much information and the sources are not always appropriate and academically correct. By learning problem solving skills first and being able to sit back from the computer and think through the problem was felt by the author to be the preferred approach. Students all learn in different ways and the author has repeatedly asked students to take their fingers off the keyboard and express their problem, by either walking through the issue verbally, in writing or by drawing visually. In computer programming a common, but potentially flawed, methodology for software/computer program development is known as the Waterfall model (Royce, 1970). By flawed it suggests that the software/computer programming development process is strictly sequential and non-iterative, but it is still an accepted model. The author would argue that whilst the Waterfall model is sequential, iteration is key to the model within each stage and within the whole model between stages. The Waterfall model follows the stages of:

- Systems Requirements
- Software Requirements
- Analysis
- Design
- Implementation
- Testing
- Evaluation

The stages are sequential (and iterative) but the students, when sat in front of a computer feel compelled to just implement (write the computer code) and not sit back and think of the earlier key stages – particularly the analysis and design. Normally the problem, i.e. system and software requirements are explicitly specified by the lecturer (sometimes called the problem specification) with the students needing to solve the problem by implementing/completing the computer code. Therefore, students will often jump straight to the computer coding/implementation, hence the importance of problem-solving first, which ensures that the students not only have 1) suitable problem solving skills, but 2) time to sit back and consider the analysis and design of the computer program before jumping to implementing/coding the solution. This, again (cf. 2.1.4) emphasises the importance of thinking like a computer or computational thinking.

4.5 Physical to Visual/Visual Programming.

4.5.1 Physical Robots.

This sub-section (cf.4.5.1) and the following two (cf.4.5.2, 4.5.3) summarise the transition experienced, whilst delivering the module CSY1020, of progressing from physical to the visual simulation of robots followed by the corresponding use of visual programming to emulate the physical or simulated robots. This illustrates the theme of robots used for the PbBL and PjBL assessment activities used within the module.

The authors predominantly discussed (Turner and Hill, 2006; 2007; 2008; 2010; Turner, Hill and Adams, 2009) the teaching of computer programming and problem solving to undergraduate first year computing students, using robots (Figure 4.2) and visual computer programming to emulate the robot tasks.

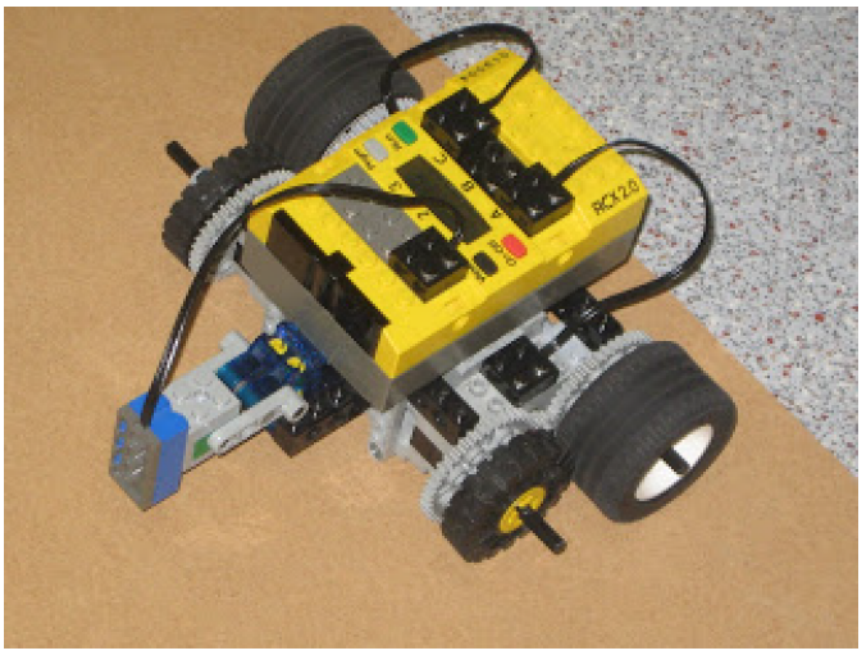


Figure 4.2: LEGO Robot used to navigate an 'M' shaped maze as in Figure 4.3 (Source: Turner).

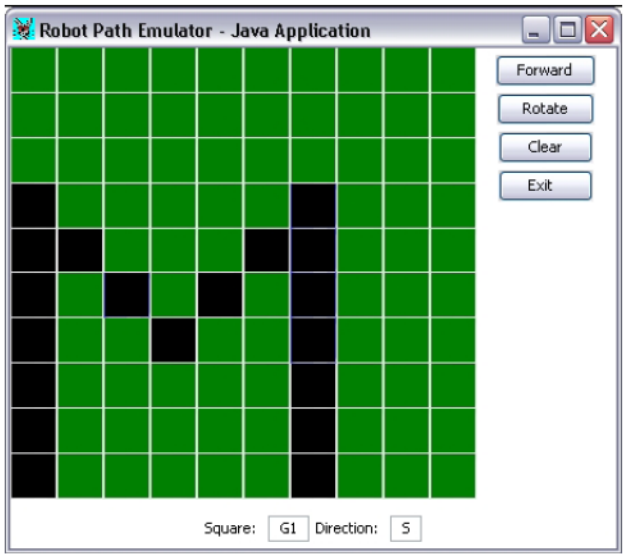


Figure 4.3: Prototype Graphical User Interface from a computer programming assignment (Source: Turner and Hill, 2007, p84).

In student interviews (Kariyawasam, Turner and Hill, 2012) two respondents reflected on their perceived benefits of using robots:

> "…it takes the concept of problem solving and places it into a physical and tangible domain…"

and

> "It's easier to understand something if you can touch it and try it in real life, rather than seeing it on a screen" (Kariyawasam, Turner and Hill, 2012, p7)

The initial emphasis was on using physical robots and this work was recognised (Gold, 2010, p11) and been cited as an example of one of the two learning models (Problem Based Learning) in a suggested framework (Štuikys, Burbaitė and Damaševičius, 2013).

4.5.2 Visual Robots.

Physical robots proved to be effective in aiding and supporting learning (cf.4.5.1) but latterly the visual robot simulators were introduced, as discussed here.

During the formal evaluation of the module (CSY1020) students had already commented (Kariyawasam, Turner and Hill, 2012) that:

> "The visual nature helped to identify errors in the programming logic and made it easier to rectify any errors made"

and

> "It helps me think through a problem if I can visualise it" (Kariyawasam, Turner and Hill, 2012, p7)

At the University of Northampton, there had also been issues with limited access to physical robots, therefore, alternative opportunities were introduced with the use of robot simulators from 2012 using Greenfoot (2015) (Figure 4.4) although Microsoft's Robotics Studio (Microsoft, 2006) had been investigated earlier (Turner and Hill, 2007). The different experiences of a visual/virtual robot against a physical robot have not been too noticeable, but the smooth transition between the problem solving and the programming section of the module has been improved due to the visual computer programming assignment partially emulating the Greenfoot Graphical User Interface (GUI).

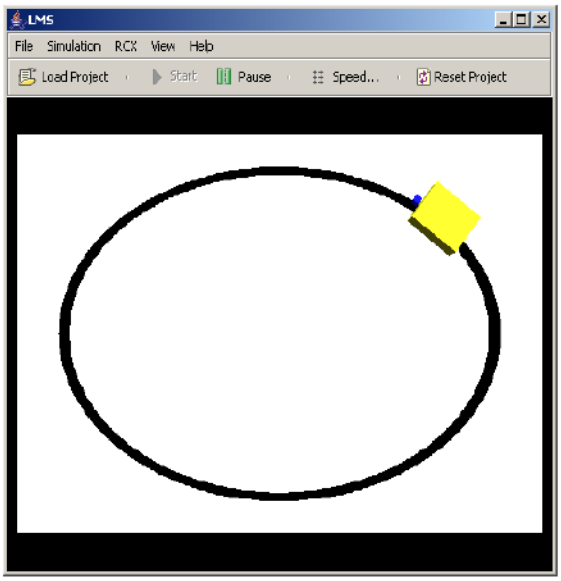


Figure 4.4: LEGO Mindstorms Simulator (LMS) – line-following robot around ellipse (Source: Turner, Hill, 2008).

4.5.3 Visual Programming.

In addition to the importance of problem-solving first, the author was adamant that, from the outset, **visual**-computer programming should be adopted – graphics-first (cf. 1.1). Examples of the output/outcome of programming non-visual and visual applications are shown in Figures 4.5. and 4.6. In addition, visual programming was used to emulate the physical or simulated robots discussed above (cf.4.5.2, 4.5.1).

Figure 4.5 shows the output as lines of text in a computer terminal/console window. The terminal/console window is an application built into the existing computers operating system that the computer code accesses to enable the displaying of the output, whereas Figure 4.6 creates 'everything' you see i.e. the Graphical User Interface (GUI) which includes the application window (cf. Figure 4.7) and all the other visual components and images contained within it. It is evident that the graphical application would be more stimulating and rewarding, to the students, when learning and developing computer programs.

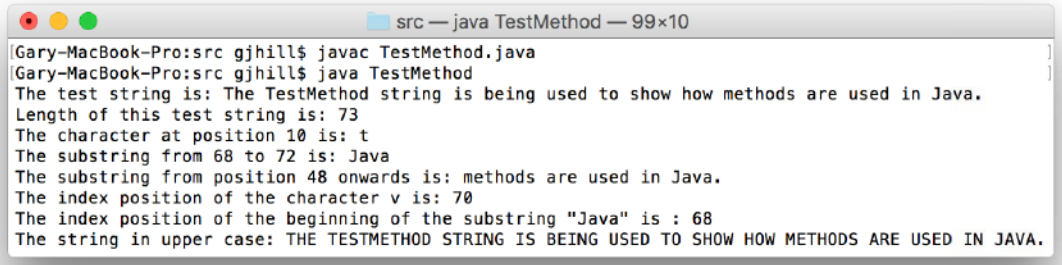


Figure 4.5: Example of a Terminal/Console Application.

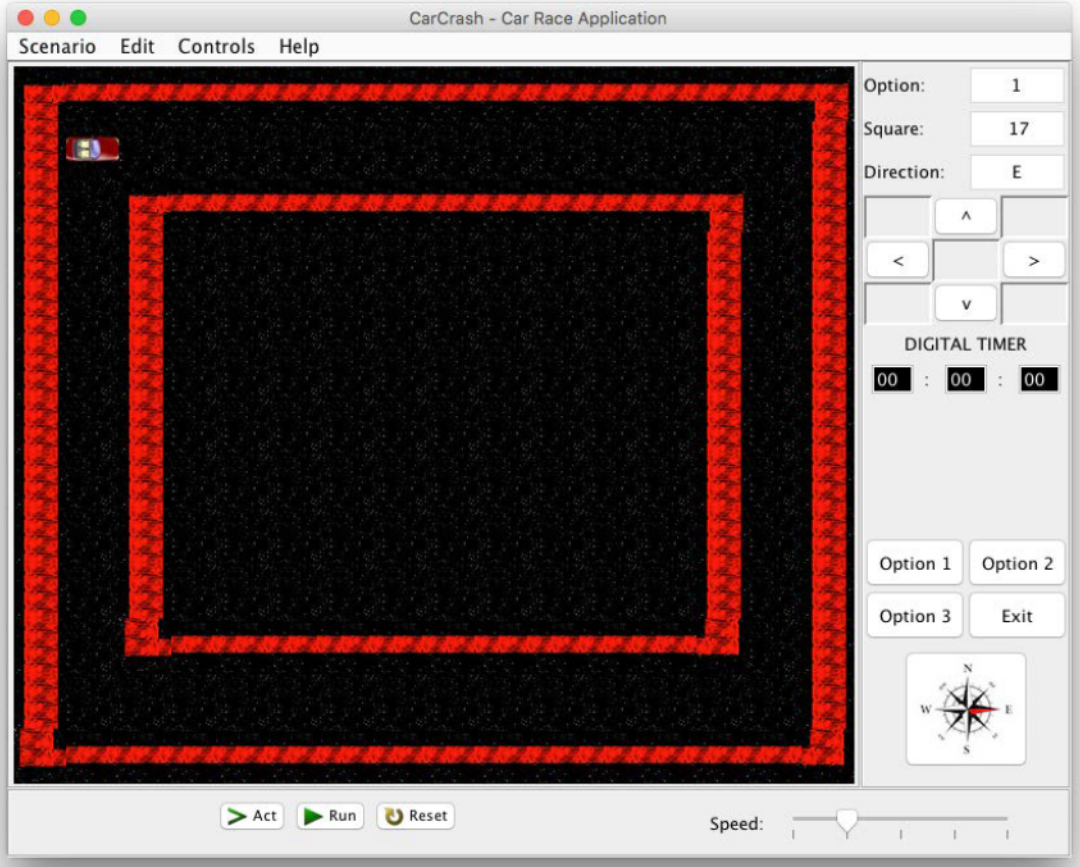


Figure 4.6: Example of a Visual/Graphical application developed by students.

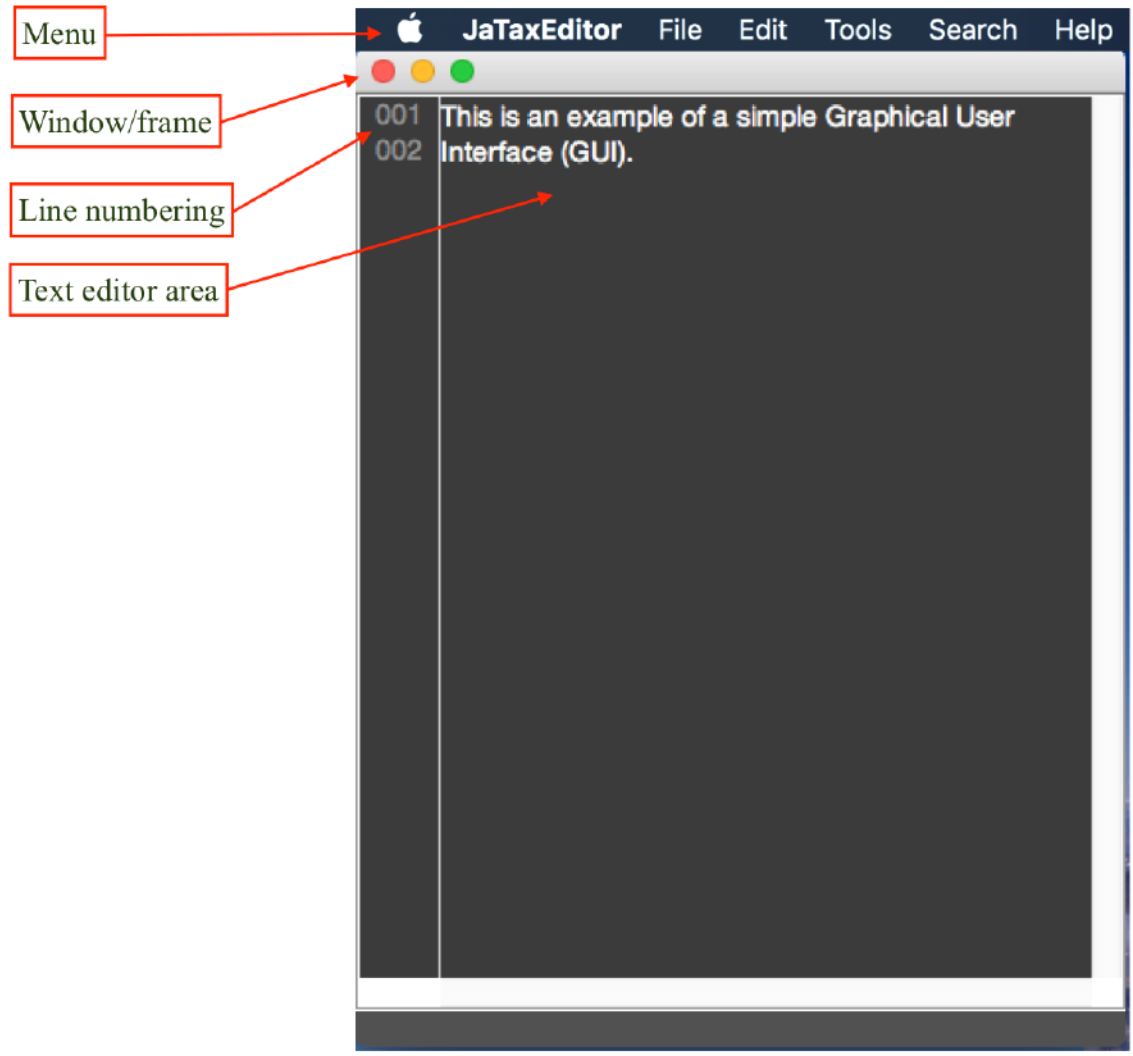


Figure 4.7: Basic Standalone Graphical User Interface (GUI) Application – JaTaxEditor.

To clarify any confusion between a Graphical User Interface (GUI) and, what the author would consider, visual/graphical programming, it is important to define a GUI as an application, as shown in Figure 4.7, where the developed computer application is a standalone application in its own 'window/frame', with a 'menu' and, in this example, a 'text editor area' with 'line numbering'.

The term visual/graphical programming should not be confused by the term 'Visual Programming Language (VPL)' where a VPL is a computer programming language that allows the development of computer programs by moving and arranging visual objects to form the logic and structure of the computer program to:

> "let users create programs through the manipulation of graphical elements, as opposed to the text edition of source code" Dehouck (2015).

A clear example of a VPL is Scratch (2017a) where a user can drag, drop and assemble blocks/jigsaw pieces to form the computer programming with the underlying computer code hidden from view (Figure 4.8).

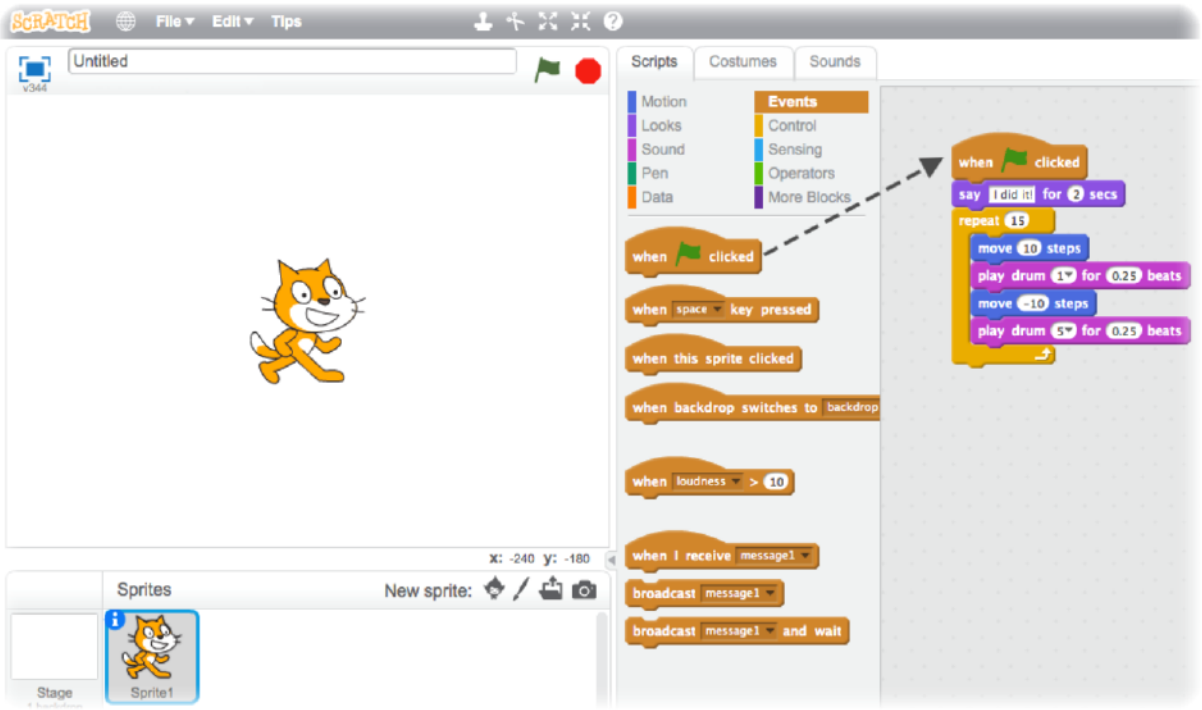


Figure 4.8: Scratch – An example of a Visual Programming Language (Source: Scratch, 2017b, p8).

This research does not include discussion around the choice of an appropriate computer programming-first language as this has been discussed and debated extensively elsewhere (Koulouri, *et al.*, 2014) within the computing community with the debate reinvigorated each time a new computer programming language appears. The author considers this to be less significant than the method of teaching of using problem solving throughout (cf. 4.4 and 4.5) using PbBL, PjBL (cf.4.6) and Visual Programming (cf. 4.5.3) as outlined in this research. The most important learning method would be experiential learning (2.6.2) with the inclusion of computational thinking (2.1.4).

However, since 2001, with advances in visual computer programming the author consciously used Java (and it's graphical and visual programming functionality) to facilitate the students in developing graphical user interfaces (GUIs) and visual computer programmes (e.g. Figure 4.3 and Figure 4.6) at the earliest possible opportunity. In fact, the students' very first computer programming exercise and introduction to the programming element of the module (CSY1020) would include the development of a simple GUI with visual content (cf. Figures 4.10, 4.11).

The module (CSY1020 Problem Solving and Programming) has been assessed by the production (and documentation) of a Java GUI/visual application that has emulated the robot problem introduced in the earlier problem-solving sessions. The author consciously used a course text (Bell and Parr, 2001) which introduced Java to students in a visual way, enabling the author to concentrate on the PbBL and PjBL aspects of the module. The book was chosen because it was considered the 'best' and most appropriate text at the time to introduce computer programming to first year undergraduate computer students. Having reviewed the course text for to the 5th (Bell and Parr, 2006) and 6th edition (Bell and Parr, 2010) the author is quoted on the back cover (Figure 4.9) of reinforcing the opinion that this is:

"the best book for my first year programming students" (Bell and Parr, 2010)

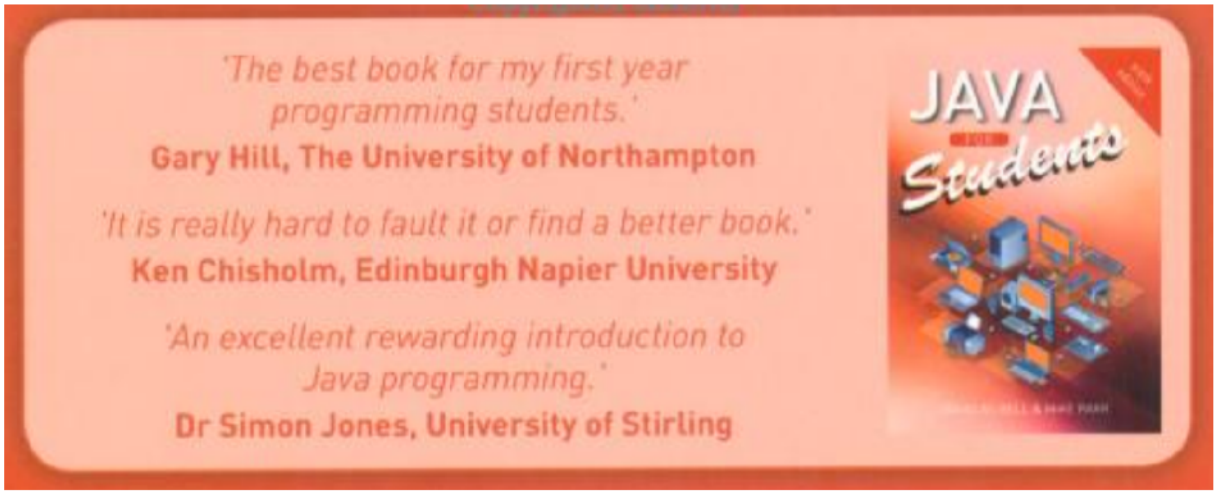


Figure 4.9: Quote by the author about the Bell and Parr, 2010, 6th Edition Java for Students book.

This is due to the focus of the book on, not only, the use of developing Graphical User Interfaces (GUIs) from the outset, but the use of visual programming/visual examples.

To emphasise and illustrate the relevance of the book to the module (CSY1020) the computer programming code and the resulting GUI application are shown in Figures 4.10 and 4.11 respectively.

```
1. import java.awt.*;
2. import java.awt.event.*;
3. import javax.swing.*;
4.
5. public class SomeShapes extends JFrame implements ActionListener
6. {
7.     private JButton button;
8.     private JPanel panel;
9.
10.    public static void main (String[] args)
11.    {
12.        SomeShapes frame = new SomeShapes();
13.        frame.setSize(400, 300);
14.        frame.createGUI();
15.        frame.setVisible(true);
16.    }
17.
18.    private void createGUI()
19.    {
20.        setDefaultCloseOperation(EXIT_ON_CLOSE);
21.        Container window = getContentPane();
22.        window.setLayout(new FlowLayout() );
23.
24.        panel = new JPanel();
25.        panel.setPreferredSize(new Dimension(300, 200));
26.        panel.setBackground(Color.white);
27.        window.add(panel);
28.
29.        button = new JButton("Press me");
30.        window.add(button);
31.        button.addActionListener(this);
32.    }
33.
34.    public void actionPerformed(ActionEvent event)
35.    {
36.        Graphics paper = panel.getGraphics();
37.        paper.drawRect(30, 30, 80, 40);
38.        paper.drawOval(130, 30, 50, 50);
39.        paper.drawOval(230, 30, 30, 50);
40.        paper.setColor(Color.lightGray);
41.        paper.fillRect(30, 100, 80, 40);
42.        paper.fillOval(130, 100, 50, 50);
43.        paper.fillOval(230, 100, 30, 50);
44.    }
45.}
```

Figure 4.10: Computer Code - SomeShapes (Bell and Parr, 2010).

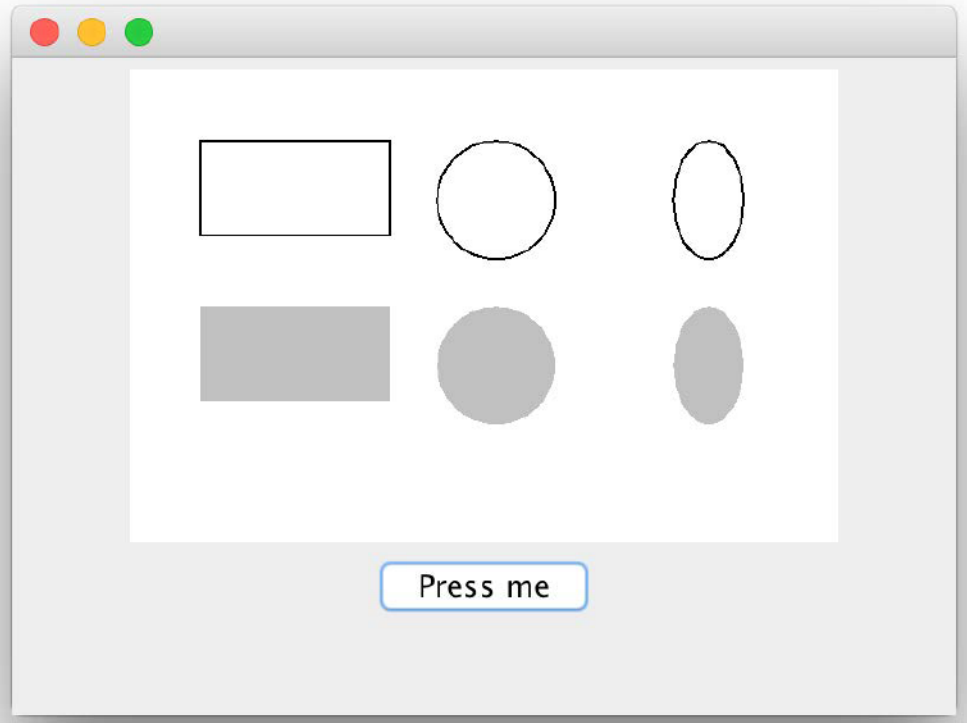


Figure 4.11 Resulting GUI - SomeShapes (Bell and Parr, 2010).

4.6 Problem Based Learning (PbBL)/ Project Based Learning (PjBL).

Previously (cf.2.1.3) the author defined PjBL as where a project/challenge is set from the outset, such that one PbBL activity leads to another and the series of linked problems form the greater challenge or project. This section explains how this approach to PbBL and PjBL was implemented within the module (CSY1020).

As in the problem-solving section of the module (CSY1020 Problem Solving and Programming) the grades, feedback and engagement with the computer programming activity/assignment were consistently positive. The idea of linking the problem-solving and

computer programming assignments, with the same task, was seen as a positive feature. One student made the comment that they felt there was a good progression from problem-solving to programming. In addition, the students commented that they could take the ideas developed in one part of the module to the second part, thus evidencing clear transferability of skills. The author considers that a simple definition of PjBL is where a project/challenge is set from the outset, such that one PbBL activity leads to another and the series of linked problems/problem based activities incrementally and cumulatively form together to achieve the greater challenge or project. Štuikys *et al.* (2013) cited the authors' work as supporting/demonstrating one of the two proposed learning models of PbBL. However, it could be argued that the other model of PjBL should also be seen to be supported/demonstrated by the author's work, but Štuikys *et al.* (2013) recognise:

> "there is a thin line among the models, nevertheless, we introduce them as slightly different teaching scenarios (in other words the models are integrated within the scenarios)" (Štuikys et al. 2013, p132).

Whether the module of Problem Solving and Programming is considered as either problem-based, project based or both, it is felt by the author to be important that active learning is seen to occur, where active learning can be defined where students:

> "must talk about what they are learning, write about it, relate it to past experiences, apply it to their daily lives. They must make what they learn part of themselves" (Chickering and Gamson, 1987, p4).

The author would consider the work (Hill and Turner, 2011; 2014a) so far to be PbBL and PjBL, to include active learning. The aspect of the Project-based module that is liked by the students is the incremental revealing/introduction of key computer programming concepts and skills (Problem Based Learning) alongside immediate reinforcement and application in the project-based visual/GUI assignment. To share a couple of quotes from the 2017 module evaluation survey:

> "Students are eased into the concept of programming, this is good so students don't develop a phobia of coding. Overall very well organised, explained and taught well."

> "Explained how every example relates to the assignment"

Each topic delivered can be seen as a self-contained lesson (learning object) with the following attributes:

a) Introduction, aim and objectives
b) Concepts introduced demonstrated (with examples)
c) Summary of the concepts (with additional formative exercises)
d) Direct discussion, consideration, reinforcement and application to the project-based assignment.

The final stage (d) not only reinforces the new concepts, but also illustrates the 'real-life' relevance to the problem. This aspect of the module can be seen as true PjBL with:

> 'integrated curriculum where one problem builds upon another…' (Savin-Baden and Wilkie, 2004, p2).

It is felt by the author that this visual PjBL approach (with the incremental reinforcement and application) to computer programming, adopted from the outset, and the transferability of the visual problem-solving to the visual computer programming task facilitates active learning. This should benefit student engagement, enjoyment, learning and, ultimately, the ability to see the relevance to software industry-oriented practices. The PjBL pedagogic approach should also be considered to include experiential learning (cf. 2.6.2).

4.7 Conclusion.

This chapter has illustrated the interrelationships between the published works to produce an overarching synopsis of the research as one coherent study. Specifically, the previous four sections (cf.4.3, 4.4, 4.5 and 4.6) illustrated the interrelationships between the published works and each of the **four** core pedagogical approaches, stated in the previous section (cf. 4.1) when related to the module (CSY1020). In summary, the module (CSY1020) evolved dramatically. Initially physical robots and problem solving techniques were introduced in the

first third of the module. This third of the module transformed during the first 5 years, away from physical robots, to include more PjBL and visual approaches, as used within the second two-thirds of the module. Instead of a balance of one-third problem solving to two-thirds visual programming and PjBL, this became closer to an 'estimated (75-80%) by the tutoring team'. The author would claim that the problem-solving part of the module was inspired by the latter part of the module, by introducing a more visual approach (Greenfoot, cf. 4.5.2 and Scratch, cf. 4.5.2) and elements of PjBL e.g. the problem-solving assignment was altered to include an assessed element of and discussion on group-work.

The following chapter (Chapter 5) will explicitly discuss the author's research work to date and its reception and resulting impact.

CHAPTER 5 IMPACT AND RECEPTION OF THE PUBLISHED WORK.

5.1 Introduction.

The impact and reception of the author's published work to date will be covered within this chapter.

Research Councils UK (RCUK) offers an overall definition of 'impact' as the:

> "demonstrable contribution that excellent research makes to society and the economy" (RCUK, 2016).

This can be achieved in a number of ways. The example that reflects the research discussed within this critical appraisal is the impact through creating and sharing new knowledge and pedagogical innovation. RCUK further define types of research impact as:

- "**Academic impact** is the demonstrable contribution that excellent social and economic research makes to scientific advances, across and within disciplines, including significant advances in understanding, method, theory and application.
- **Economic and societal impact** is the demonstrable contribution that excellent social and economic research makes to society and the economy, of benefit to individuals, organisations and nations" (RCUK, 2016).

In addition, research impact can include:

- **"Instrumental**: influencing the development of policy, **practice** or service provision, **shaping legislation**, **altering behaviour**
- **Conceptual**: contributing to the understanding of policy issues, reframing debates
- **Capacity building**: through technical and personal skill development" (ESRC, 2016).

Although:

> "Quantification of impacts numerically can be useful to demonstrate tangible, measureable benefits, but for many impacts, qualitative evidence is more appropriate…" Reed (2016, p232)

This chapter will evidence both quantitative and qualitative impacts. However, in a recent Higher Education Funding Council for England (HEFCE) paper (HEFCE, 2017) it is

recognised that there are inconsistencies in the definition of 'impact'. There is a recommendation to harmonise the existing Research Council definitions to:

> "enhance the complementarity of impact policies"

and

> "to align their definitions of 'academic impact' and 'wider impact' for the purposes of REF" (HEFCE, 2017, p7).

With the introduction of the Teaching Excellence Framework (TEF) in 2016, the impact of the author's research is timely due to its proposed pedagogical innovation, in relation to the TEF Assessment Criteria of: Teaching Quality (TQ), Learning Environment, Student Outcomes and Learning Gain (TEF, 2016, p24). The TEF declares that:

> "The emphasis in the provider submission should be on demonstrating the impact and effectiveness of teaching on the student experience and outcomes they achieve" (TEF 2016, p42)

and suggests one such example of evidence for the aspect of Teaching Quality (TQ) as the:

> "Impact and effectiveness of innovative approaches, new technology or educational research" (TEF, 2016, p44).

Before discussing specific stakeholders that have been influenced by this research (cf. 5.3) the author will highlight the academic impact on the research community via the request for papers/chapters based on earlier publications by the author.

5.2 Academic Impact.

5.2.1 Requests for papers/chapters and citations.

This sub-section and the next three (cf. 5.2.2, 5.2.3 and 5.2.4) illustrate the evolution of the topic/research by evidencing where there have been requests for further published works and citations on existing papers in the context of the selected published works. Table 5.1 illustrates where there have been requests for further published work (papers or chapters). This sub-section (cf. 5.2.1) discusses the specific requests for further published works and mentions any citations, before the following sub-sections 5.2.2, 5.2.3 and 5.2.4 relate the

citations in the context of the three areas of problem-solving using robots (cf. 5.2.2) problem-solving-first (cf. 5.2.3) PbBL and PjBL (cf. 5.2.4).

‹r Number Table 1.1)	Invitation requests (Peer review	Published work
		Turner, S., **Hill, G. J.** (2007)
		Turner, S., **Hill, G. J.** (2010)
		Kariyawasam, K. A., Turner, S., **Hill, G.** (2012
		Hill, G. J. (2016)

Table 5.1: Requests for Published Works.

There were three specific requests for further works as follows:

1. The Turner and Hill (2007) Higher Education Academy-Information and Computer Science (HEA-ICS) paper led to the authors being invited to write another paper for the HEA-ICS 'Italics' journal (Turner and Hill, 2008) (cf. 2.3).
2. The Turner and Hill (2010) paper led to direct approach to the authors from publisher IGI-Global to write a chapter (Hill and Turner, 2011) about the 'Problems-first' pedagogic approach and to summarise the experiences and findings. The chapter was published in a special edition entitled *'Software Industry-Oriented Education Practices and Curriculum Development: Experiences and Lessons'* (Hussey, Wu and Xu, Eds., 2011) (cf. 2.4).
3. The Hill and Turner (2011) paper led, three years after their first approach in 2011, to another approach from IGI-Global, to write a further update on the previous chapter (Hill and Turner, 2014a) (cf. 2.4). This paper emphasized the authors subsequent view that, whilst the problems-first approach was felt by the author to be an enhancement to

> teaching computer programming, it was considered to be important to continue the problems approach through the teaching of this subject, hence the title of 'Problems first, second and third' to reflect the problems-first and progression to PbBL and PjBL approaches advocated by the author (cf. 2.5).

The 2016 paper (Hill, 2016) gave the opportunity to review the approaches adopted by the author since 2001, of problems-first, graphics-first, visual computer programming, PbBL and PjBL to undergraduate computer programming students. This paper (Hill, 2016) in addition to the three requests for the authors findings (Turner and Hill, 2008, Hill and Turner, 2011, Hill and Turner, 2014a) demonstrates interest that the author's research into teaching has stimulated as the 2016 paper additionally referenced several authors that cited the work appraised here, confirming the academic impact and broad reach of this research. Chapter 4 (cf. 2.3, 2.4) offered a critique of the author's published work for the first eleven years (1999-2009) (cf. 2.3) and then the last eight years (2010-2017) (cf. 2.4) evidencing the requests for follow up publications.

Another tangible indicator of academic impact can be seen in the number of views received of the papers via Academia.edu (2017). Table 5.2 shows the number of all-time views of the various papers. The two most popular of the 'key' papers (cf. Table 1.1) are Turner and Hill (2007) at 1,856 and Turner and Hill (2010) at 28, respectively. Turner and Hill (2007) is the most popular paper, but is also one of the earliest papers and has been in circulation for three years longer. The paper was presented at the 8th Annual Conference of the HEA-ICS at the University of Southampton.

Publication	Title	All-Time Views
Turner, S., **Hill, G. J.** (2007)	Robots in Problem-Solving and Programming.	1,856
Hill, G., Svennevik, E., Turner, S. (2014)	Computer Science Courses Using Laptops	29
Turner, S., **Hill, G. J.** (2010)	Innovative Use of Robots and Graphical Programming In Software Education	28
Turner, S., **Hill, G. J.** (2006)	The Inclusion of Robots within the Teaching of Problem-Solving: Preliminary Results	9
Hill, G., Svennevik, E., Turner, S. (2011)	Green computer science courses. No more labs full of computers, we're going mobile!	10
Kariyawasam, K. A., Turner, S., **Hill, G.** (2012)	Is it Visual? The importance of a Problem Solving Module within a Computing course	7
Hill, G. J., Turner, S., Childs, K (2017)	The answer's not on the screen	3
Hill, G**., Turner, S. (2012)**	Referencing within Code in Software Engineering Education!	1

Table 5.2: All-Time Views from Academia.edu (2017).

Table 5.3 illustrates explicitly which papers have been cited, how many times and by which source (when not from own published works). This table (Table 5.3) has been compiled using data obtained from Google Scholar (2017) and Research Gate (2017) and indicates 12 unique citations (i.e. excluding self-citations). These citations and their relationship to the author's published work are all referenced in the following sections and sub-sections (cf. 5.2.2, 5.2.3, 5.2.4, 5.3 and 5.4).

This sub-section (cf. 5.2.1) has clarified where interest has been generated, resulting in citations and requests for papers. The following sub-sections now consider the impact and citations, within the contexts of the pedagogical approaches of Problem-solving using robots (cf. 5.2.2) problem-solving-first (cf. 5.2.3) PbBL and PjBL (cf. 5.2.4).

Paper Number	Published work (See Table 1.1)	R/G	Citing papers from Research Gate (2017) (R) Google Scholar (2017) (G)
1	Turner, S., **Hill, G. J**. (2006)	2R 5G	Adams, J. P., Kaczmarczyk, S., Picton, P. and Demian, P. (2010) Adams, J., P., Turner, S. (2008)
2	Turner, S., **Hill, G. J.** (2007)	10G 4R	Atmatzidou, S., Demetriadis, S. and Nika, P. (2017) Štuikys, V., (2015) Koulouri, T., Lauria, S., Macredie, R., D. (2014) Štuikys, V., Burbaitė, R., Damaševičius, R. (2013) Dravid, R., Duncan. A., (2011) Oddie, A., Hazlewood, P., Blakeway, S., Whitfield, A. (2010)
3	Turner, S., **Hill, G. J**. (2008)	8G 3R	Gold, N. (2010) Vallance, M. (2016)
4	Turner, S., **Hill,** G., Adams, J. (2009)	2R	Adams, J. P., Kaczmarczyk, S., Picton, P. and Demian, P. (2010)
5	Turner, S., **Hill, G. J.** (2010)	2G 3R	
6	**Hill, G. J.**, Turner, S. (2011)	2G	
7	Kariyawasam, K., A., Turner, S., **Hill, G**. (2012)	2G	
8	**Hill, G. J.,** Turner, S. (2014a)	1G 1R	
9	**Hill, G. J.** (2015)		
10	**Hill, G. J.** (2016)	1G	

Table 5.3: Citations for Published Works from Google Scholar (2017) and Research Gate, (2017).

5.2.2 Problem-solving using robots.

The importance of linking the problem-solving robot activity and the visual computer programming assignment, whilst maintaining the visual nature of the problem, has been discussed previously (cf. 4.6). Also, there has been a comparison of the research on the author's teaching, appraised here with similar research reported by other authors relating to teaching computer programming using robots citing Williams (2003) and being cited by Štuikys, Burbaitė and Damaševičius (2013) (cf. 4.5.1). The initial pedagogical approach incorporated the use of using physical robots and this published work was recognised and cited by Gold (2010, p11) (cf. 4.5.1). The research identified by the Turner and Hill (2008) paper was also recognised by Vallance (2016) citing that:

> "Turner et al. [26] used LEGO Mindstorms as a prerequisite for teaching Java programming at a university. They focus on problem-solving and robot maze emulation" (Vallance, 2016, p279).

Vallance (2016) was using robots that were programmed to solve 'systematic' problems to investigate meta-learning, where:

> "students evaluate and subsequently 'make sense' of their educational experiences" (Vallance, 2016, p274)

Suggesting that there has been limited research on meta-learning versus metacognition.

The 2007 paper (Turner and Hill, 2007) was cited and credited with contributing to:

> "research and ongoing developments…"

in:

> "….robotics" (Oddie et al., 2010, p2)

When the authors were also considering 'Introductory Problem Solving and Programming' and cited:

> "that when using robotics there was an improvement in grades and that there was a positive response from students stemming from the benefit that the use of robots provided a more tangible way of visualising the outcome of their programs" (Oddie et al., 2010, p3).

Atmatzidou, Demetriadis and Nika (2017) also cited Turner and Hill (2007) in their study that concluded that strong guidance in solving problems can have a positive impact on students' metacognitive and problem-solving skills.

As a direct consequence of the second conference presentation (Turner and Hill, 2007) the authors were approached by the Higher Education Academy (HEA) to write a paper for the HEA-Information and Computer Science, ITALICS journal (Turner and Hill, 2008) (cf. 5.2.1). The 2008 paper (Turner and Hill, 2008) discussed the initial findings and emphasis on problem solving and computer programming, using physical robots for the problem-solving phase of the module. This paper (Turner and Hill, 2008) was recognised by Dr Nicolas Gold of University College London and the use of robots in providing a:

> "motivational platform and domain" (Gold, 2010, p2).

5.2.3 Problem-solving first.

The idea that problem solving was an important aspect of computer programming and should be taught within computer programming modules was only beginning to emerge through the JICC5 (JICC5, 2001) discussions and a joint IEEE Computer Society and ACM task force, Computing Curricula (IEEE/ACM, 2001) report (cf. 2.6.1, 2.6.2, 4.1 and 4.2). Although the report:

> "sought to identify curricular models that minimize the weaknesses of the computer programming-first approach by focusing on algorithmic and problem-solving concepts rather than the vagaries of language syntax" (IEEE/ACM, 2001, p28).

However, they suggested three alternative models:

> "breadth-first approach that begins with a general overview of the discipline, an algorithms-first strategy that focuses on algorithms over syntax, and a hardware-first model …" (IEEE/ACM, 2001, p28).

Whereas the author proposed the problem-solving-first approach, as recognised in 2014 by Koulouri, Lauria and Macredie (2014) citing that:

> "evidence of the approach's value has been demonstrated by Turner and Hill [2007], who adopted it and reported positive student perceptions" (Koulouri et al., 2014, p22).

Most significantly Koulouri *et al*. (2014) concluded, after a 4-year '*iterative methodological*' study that:

> "teaching problem solving before programming yielded significant improvements in student performance" (Koulouri et al., 2014, p1).

5.2.4 Problem Based Learning (PbBL) and Project Based Learning (PjBL).

This research work has also been cited as an example of supporting/demonstrating one of the:

> "two learning models derived from the constructivist- based pedagogical approach: problem-based learning" (Štuikys, Burbaitė and Damaševičius, 2013, p127, p132)

in a suggested framework by Štuikys *et al*. (cf. 4.5.1).

However, it could be argued that in addition to PbBL cited by Štuikys, et al., (2013) the other pedagogical approach of PjBL advocated by the author should also be seen to be supported/demonstrated, but Štuikys *et al*., recognise:

> "there is a thin line among the models, nevertheless, we introduce them as slightly different teaching scenarios (in other words the models are integrated within the scenarios)" (Štuikys et al., 2013, p132, Štuikys, 2015, p296).

5.3 Impact on stakeholders.

The previous section (cf. 5.2) has clarified where interest has been generated, resulting in citations and requests for papers (cf. 5.2.1) and the impact and citations, within the contexts of the pedagogical approaches of Problem-solving using robots (cf. 5.2.2) problem-solving-first (cf. 5.2.3) PbBL and PjBL (cf. 5.2.4). Here, the author will itemise all the interest in this research and impact on any stakeholder. These stakeholders will be identified and the impact of this published work on subsequent research (Figure 5.1). This extensive range includes students (at all academic levels); lecturers within or outside the discipline; other Universities UK, EU and internationally; UK school education (primary and secondary) and even a commercial robot manufacturer that have been influenced by this research.

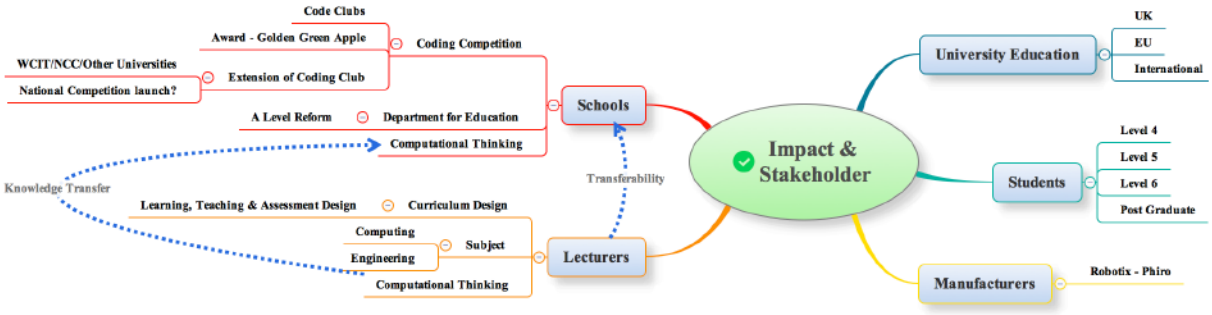


Figure 5.1: Impact and Stakeholders.

Stakeholders' influenced by this research can be categorised as:

- University Education
- Lecturers
- Students
- Schools
- Manufacturers

These can be further subdivided as:

i) University Education:

- UK
- EU
- International

ii) Lecturers:

- Curriculum Design
    - Learning, Teaching and Assessment Design
- Subject
    - Computing
    - Engineering

iii) Students: Students at all Higher Education (HE) level 4, 5, 6 and postgraduate have been influenced at

- Level 4
- Level 5
- Level 6
- Post Graduate

iv) Schools:

- Coding Competition (Code Clubs)
- Department for Education - A Level Reform

v) Manufacturers:

- Robotix - Phiro

The academic impact of the author's research has already been discussed (cf. 5.2) and is seen as impacting on the stakeholders of: i) University Education, ii) Lecturers and, iii) Students. In addition, this research and pedagogical approaches of PbBL, PjBL, Visual Programming (Hill and Turner, 2011; 2014a; 2016; Kariyawasam, Turner and Hill, 2012) have been disseminated to academics, not only within the author's University, but throughout the UK, Europe and internationally via the number of conference presentations and paper requests. Locations are indicated in Table 5.4.

Paper Number (See Table 1.1)	Location/impact of Published works
1	Dublin, Ireland, Europe.
2	Southampton, UK.
3	HEA journal (Higher Education Academy) – invitation based on 2.
4	Bath, UK.
5	Xian, China
6	Book chapter – invitation based on 5
7	Shanghai, China
8	International journal – invitation based on 6.
9	Zwickau, Germany, Europe.
10	Paper published from 9

Table 5.4: Location/impact of Published works).

Development funding support has also been received (Higher Education Academy-Information and Computer Science (HEA-ICS) Development Fund, 2015; HEA-ICS/Microsoft Innovative Teaching Fund, 2015) (cf. 4.5.1). In addition, funding was received to support a student researcher under an initiative within the University (**URB@N** - Undergraduate Research Bursaries at Northampton) this work led to the publication and dissemination of the Kariyawasam, Turner and Hill (2012) paper (cf. 3.2, 4.5.1 and 4.5.2).

The research on the author's pedagogical approaches, appraised here, has not just influenced curriculum design within the discipline of Computing, but as early as 2008 a collaboration between computing and engineering explicitly indicated that this research was innovative and had an impact on teaching in another allied discipline (cf. 2.3). In 2009, a paper was published to share and test the research across two related disciplines: computing and engineering (Turner, Hill and Adams, 2009*)*. Adams disseminated this research within Engineering via a number of papers (Adams and Turner, 2008, Adam *et al.*, 2008 and 2010) building on the research work by the author and Turner, having developed a dedicated problem solving and creativity module for engineering undergraduates. This indicated that, the problems first, PbBL and PjBL nature of the computing module (CSY1020 Problem Solving and Programming) had been adopted throughout the undergraduate curriculum, not only in Computing, but also within Engineering. This collaboration between computing and engineering explicitly indicated at an early stage that this research was innovative and had an

impact on teaching in another allied discipline (cf. 2.3). The author also adopted a PbBL and PjBL learning approach in the delivery of a postgraduate masters Java Programming module (CSYM021).

The research on the author's pedagogical approaches, appraised here, has also influenced education within schools, as the author was invited (in June 2012) to the Department for Education to discuss the Level 3 (A level/vocational) qualification reform. The meeting was a roundtable discussion as part of a specialist vocational focus group, where the author was able to suggest that the current curriculum was inadequate in preparing A level students for higher education and particularly technical computer programming.

The author (in 2015) was also instrumental in setting up, facilitating and judging a regional school Coding Competition (2017). The Coding Competition was a collaboration between the University of Northampton (UoN) Worshipful Company of Information Technologists (WCIT) (2017) Code Club (CC) (2017) and Northamptonshire County Council (NCC) (2017a) and aimed at primary and secondary school pupils with an overarching aim of *'Inspiring young people to achieve digital fluency through collaboration and challenge'* with a post competition quote summary reinforcing that:

> "Coding promotes problem solving, team-work and analytical thinking – and those who code from an early age will have a real advantage in the future jobs market." (Northamptonshire County Council, 2017b)

In addition, the author was influential in ensuring that a choice of two Visual Programming Languages (VPL) (cf. 4.5.3) i.e. Scratch (2017a) and App Inventor (2017) were used for the problem/project scenario.

The competition was awarded a GOLD Green Apple Award in 2016 for its contribution to 'Education and Training' by The Green Organisation (2017). Another tangible outcome of the project was the setting up of additional Code Clubs within the region. Given the success of the competition, the launch of a regional/national version occurred in Autumn 2017, with the potential of attracting additional high-profile sponsors.

Further, the author and his University have been credited by a robotics manufacturer (Robotix, 2015) with the statement (see Figure 5.2):

"programming robots in the context of problem solving provides a visual and physical way to see the outcome of problem solving" (Robotix, 2015).

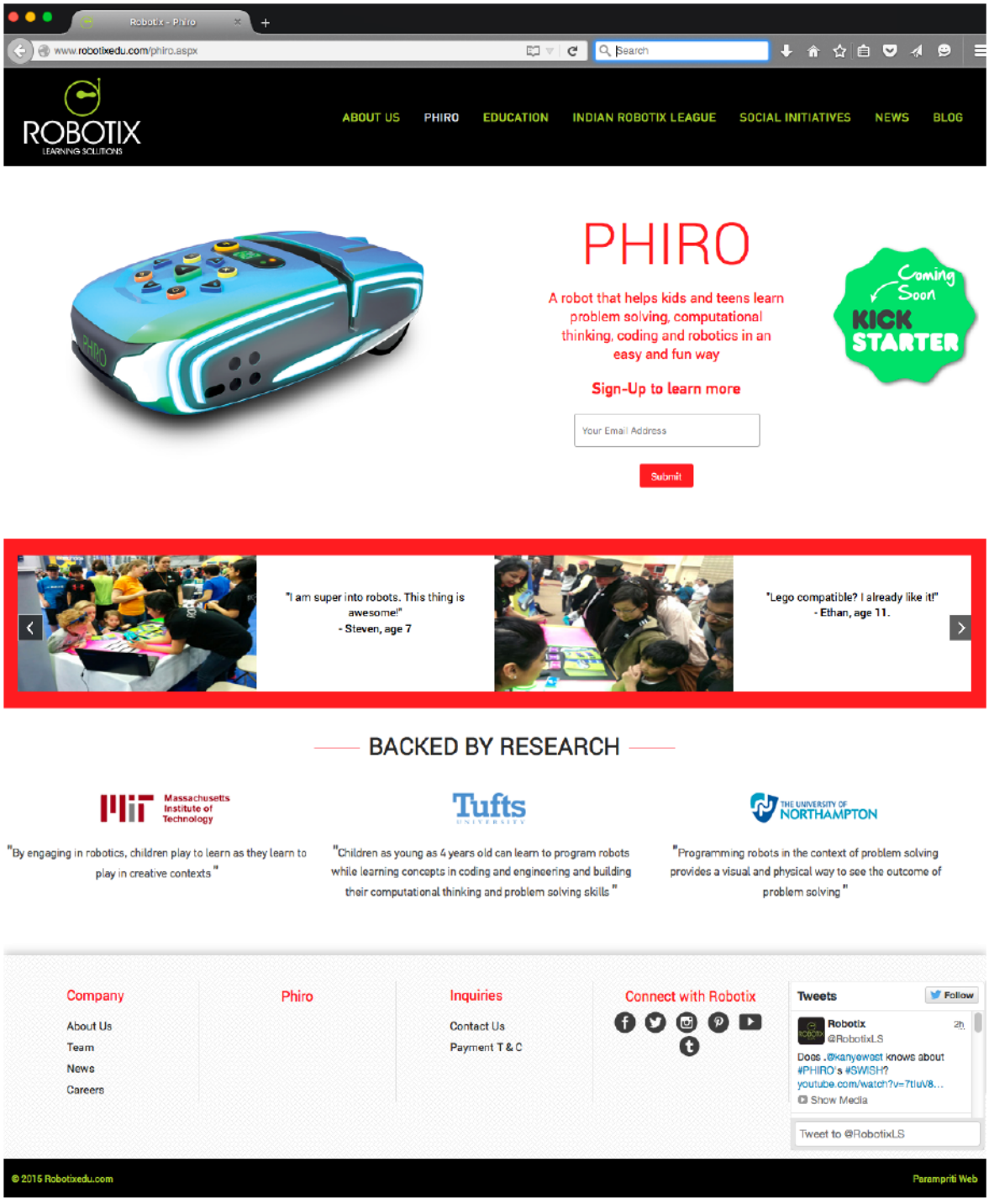


Figure 5.2: Backed by Research (Source: ROBOTIX, 2015).

This was cited, alongside Massachusetts Institute of Technology (MIT), by Robotix (2015) when promoting their new product PHIRO (Figure 5.2) stating that the product is "backed by research" (Robotix, 2015). The Phiro Robotix robot is also promoted as:

> "A robot that helps kids and teens learn problem solving, computational thinking, coding and robotics in an easy and fun way" (Robotix, 2015).

This section (cf. 5.3) has discussed the impact of the various 'stakeholders' influenced by this research.

5.4 Conclusion.

This chapter (cf. 5) has evidenced both quantitative and qualitative impact and reception of the author's published work to date. Specifically, by illustrating the evolution of the topic/research by evidencing where there have been requests for further published works and citations of the author's published research (cf. 5.2.1) and how the citations reinforced the author's research on Problem-solving using robots (cf. 5.2.2) Problem-solving first (cf. 5.2.3) Problem Based Learning (PbBL) and Project Based Learning (PjBL) (cf. 5.2.4). Finally, a broader overview of the impact on stakeholders (cf. 5.3).

The pedagogical approaches discussed have been disseminated to colleagues, not only within the University of Northampton, but also in Europe and Internationally. It is difficult to measure the true impact of these pedagogical approaches, but these have all naturally become a key enhancement to Computing (and other) higher education disciplines.

CHAPTER 6 CONCLUSION.

6.1 Introduction.

This critical appraisal has explicitly demonstrated the contribution to knowledge evidenced through the pedagogical innovation and impact on student learning the author's published work has achieved. In addition, and more generically, the critical appraisal has detailed the extent to which the author's published works provide a coherent demonstration of:

> "1. The creation and interpretation of new knowledge, through original research or other advanced scholarship, of a quality to satisfy peer review, extend the forefront of the discipline, and merit publication*,
>
> 2. A systematic acquisition and understanding of a substantial body of knowledge which is at the forefront of an academic discipline or area of professional practice" (The University of Northampton, 2015, p6).

[* All the author's work considered here has been published].

The focus of the published work has been the teaching of computer programming and problem solving to undergraduate first year computing students, using visual computer programming, together with robot's/robot simulators. The author is responsible for the original idea, design, development and introduction of a new first year undergraduate module at the University of Northampton in 2004. The new first year module called "CSY1020 Problem Solving and Programming" enabled proposed pedagogical approaches to be implemented and evaluated. The overarching new and innovative pedagogical approach introduced and implemented by the author, that improved and enhanced student learning was called 'Problems-first' and 'Graphics-first'. The author has used the term 'Graphics-first' to define the approach, whereby visual computer programming and graphical user interfaces (GUI) are developed from the outset and throughout the module.

The teaching and learning strategy adopted and proposed by the author is multifaceted and has focussed primarily on:

- **Problems-first** before computer programming.
- Visual computer programming that still emphasises problem solving.

- Taught via a **PbBL** approach within a **PjBL** context.
- Throughout there has always been a strong emphasis on the physical and **visual** nature of the topics.

The importance of problem solving first before computer programming has been evidenced, together with the evolution of the problems first approach into the 'Problems first, second and third' (Hill and Turner, 2014a) to reflect problems-first and progression to PbBL and PjBL approaches advocated by the author (cf. 2.4). These together with the importance of computer programming always requiring problem solving skills and computational thinking.

This critical appraisal and the author's published works provide a coherent demonstration of the new pedagogical innovation which focused on and emphasised the importance of the 'Problems-first' and 'Graphics-first' learning and teaching strategies, that were subdivided into **four** core pedagogical approaches (cf. 4.1):

- Problem Solving (PS).
- Problem-Solving-First (PSF) /Problems-First (PF).
- Problem Based Learning (PbBL)/Project Based Learning (PjBL).
- Physical to Visual/Visual Programming.

These **four** core pedagogical approaches have been applied by the author to the teaching of computer programming and problem solving to undergraduate first year computing students, using robots/robot simulators and visual programming to emulate the robot tasks. Students were encouraged to solve problems as part of a project/assignment that is related to a pseudo real-world problem which ultimately aimed to stimulate deeper learning and its transferability.

The need to focus initial computer programming education on problem solving, prior to the teaching of computer programming syntax and software design methodology, is also recommended by the author. The main vehicle for this approach is a robot/robot simulation programmed in the Java programming language, followed by the visual computer programming of a graphical representation/simulation to develop computer programming skills all delivered by the four pedagogical approaches.

The author also introduced a 'Graphics-first' approach to computer programming from the outset, such that the visual computer programming and graphical user interfaces (GUI) were developed throughout the module.

The author has been the first to define and differentiate between Problem Based Learning and Project Based Learning by using the abbreviations PbBL and PjBL and defining PjBL as where a project/challenge is set from the outset, such that one PbBL activity leads to another and the series of linked problems form the greater challenge or project. The author's innovative approach can be seen through adhering strictly to the above definition, throughout the delivery of the Programming module, which combined visual PbBL and visual PjBL. This is also due to the integrated curriculum and that the problems and potential projects continue, not only through one semester or year, but also could be continued from year to year, possibly even integrated across the whole curriculum – not just one silo module - that has progression from year to year within the widely used modular type curriculum.

The methodology used (cf. 3.1) was action/practitioner research, where the proposed learning and teaching strategies were designed, developed, implemented, evaluated and reflected upon by the author to enable refined approaches to be further re-implemented. The findings and recommendations of this research are that the author's contribution and impact evidences to include the following learning and teaching strategies:

- Problem Solving (PS) throughout the teaching of computer programming (and allied subjects);
- Visual Problem Based Learning (PbBL) visual Project Based Learning (PjBL) and;
- Physical to visual/visual programming (VP).

The previous chapter (cf. Chapter 5) has evidenced the impact on the stakeholders (See Figure 6.1):

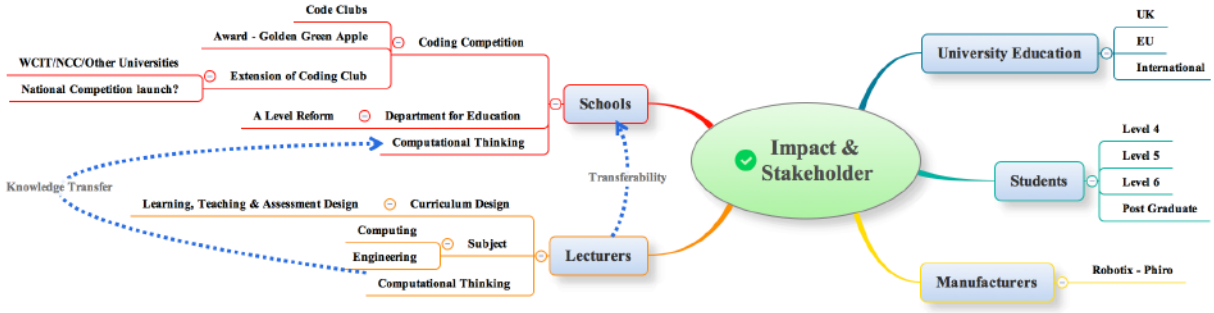


Figure 6.1: Impact and Stakeholders (cf. Figure 5.1)

The recognised importance of problem solving has been reiterated and the idea of problems first, supported by Koulouri *et al*. (2014). The experiences described here indicate that students liked the visual nature of the Problem Solving and Programming module and that this helped their skills, but also that the students appreciate the importance of these skills as they advance in academic level (Kariyawasam, Turner and Hill, 2012) with student satisfaction consistently over 90%. PbBL/PjBL is further supported and advocated by Štuikys *et al*. (2013) but the importance of active learning (Chickering and Gamson, 1987) has also been recognised. There has been an interesting shift from abstract to physical to simulation and finally the use of graphics-first/visual computer programming and the visual simulation of robots.

6.2 Conclusions and Recommendations for the Future.

The author since 2001 has advocated the importance of 'Problems-first' and 'Graphics-first' via visual computer programming to first year undergraduates, with an emphasis on computational thinking. In parallel, the aim has been to achieve the most 'active learning' and instil a lifelong ability of 'learning to think' which is transferable to all future learning. The use and effectiveness of PbBL and PjBL to, not only the computer programming curriculum, but to other subjects within Computing and allied subject disciplines of Engineering. Finally, the vehicle for these approaches has been visual/graphical where students get instant reinforcement of the learning as the output of their computer programming logic manifests itself into a visual outcome on the computer screen.

The introduction to this critical appraisal (cf. Chapter 1) emphasised the importance of a summary paper to offer a review of a problems-first approach to first year undergraduate computer programming (Hill, 2016). The short paper attempted to review the pedagogic approach adopted by the author since 2001, of problems-first and visual computer programming to first year undergraduates. This paper, in addition to the three requests for the authors findings (Turner and Hill, 2008, Hill and Turner, 2011, Hill and Turner, 2014a) demonstrates the interest this published research has stimulated. The 2016 paper additionally referenced several authors that have citied this research, confirming the impact and broad reach of the research on the author's teaching, appraised here.

The author's research is also timely, relevant and impactful due to the introduction of the Teaching Excellence Framework (TEF) (cf. 5.1) in 2016, due to its proposed pedagogical innovation. Also, with the introduction of the new Computing National Curriculum (Department for Education, 2012) in September 2013, not only has computer programming become prominent within Schools, but both the introduction of computational thinking (CAS, 2015a, b) (cf. 2.1.4) and the use of visual programming languages (cf. 4.5.3).

The author will continue to disseminate and promote this research as it continues to be relevant and timely as follows:

1. Schools/Community Groups: The Coding Competition (2017) launched in academic year (AY) 2015/16 was a county wide initiative, this is to be promoted in AY 2017/18 as predominantly a Northamptonshire and surrounding counties competition with entries welcomed from anywhere in the UK. The aim is to launch this as a national competition for AY2018/19. The Coding Competition (2017) received ongoing funding and support from the Worshipful Company of Information Technologists (WCIT, 2017) for the three events. In addition to the WCIT, the other two partners were the Code Club (CC) (2017) and Northamptonshire County Council. In 2016, the competition was also awarded a GOLD Green Apple Award for Education and Training, presented in a ceremony at the Houses of Parliament, Westminster, London (November 2016).

2. Publications: Due in part, to the link with the Code Club (CC) (2017) the author has since published a paper (Hill, Turner, Childs, 2017) as first author, with a University

colleague and a representative of the Code Club. This paper is partly based on the findings of this thesis, that could not be included as a primary paper as it was published after the registration for this PhD by means of published works, but some of the discussions within this critical appraisal were articulated in the paper (cf. 4.4). The paper, entitled *'The answer's not on the screen'* emphasises the importance that Problem Solving, Problem-Solving-First, PbBL and PjBL play in learning computer programming, whilst resisting the temptation to try and find the solution on line, or by typing in every permutation of computer code into their computer programme (cf. 4.4).

3. University Education: Within the University of Northampton, there is a demand to use Active Blended Learning (ABL) as the preferred delivery method for the University specifically from 2018, when the University moves to a new campus and the 'new' delivery style. The author, as Head of Computing, will support Computing and other areas within the University (UoN) during the transition. The author would argue all the research on the author's teaching, appraised here so far, fits within the definition of ABL.

4. Finally, as recently as September 2017, the author has been approached by the publisher Pearson to contribute to the new/8th edition of the book "Java: An Introduction to Problem Solving and Programming" authored by Savitch/Mock:

   "In view of your extensive experience in Java, I would like to invite you to participate as a contributor to this edition. Your expertise will be of great help in ensuring the content for the Global Editions is optimized for the international readership".

REFERENCES.

Academia.edu (2017) *Analytic*s [online] Available from: https://northampton.academia.edu/GaryHill/Analytics/activity/documents [Accessed: 30 June 2017].

Adams, J. P. and Turner, S. J. (2008) *Problem Solving and Creativity for Undergraduate Engineers: process or product?* In: International Conference on Innovation, Good Practice and Research in Engineering Education July 14-16, 2008, Loughborough, England, Higher Education Academy. 9781904804659.

Adams, J., Turner, S., Kaczmarczyk, S., Picton, P. and Demian, P. (2008) *Problem solving and creativity for undergraduate engineers: Findings of an action research project involving robots*. In: International Conference on Engineering Education (ICEE 2008) Budapest, Hungary.

Adams, J. P., Kaczmarczyk, S., Picton, P. and Demian, P. (2009) *Problem solving and creativity in engineering: perceptions of novices and professionals*. In: Ao, S. I., Douglas, C., Grundfest, W. S. and Burgstone, J. (eds.) Proceedings of the World Congress on Engineering and Computer Science 2009, Vol. 1. Hong Kong: Newswood Limited. 9789881701268.

Adams, J. P., Kaczmarczyk, S., Picton, P. and Demian, P. (2010) *Problem solving and creativity in engineering: conclusions of a three-year project involving reusable learning objects and robots*. Paper presented to: Engineering Education 2010 (EE 2010) Aston University, Birmingham, England, 06-08 July 2010.

Alexander, S., Irons, A., (eds.) (2004) *Effective Learning and Teaching in Computing*. Effective Learning and Teaching in Higher Education . Routledge, ISBN 9780415335010.

App Inventor (2017) [online] Available from: http://appinventor.mit.edu/explore/ [Accessed: 22/06/17].

Atmatzidou, S., Demetriadis, S. and Nika, P. (2017) *How Does the Degree of Guidance Support Students' Metacognitive and Problem Solving Skills in Educational Robotics?* Journal of Science Education and Technology, https://doi.org/10.1007/s10956-017-9709-x pp1-16.

Barbe, W. B., Swassing, R. H., Milone, M. N., (1979) *Teaching through modality strengths: concepts and practices*. Columbus, Ohio: Zaner-Bloser. ISBN 0883091003. OCLC 5990906.

Beaumont, C. and Fox, C. (2003) *Learning programming: Enhancing quality through problem-based learning.* Proceeding of 4th Annual Conference of the subject centre for Information and Computer Sciences of the Higher Education Academy (pp. 90-95). Newtownabbey, Northern Ireland: Higher Education Academy.

Bell, D. and Parr, M. (2001) *Java for Students*, 3rd ed. Upper Saddle River, NJ: Prentice Hall, ISBN 13: 9780130323774.

Bell, D. and Parr, M. (2006) *Java for Students*, 5th ed. Upper Saddle River, NJ: Prentice Hall, ISBN 13: 9780131735798.

Bell, D. and Parr, M. (2010) *Java for Students*, 6th ed. Upper Saddle River, NJ: Prentice Hall, ISBN 13: 9780273731221.

Blackboard (2017) [online] Available from: http://uk.blackboard.com/about-us/index.aspx [Accessed 02/09/2017].

Bloom, B. S. (Ed.) (1956) *Taxonomy of educational objectives*. In: Handbook I: Cognitive domain. White Plains, NY: Longman.

Buck Institute for Education (2017) *Buck Institute for Education –About* [online] Available from: http://www.bie.org/about [Accessed: 23 February 2017].

Burton, D., Bartlett, S. (2009) *Key Issues for Education Researchers*, 192 pages | SAGE Publications Ltd, DOI: http://dx.doi.org/10.4135/9781446269480, ISBN: 9781847873583.

Caldwell, H., (Editor) Smith, N., (Editor) (2016) *Teaching Computing Unplugged in Primary Schools*, Sage Publications Ltd, ISBN-10: 147396170X.

CAS - Computing At School (2015a) *QuickStart Computing*: *A CPD toolkit for primary teachers*, CAS, ISBN: 978-1-78339-521-7.

CAS - Computing At School (2015b) *Computational thinking: A guide for teachers, CAS,* [online] Available: http://community.computingatschool.org.uk/files/6695/original.pdf [Accessed: November 13, 2015].

Chickering, A. W., Gamson. Z. F. (1987) *Seven Principles for Good Practice in Undergraduate Education.* AAHE Bulletin 39:3-7. ED 282 491.6 pp. MF-01; PC-01.

Code Club (2017) [online] Available from: https://www.codeclub.org.uk/ [Accessed: 22/06/17].

Cohen, L., Manion, L., Morrison, K. (2011) *Research Methods in Education*, 7th edition, London: Routledge, ISBN 13:978-0-415-58336-7.

Colyer, J., (2013) *Making inquiry work in your classroom – OHASSTA* [online] Available from http://en.ohassta-aesho.org/wp-content/uploads/2013/09/OHASSTA_2013_Colyer_Inquiry-1.pdf [Accessed: February, 2017].

Cook, T. (2009) *The purpose of mess in action research: building rigour though a messy turn*, Educational Action Research, Volume 17, Issue 2, Routledge, pp. 277-291.

de Graaff, E., Kolmos, A. (2007) *'History of problem-based and project-based learning'*. in E de Graaff and A Kolmos (eds) Management of change: implementation of problem-based and project-based learning in engineering. Sense Publishers, Rotterdam, pp. 1-8.

Dehouck, R., (2015) *The maturity of visual programming* [online] Available from: http://www.craft.ai/blog/the-maturity-of-visual-programming/ [Accessed: June, 2017].

Department for Education (2012) *Statutory guidance National curriculum in England: computing programmes of study* [online] Available from: https://www.gov.uk/government/publications/national-curriculum-in-england-computing-programmes-of-study/national-curriculum-in-england-computing-programmes-of-study [Accessed: December, 2015].

Dewey, J. (1897) *My Pedagogic Creed*, The School Journal, Volume LIV, Number 3, pp 77-80.

Dewey, J., (1916) *Democracy and Education: An Introduction to the Philosophy of Education*. New York: Macmillan [online] Available from: http://www.johndeweyphilosophy.com/books/democracy_and_education/ [Accessed: February, 2017].

Dewey, J. (1933). *How we think: A restatement of the relation of reflective thinking to the educative process*. New York: D.C. Heath and Company.

Dewey, J. (1938) *Experience and Education*, New York: Collier Books.

Donaldson, J., (2014) *The Maker Movement and the Rebirth of Constructionism*. [online] Available from http://www.hybridpedagogy.com/journal/constructionism-reborn/ [Accessed: October 26, 2015].

Dravid, R., Duncan. A. (2011) *Engineering soft skills development to avoid hard knocks*, Global Engineering Education Conference (EDUCON) 2011 IEEE, pp354-357, DOI: 0.1109/EDUCON.2011.5773160.

ESRC (2016) *What is impact? Economic and Social Research Council,* [online] Available from http://www.esrc.ac.uk/research/impact-toolkit/what-is-impact/ [Accessed: 01/12/16]

Fagin, B. (2003). *Ada/Mindstorms 3.0*. Institute of Electrical and Electronic Engineering Robotics & Automation Magazine, 10(2), 19-24.

Fine, G.A. & Deegan, J.G. (1996) *Three Principles of Serendip: insight, chance, and discovery in qualitative research*, International Journal of Qualitative Studies in Education, 9, pp. 434-447.

Fleming, N.D., Mills, C. (1992) *Not Another Inventory, Rather a Catalyst for Reflection. To Improve the Academy*, 11, pp 137-155.

Gallopoulos, E., Houstis, E., Rice, J. R. (1994) *Computer as Thinker/Doer*. Problem-Solving Environments for Computational Science, IEEE Computational Science and Engineering pp 11-23.

Gold, N. (2010) *Motivating Students in Software Engineering Group Projects: An Experience Report*. Innovation in Teaching and Learning in Information and Computer Sciences 9(1) 10-19. DOI: 10.11120/ital.2010.09010010.

Google Scholar (2017) *Citations: Gary Hill* [online] Available from https://scholar.google.co.uk/citations?hl=en&user=gKBvlg8AAAAJ [Accessed: 03/09/17].

GradeMark (2017) [online] Available from: https://guides.turnitin.com/01_Manuals_and_Guides/Instructor_Guides/Turnitin_Classic_(Deprecated)/25_GradeMark [Accessed 02/09/2017].

Greenfoot (2015) *Teach and Learn Java Programming*. [online] Available from http://www.greenfoot.org/ [Accessed: February 1, 2015].

HEA-ICS Development Fund (2015) *HEA-ICS Development Fund* [online] Available from: http://www.ics.heacademy.ac.uk/projects/development-fund/index.php [Accessed: February 2015].

HEA-ICS/Microsoft Innovative Teaching Fund (2015) *Developing problem-solving teaching materials based upon Microsoft Robotics Studio* [online] Available from: http://www.ics.heacademy.ac.uk/projects/development-fund/fund_details.php?id=88 [Accessed February 2015].

HEFCE (2017) *Initial decisions on the Research Excellence Framework 2021 (REF2017/01-September 2017*) [online] Available from: http://www.hefce.ac.uk/media/HEFCE,2014/Content/Pubs/Independentresearch/2017/REF,201701/REF2017_01.pdf [Accessed September 2017].

Hill, G. J., Turner S. (2011) *Chapter 7: Problems First.* In: Software Industry-Oriented Education Practices and Curriculum Development: Experiences and Lessons, M Hussey, X Xu and B Wu (Eds.) IGI Global, USA, pp 110-126, ISBN: 978-1-60960-797-5.

Hill, G., Turner, S. (2013) *Electronic Online Marking of Software Assignments (EOMOSA)* 9th China - Europe International Symposium on Software Industry Engineering Education, 13-14 May 2013, AICA, Milan, Italy.

Hill, G., Turner, S. J. (2014a) *Problems first, second and third.* International Journal of Quality Assurance in Engineering and Technology Education (IJQAETE). **3**(3) pp. 88-109. 2155-496X.

Hill, G., Turner, S. (2014b) *"Chapter 5: Electronic Online Marking of Software Assignments"*, Progress in IS: Software Engineering Education for a Global E-service Economy, Gianmario M., Wu B (Eds.) Springer, ISBN 978-3-319-04216-9, DOI 10.1007/978-3-319-04217-6_5, pp 41-48.

Hill, G. J. (2015) *Review of a problems-first approach to first year undergraduate programming,* 11th China – Europe International Symposium on Software Industry Orientated Education: 29-30th April 2015, Zwickau, Germany. (Not referenced yet)

Hill, G. J. (2016) *Review of a problems-first approach to first year undergraduate programming*, Software Engineering Education Going Agile: 11th China–Europe International Symposium on Software Engineering Education (CEISEE 2015): Kassel S., Wu B (Eds.) Springer, pp. 73-80, ISBN 978-3-319-29165-9.

Hill, **G. J**., Turner, S., Childs, K (2017) *The answer's not on the screen*, Paper presented to: China Europe Symposium on Software Engineering Education (CEISEE) Athens, Greece, 24th-25th May 2017.

Houghton, W. (2004) *How can Learning and Teaching Theory assist Engineering Academics?* School of Engineering - University of Exeter. [online] Available from: https://www.heacademy.ac.uk/sites/default/files/learning-teaching-theory.pdf [Accessed: November 2015].

Hung, W., Jonassen, D. H. and Liu, R. (2008) *Problem-based learning*, In J. M. Spector, J. G. van Merriënboer, M. D., Merrill and M. Driscoll (Eds.) *Handbook of research on educational communications and technology* (3rd ed., pp. 485-506). Mahwah, NJ: Erlbaum.

IEEE/ACM (2001) *IEEE CS, ACM Joint Task Force on Computing Curricula,* IEEE Computer Society Press and ACM Press. [online] Available from: http://www.acm.org/education/curricula.html [Accessed: February, 2015].

Inhelder, B., Piaget, J. (1958) *The growth of logical thinking from childhood to adolescence: An essay on the construction of formal operational structures*, New York: Basic Books.

Jenkins, T. (2002) *On the Difficulty of Learning to Program*. 3rd Annual Conference of the LTSN Centre for Information and Computer Sciences, Loughborough, LTSN-ICS, 53-58, [online] Available from: http://www.psy.gla.ac.uk/~steve/localed/jenkins.html [Accessed: August 2017].

JICC5 (2001) *Java and the Internet in the Computing Curriculum*, Higher Education Academy (HEA) – Information and Computer Sciences (ICS) Conference, South Bank University, London, 22nd Jan, [online] Available from: http://www.ics.heacademy.ac.uk/events/displayevent.php?id=127 [Accessed: February 2015].

Kariyawasam, K. A., Turner, S., **Hill, G**. (2012) *Is it Visual? The importance of a Problem Solving Module within a Computing course*. Computer Education, Volume 10, Issue 166, May 2012, pp. 5-7, ISSN: 1672-5913.

King, A., (1993) *From Sage on the Stage to Guide on the Side*, College Teaching, Taylor and Francis, Ltd., Vol. 41, No. 1 (Winter, 1993) pp. 30-35

Kolb, D. (1984) *Experiential Learning: Experience as the source of learning and development*, Englewood Cliffs, N. J.: Prentice-Hall.

Kolb, A. and Kolb, D. (2009) *Experiential learning theory: A dynamic, holistic approach to management learning, education and development*. In S. Armstrong and C. Fukami (Eds.) The SAGE handbook of management learning, education and development. (pp. 42-69). London: SAGE Publications Ltd. doi: http://dx.doi.org/10.4135/9780857021038.n3.

Koulouri, T., Lauria, S., Macredie, R., D. (2014) *Teaching introductory programming: A quantitative evaluation of different approaches*. In: ACM Transactions on Computer Education (TOCE) Volume 14, Issue 4, Article 26 (December 2014) 28 pages, DOI: http://dx.doi.org/10.1145/2662412.

Larmer. J., Mergendoller, J., R. (Eds.) Boss, S., (2013) *PBL for 21st Century Success, Teaching Critical Thinking, Collaboration, Communication, and Creativity*, Project Based Learning Toolkit Series, Buck Institute for Education, ISBN 978-0-9740343-6-2.

Lathram, B., Lenz, B., Vander Ark, T. (2016) *Preparing Students for a Project-Based World*, Buck Institute for Education, Getting Smart [online] Available from: http://www.gettingsmart.com/wp-content/uploads/2016/08/Preparing-Students-for-a-ProjectBasedWorld-FINAL.pdf [Accessed: 23 February 2017].

Lawhead, P. B., Bland, C. G., Barnes, D. J., Duncan, M. E., Goldweber, M., Hollingsworth R. G., & Schep, M. (2003). *A road map for teaching introductory programming using LEGO Mindstorms robots.* Association for Computing Machinery Special Interest Group on Computer Science Education Bulletin, 35(2), 191-201.

Leat, D., (1998) *Thinking through geography*. Cambridge: Chris Kington Publishing, ISBN-10:1899857427.

Lee, R., Kwan, C. (1997) *The use of problem-based learning in medical education*, Journal of Medical Education, Volume 1, No. 2, pp 149-58.

Lewin, K. (1946) *Action Research and Minority Problems*, Journal of Social Issues, 2: 34–46. doi:10.1111/j.1540-4560.1946.tb02295.x.

Lewin, K., Lippitt, R. and White, R. K. (1939) *Patterns of aggressive behaviour in experimentally created "social climates",* Journal of Social Psychology, Volume 10, Issue 2, pp 271-299.

Lewin, K. (1951) *Field theory in social science; selected theoretical papers*, D. Cartwright (ed.). New York: Harper and Row.

Martinez, S.L. and Stager, G. (2013) *Invent to learn: Making, tinkering, and engineering in the classroom*. Torrance, CA: Constructing Modern Knowledge Press.

Mellor (2001) *Messy method: the unfolding story*, Educational Action Research, 9:3, Routledge, pp. 465-484, ISSN: 0965-0792

Microsoft: Microsoft robotics studio (2015) *Microsoft robotics studio* [online] Available from: http://msdn2.microsoft.com/en-us/robotics/aa731520.aspx [Accessed: February 2015].

Miettinen, R. (2000) *The concept of experiential learning and John Dewey's theory of reflective thought and action*, International Journal of Lifelong Education, 19:1, 54-72, DOI: 10.1080/026013700293458.

NECTAR (2016) *NECTAR - Northampton Electronic Collection of Theses and Research,* [online] Available from http://nectar.northampton.ac.uk/ [Accessed: November 2016].

New Zealand Ministry of Education (2015) *How Experiential Learning relates to other experiences* [online] Available from: http://health.tki.org.nz/Key-collections/Curriculum-in-action/Making-Meaning/Teaching-and-learning-approaches/Experiential-learning-cycle [Accessed: December 2015].

Northamptonshire County Council, (2017a) [online] Available from: http://www3.northamptonshire.gov.uk [Accessed: 22/06/17].

Northamptonshire County Council, (2017b) [online] Available from: http://www3.northamptonshire.gov.uk/news/council-news/Pages/2016/Winners-announced-in-Race-To-The-Top-coding-competition-for-schools.aspx [Accessed: 22/06/17].

NTRP - National Teacher Research Panel (2003) *Vygotsky's ideas on teaching and learning,* [online] Available from http://www.ntrp.org.uk/content/vygotskys-ideas-teaching-and-learning [Accessed: January 19, 2017].

Oddie, A., Hazlewood, P., Blakeway, S., Whitfield, A. (2010) *Introductory Problem Solving and Programming: Robotics Versus Traditional Approaches*, Innovation in Teaching and Learning in Information and Computer Sciences, Vol. 9, Issue. 2.

Papert, S. (1993) *Mindstorms: Children, computers and powerful ideas*. 2nd edn. Hemel Hempstead: Harvester Wheatsheaf, ISBN 0-465-04629-0.

Piaget, J. and Roberts, G.-A. (1973) *To understand is to invent: The future of education*, New York: Grossman Publishers.

Piaget. J., Kamii, C. (1978) *What is psychology?* American Psychologist, Vol 33(7) Jul 1978, pp 648-652.

Powell, S., D. (2005) *Introduction to Middle School*, Pearson, 416 pp, ISBN13: 9780130600905.

Price, B. A., Richards, M., Petre, M., Hirst, A., & Johnson, J. (2003). Developing robotics e-teaching for teamwork. International Journal of Continuing Engineering Education and Lifelong Learning, 13(1/2), 190-205.

Quality Assurance Agency (2014) *The Frameworks for Higher Education Qualifications of UK Degree-Awarding Bodies*, [online] Available from: http://www.qaa.ac.uk/en/Publications/Documents/qualifications-frameworks.pdf [Accessed: 18/08/17].

RCUK (2016) *Pathways to impact*, Research Councils UK [online] Available from http://www.rcuk.ac.uk/innovation/impacts/ [Accessed: 01/12/16].

Research Gate (2017) *Citations: Gary Hill* [online] Available from https://www.researchgate.net/profile/Gary_Hill3 [Accessed: 03/09/17].

Resnick, M. (2013) *Lifelong Kindergarten. Cultures of Creativity*, LEGO Foundation, p50-52 [online] Available from: http://web.media.mit.edu/~mres/papers/CulturesCreativityEssay.pdf [Accessed: 20/02/17].

ROBOTIX (2015) *PHIRO*. [online] Available from http://www.robotixedu.com/phiro.aspx [Accessed: October 26, 2015].

Robson, C. and McCartan, K. (2015) *Real World Research*, 4th ed. Wiley, ISBN 9781118745236.

Royce, W., (1970) *Managing the Development of Large Software Systems* (PDF) Proceedings of IEEE WESCON, 26 (August): 1–9.

Savin-Baden, M. (2000) *Problem-based Learning in Higher Education: Untold Stories*, Buckingham: Open University Press/SRHE.

Savin-Baden, M. and Wilkie, K.: (eds) (2004) *Challenging Research in Problem-based Learning.* Maidenhead: Open University Press/SRHE.

Scratch (2017a) *About Scratch* [online] Available from: https://scratch.mit.edu/about/ [Accessed: 02/06/17].

Scratch (2017b) *Getting Started Guide* [online] Available from: https://scratch.mit.edu/help/ [Accessed: 02/06/17].

Smith. P., (2015) *The spiral of experiential learning* [online] Available from: http://www.petersmith.org/journal/2012/07/the-spiral-of-experiential-learning/ [Accessed: December 2015].

Solomon, C., (2013) *Invent to learn: Making, tinkering, and engineering in the classroom*, Rev. of: Invent to learn: Making, tinkering, and engineering in the classroom by Martinez, S.L. and Stager, G. (2013) p(v).

Štuikys, V., Burbaitė, R., Damaševičius, R. (2013) *Teaching of Computer Science Topics Using Meta-Programming-Based GLOs and LEGO Robots.* Informatics in Education - An International Journal Vol12 (1) pp125-142.

Štuikys, V. (2015) *Smart Learning Objects for Smart Education in Computer Science: Theory*, Methodology and Robot-Based Implementation. Springer.

TEF (2016) *Teaching Excellence Framework: Year Two additional guidance (HEFCE 2016/32)* [online] Available from: http://www.hefce.ac.uk/media/HEFCE,2014/Content/Pubs/2016/201632/HEFCE2016_32.pdf [Accessed September 2017].

The Green Organisation (2017) [online] Available from: http://thegreenorganisation.info/ [Accessed: 22/06/17].

Turner, S., **Hill,** G. J. (2006) *The Inclusion of Robots within the teaching of problem solving: Preliminary results*. 7th Annual Conference of the ICS HE Academy, Trinity College, Dublin, 29th - 31st August 2006, Proceedings pg 241-242 ISBN 0-9552005-3-9.

Turner, S., **Hill,** G. J. (2007) *Robots in Problem-Solving and Programming.* 8th Annual Conference of the Subject Centre for Information and Computer Sciences, University of Southampton, 28th - 30th August 2007, pp 82-85 ISBN 0-978-0-9552005-7-1.

Turner, S., **Hill, G. J**. (2008*) Robots within the teaching of Problem-Solving.* ITALICS, HEA-ICS, Volume 7 Issue 1, June 2008, pp. 108-119, ISSN: 1473-7507.

Turner, S., **Hill,** G., Adams J. (2009) *Robots in problem solving in programming.* 9th 1-day Teaching of Programming Workshop, University of Bath, 6th April 2009.

Turner, S., **Hill, G. J.** (2010) *Innovative Use of Robots and Graphical Programming in Software Education*. In: Computer Education, Volume 9, May 2010, pp. 54-6, ISSN: 1672-5913 (2010).

Turnitin (2016) *Turnitin* [online] Available from: http://www.turnitinuk.com/en_gb/ [Accessed: November 2016].

University of Northampton (2015) *Research Degrees Handbook (Section A 2014-15)*: A2.1.3 Application for PhD by Means of Published Works p6.

University of Northampton (2017) *Active Blended Learning*, The Institute of Learning and Teaching in Higher Education (ILT) [online] Available from: https://www.northampton.ac.uk/ilt/current-projects/defining-abl/ [Accessed: 30/06/17].

Vallance. M. (2016) *Establishing Meta-Learning Metrics When Programming Mindstorms EV3 Robots*. In: Uden L., Liberona D., Feldmann B. (eds) Learning Technology for Education in Cloud – The Changing Face of Education. LTEC 2016. Communications in Computer and Information Science, vol 620. Springer, Cham, pp 274-288, DOI: 10.1007/978-3-319-42147-6_23.

VOSviewer (2016) *VOSviewer - constructing and visualizing bibliometric networks*, Version 1.6.5 [online] Available from: http://www.vosviewer.com/ [Accessed: November 2016].

Vygotsky. L., S. (1978) *Mind in Society, The Development of Higher Psychological Processes*, Massachusetts/London: Harvard University Press.

Weisfeld, M. (2013) *What Skills Employers Want in a Software Developer: My Conversations with Companies Who Hire Programmers*, Nov 12, 2013 [online] Informit.com. Available at: http://www.informit.com/articles/article.aspx?p=2156240 [Accessed 18 Aug. 2017].

Williams, A. B. (2003) *The qualitative impact of using Lego Mindstorms robot to teach computer engineering*. In: Institute of Electrical and Electronic Engineering (IEEE) Transactions on Education, 46, 206.

Wing, J. (2006) *Computational thinking*. Communications of the Association for Computing Machinery, 49(3) 33.

Wing, J. (2008a) *Computational thinking*. Phil. Trans. R. Soc. A (2008) 366, 3717–3725 doi:10.1098/rsta.2008.0118.

Wing, J. M. (2008b) *Five deep questions in computing*. Commun. ACM 51, 58–60. (doi:10.1145/1327452.1327479).

Woods, D., R. (2006) *Preparing for PBL*, 3rd ed., McMaster University, Hamilton, ON, [online] Available from http://chemeng.mcmaster.ca/problem-based-learning/woods-preparing-for-pbl/ [Accessed: November 30, 2015].

Worshipful Company of Information Technologists (2017) [online] Available from: https://www.wcit.org.uk/ [Accessed: 22/06/17].

APPENDIX 1SUMMARY OF PUBLISHED WORKS (IN CHRONOLOGICAL ORDER).

#	Publication Title
1	Turner, S., **Hill, G. J**. (2006) *The Inclusion of Robots Within The Teaching OF Problem Solving: Preliminary Results*, 7th Annual Conference of the ICS HE Academy, Trinity College, Dublin, 29th - 31st August 2006, Proceedings pg 241-242 ISBN 0-9552005-3-9 (Poster).
2	Turner, S., **Hill, G. J.** (2007) *Robots in Problem-Solving and Programming,* 8th Annual Conference of the Subject Centre for Information and Computer Sciences, University of Southampton, 28th - 30th August 2007, pp 82-85 ISBN 0-978-0-9552005-7-1
3	Turner, S., **Hill, G. J**. (2008) *Robots within the teaching of Problem-Solving*, ITALICS, HEA-ICS, Volume 7 Issue 1, June 2008, pp. 108-119, ISSN: 1473-7507.
4	Turner, S., **Hill, G.,** Adams, J. (2009*) Problem Solving and Creativity for Undergraduate Computing and Engineering students***,** 9th 1-day Teaching of Programming Workshop, University of Bath, 6th April 2009.
5	Turner, S., **Hill, G. J.** (2010) *Innovative Use of Robots and Graphical Programming in Software Education*, Computer Education, Volume 9, May 2010, pp. 54-6, ISSN: 1672-5913.
6	**Hill, G. J.**, Turner, S. (2011) *Chapter 7: Problems First, Software Industry-Oriented* Education Practices and Curriculum Development: Experiences and Lessons, M Hussey, X Xu and B Wu (Eds.) IGI Global, USA, pp 110-126, ISBN: 978-1-60960-797-5.
7	Kariyawasam, K., A., Turner, S., **Hill, G**. (2012) *Is it Visual? The importance of a Problem Solving Module within a Computing course*, Computer Education, Volume 10, Issue 166, May 2012, pp. 5-7, ISSN: 1672-5913.
8	**Hill, G. J.,** Turner, S. (2014a) *Problems First, Second and Third*, International Journal of Quality Assurance in Engineering and Technology Education (IJQAETE) IGI Global, USA, Volume 3 Issue 3, pp. 88-109, DOI: 10.4018/ijqaete.2014070104, ISSN: 2155-496X, EISSN: 2155-4978.

#	Publication Title
9	**Hill, G. J.** (2015) *Review of a problems-first approach to first year undergraduate programming,* 11th China – Europe International Symposium on Software Industry Orientated Education: 29-30th April 2015, Zwickau, Germany.
10	**Hill, G. J.** (2016) *Review of a problems-first approach to first year undergraduate programming*, Software Engineering Education Going Agile: 11th China–Europe International Symposium on Software Engineering Education (CEISEE 2015): Kassel S., Wu B (Eds.) Springer, pp. 73-80, ISBN 978-3-319-29165-9.

APPENDIX 2 SUPPLEMENTARY PUBLICATIONS.

#	Publication Title
11	Minai, A., Turner, S., **Hill, G. J**. (2008) *Motivational Differences In Learning Internet Programming Between The Arts And Computing Students*, 9th Annual Conference of the Subject Centre for Information and Computer Sciences, Liverpool Hope University, 26th - 28th August 2008, p197, ISBN 978-0-9559676-0-3 (Poster).
12	Zhao, F., Turner, S., **Hill, G.**, Dravid, R., Zhang, Y. (2010) "*A Virtual Environment Training System for Haptic Laparoscopic Surgery*", 16th International Conference on Automation and Computing, University of Birmingham, Birmingham, UK, 11 September 2010.
13	**Hill, G.**, Svennevik, E., Turner, S. (2011) *"Green Computer Science Courses! We're going mobile!"*, 7th China - Europe International Symposium on Software Industry Oriented Education, 23-24 May 2011, University of Northampton, Northampton, UK. .
14	**Hill, G**., Turner, S. (2012) "*Referencing within Code in Software Engineering Education!*", Computer Education, Volume 10, Issue 166, May 2012, pp. 1-4, ISSN: 1672-5913.
15	**Hill, G.**, Turner S. (2014b) *"Chapter 5: Electronic Online Marking of Software Assignments"*, Progress in IS: Software Engineering Education for a Global E-service Economy, Motta, Gianmario; Bing, Wu (Eds.) Springer, ISBN 978-3-319-04216-9, DOI 10.1007/978-3-319-04217-6_5, pp 41-48.
16	**Hill, G.**, Svennevik, E., Turner, S. (2014) "*Computer Science Courses Using Laptops*", Innovation in Teaching and Learning in Information and Computer Science, ITALICS, HEA-ICS, Volume 13 Issue 1, June 2014, pp 1-7, DOI: 10.11120/ital.2014.00011.

APPENDIX 3ETHICAL CONSIDERATIONS FOR ENGAGING WITH PARTICIPANTS FOR INTERVIEWS, QUESTIONNAIRES.

Ethical Considerations for Researcher's for engaging with participants for interviews, questionnaires etc…	
Issues	Strategies
Preliminary papers and authority	Evaluation of learning and teaching has run concurrently with the normal professional conduct of a University lecturer. Feedback from the students has been sought formally through the student voice, using University standard feedback questionnaires. Informal anecdotal feedback has been sought throughout as part of the normal day to day role of a module leader.
Choice/recruitment of participants	University of Northampton students enrolled on the CSY1020 Problem Solving module.
Training	Standard University staff development had been given as part of the standard approaches to obtaining student feedback.
Involvement	Each participant was given the opportunity to positively decide to be involved in feedback questionnaires and the research
Rights, safety and wellbeing of participant and researcher	Standard University of Northampton procedures, facilities and conditions for completing the questionnaire were used.
Permission from immediate authorities	See above
Suitability of premises	See above
Method of interview	One paper (Kariyawasam, Turner and Hill, 2012) considered the student's perspective from research collected/collated by a student researcher under a initiative within the University (URB@N - Undergraduate Research Bursaries at Northampton). All students interviewed either had completed the module within the two years of the

	survey or were completing the problem-solving module in their first year. The methodologies adopted included interviews, questionnaires and focus groups. Questionnaires were ethically considered as part of the URB@N project.
Method of recording data	For the URB@N project interview, questionnaires and focus groups consent was obtained. The data was not retained once the project had been completed.
Interviewers	The URB@N student researcher was the interviewer.
Transcribers	No transcription was required.
Translators	No translators were used.
Attendees	Voluntary group of existing University of Northampton Computing students took part in interviews, questionnaires and focus groups
Consent	Voluntary consent was given by all participants.
Confidentiality and Anonymity	As part of the URB@N project. The researcher made it clear that the participants would not be identified in the research report. The data protection legislation was followed.
Feedback	Each Participant received a summary of the University questionnaire feedback through the normal University procedures.

APPENDIX 4STATEMENT ON THE REVIEW PROCESSES OF THE JOURNALS (OR EQUIVALENT) IN WHICH THE WORK HAS BEEN PUBLISHED.

Summary of Published Works		Supplementary Publications	
1	Peer reviewed (Abstract).	11	Peer reviewed (Abstract).
2	Peer reviewed.	12	Peer reviewed
3	Peer reviewed. Invited based on paper 2.	13	Peer reviewed.
4	Peer reviewed (Abstract).	14	Peer reviewed.
5	Peer reviewed.	15	Peer reviewed.
6	Peer reviewed. Invited based on paper 5.	16	Peer reviewed.
7	Peer reviewed.		
8	Peer reviewed. Invited to update 6.		
9	Peer reviewed.		
10	Peer reviewed.		

APPENDIX 5STATEMENTS FROM CO-AUTHORS ON THE EXTENT OF THE APPLICANT'S CONTRIBUTION TO THE RESEARCH.

Publication Number	Nature and extent of candidate's contribution	Contribution to Publication Writing (%)
1,2,3,5 (ST[1]/GH)	ST lead author, but equal contribution.	50%
4 (ST[1]/GH/JA[2])	ST lead author, but equal contribution.	33%
6,8,14,15 (GH/ST[1])	GH lead author, but equal contribution	50%
7 (KK[3]/ST[1]/GH)	KK lead author, but equal contribution	33%
11 (AM[4]/ST[1]/GH)	AM lead author, but equal contribution	33%
13,16 (GH/ES[5]/ST[1])	GH lead author, but equal contribution	33%
9, 10	GH sole author	100%
12 (FZ[6])	Minor contribution, but secured Lever Hulme Fellowship to support a visiting Post-Doctoral Fellow (the first author – FZ) to carry out this research.	20%

[1] Email 1, [2] Email 2, [3] Email 3, [4] Email 4, [5] Email 5, [6] Email 6.

Previously, in 1.2, it was stated that:

> "In all the co-authored papers, there is an admission that all the authors took an equal responsibility for the writing of the papers/publications. However, the concept of the CSY1020 Problem Solving & Programming module and the definition and implementation of PbBL, PjBL with the emphasis on visual PbBL, visual PjBL, Visual Programming (VP) and Graphical User Interfaces (GUI) were the sole contribution to new knowledge by the author" (cf. 1.2).

The following 6 emails, provide confirmation from the authors of the contribution to writing the papers.

Email 1: Contribution Confirmation from Scott Turner

From: UoN Turner
Date: Wednesday, 25 March 2015 12:46
To: Gary Hill
Subject: RE: PhD by Means of Published Works

Hi Gary

I am very happy to confirm as a co-author the contribution is as stated in the table.

Kind regards

Dr Scott Turner, BEng, PGCTHE, MSc, PhD, MIET, MIEEE, MBCS, FHEA
Associate Professor in Computing and Immersive Technologies
Deputy Subject Leader
Department of Computing and Immersive Technologies
School of Science and Technology
University of Northampton
NN2 6JD, UK

Tel: 01604 893028
website: www.computing.northampton.ac.uk/~scott
website: http://www.northampton.ac.uk/directories/people/scott-turner
Blog: http://computingnorthampton.blogspot.com/
Blog: http://apslandt.blogspot.com/
Personal Twitter: @scottturneruon
http://orcid.org/0000-0003-2735-3220

Email 2: Contribution Confirmation from Jonathan Adams

From: UoN Jonathan Adams <Jonathan.Adams@northampton.ac.uk> **Date:** Thursday, 11 January 2018 at 12:20 **To:** "Gary Hill (Work)" <Gary.Hill@northampton.ac.uk> **Subject:** Contribution to publications and workshops

Hello Gary,

Many thanks for your request.

With regards to the following publication (workshop), I would like to confirm that we all contributed equally to this paper i.e. 33.3%, 33.3% and 33.3% respectively:

Turner, S., Hill, G., Adams, J. (2009) Robots in problem solving in programming. 9th 1-day Teaching of Programming Workshop, University of Bath, 6th April 2009.

With kind regards

Jon

Dr. Jonathan Adams BEng CEng MA PhD Cert. Ed. MIET FHEA
Faculty of Arts, Science and Technology (FAST)
Head of Department: Engineering & Technology Avenue Campus
St Georges Road
Northampton
NN2 6JD
EXT: 3074
DDI: +44 (0)1604 893074]

University of Northampton

See out industry links: http://www.netpengineering.co.uk
WWW.northampton.ac.uk Follow the story on social media
http://www.northampton.ac.uk/social-media-hub/

Email 3: Contribution Confirmation from Kumuditha Achini Kariyawasam

From: Kumuditha Kariyawasam <kakariyawasam@gmail.com> **Date:** Wednesday, 10 January 2018 at 21:07 **To:** "Gary Hill (Work)" <Gary.Hill@northampton.ac.uk> **Subject:** Re: Contribution Confirmation - PhD by Means of Published Works

Hello Gary,

With regards to the following publication, I would like to confirm that we all contributed equally to this paper i.e. 33.3%, 33.3% and 33.3% respectively:

Kariyawasam, K. A., Turner, S., Hill, G. (2012) *Is it Visual? The importance of a Problem Solving Module within a Computing course*, Computer Education, Volume 10, Issue 166, May 2012, pp. 5-7, ISSN: 1672-5913.

Please you use this email as the confirmation from me as one of the co-authors in this publication.

Thank you,
Best Regards,
Kumuditha Achini Kariyawasam

Email 4: Contribution Confirmation from Amir Minai

From: UoN Minai <Amir.Minai@northampton.ac.uk> **Date:** Tuesday, 9 January 2018 at 15:30 **To:** "Gary Hill (Work)" <Gary.Hill@northampton.ac.uk> **Subject:** RE: Contribution Confirmation - PhD by Means of Published Works

Gary,

Hello.
With regards to the following publication, I would like to confirm that we all contributed equally to this paper i.e. 33.3%, 33.3% and 33.3% respectively.

Minai, A., Turner, S., **Hill, G. J**., (2008) *Motivational Differences In Learning Internet Programming Between The Arts And Computing Students*, 9th Annual Conference of the Subject Centre for Information and Computer Sciences, Liverpool Hope University, 26th - 28th August 2008, p197, ISBN 978-0-9559676-0-3 (Poster).

Regards.

Amir.

Email 5: Contribution Confirmation from Espen Svennevik

From: UoN svennevik <Espen.Svennevik@northampton.ac.uk> **Date:** Tuesday, 9 January 2018 at 14:51 **To:** "Gary Hill (Work)" <Gary.Hill@northampton.ac.uk> **Subject:** Re: Contribution Confirmation - PhD by Means of Published Works

Hello Gary,

With regards to the following publication, I would like to confirm that we all contributed equally to this paper i.e. 33.3%, 33.3% and 33.3% respectively:

Hill, G., Svennevik, E., Turner, S. (2011) *Green Computer Science Courses! We're going mobile!*, 7th China - Europe International Symposium on Software Industry Oriented Education, 23-24 May 2011, University of Northampton, Northampton, UK.

Regards

Espen

Espen J. Svennevik, MSc, MBCS CITP MIET
Senior Lecturer
Course Leader BSc Computing (Mobile Computing) Computing Division Faculty of Arts, Science and Technology, The University of Northampton, Newton Building, St Georges Avenue, Northampton, NN2 6JB, United Kingdom **t:** +44 (0)1604 893073 **e:** espen.svennevik@northampton.ac.uk **f:** +44 (0)1604 893397

Email 6: Contribution Confirmation from Fendga Zhao

From: zfd@ysu.edu.cn **Date:** Wednesday, 10 January 2018 at 03:22 **To:** Work <Gary.Hill@northampton.ac.uk> **Subject:** About the contribution

Dear Gary,

With regards to the following publication, I would like to confirm that we all contributed equally to this paper i.e. 20%, 20%, 20%, 20% and 20% respectively:

Zhao, F., Turner, S., Hill, G., Dravid, R., Zhang, Y. (2010) *A Virtual Environment Training System for Haptic Laparoscopic Surgery*, 16th International Conference on Automation and Computing, University of Birmingham, Birmingham, UK, 11 September 2010.

Best Regards,

Fengda Zhao --

School of Information Science and Engineering (School of Software),Yanshan University

APPENDIX 6PUBLISHED WORKS (FULL TEXT).

#	Publication Title	Page
9	**Hill, G. J.** (2015) *Review of a problems-first approach to first year undergraduate programming,* 11th China – Europe International Symposium on Software Industry Orientated Education: 29-30th April 2015, Zwickau, Germany.	210
10	**Hill, G. J.** (2016) *Review of a problems-first approach to first year undergraduate programming*, Software Engineering Education Going Agile: 11th China–Europe International Symposium on Software Engineering Education (CEISEE 2015): Kassel S., Wu B (Eds.) Springer, pp. 73-80, ISBN 978-3-319-29165-9.	216

Previously published content has been redacted from this thesis. Publication details and sources are available on pages 152-153.

DOI links are available for the following items:

Turner, S. and Hill, G. J. (2008) Robots within the teaching of Problem-Solving. *ITALICS.* **7**(1), pp. 108-119. https://doi.org/10.11120/ital.2008.07010108

Hill, G. J. and Turner, S. (2011) Chapter 7: Problems First. In: Hussey, M., Xu, X. and Wu, B. (eds.) *Software Industry Oriented Education Practices and Curriculum Development: Experiences and Lessons*. IGI Global, pp 110-126. https://doi.org/10.4018/978-1-60960-797-5.ch007

Hill, G. J. and Turner, S. (2014) Problems First, Second and Third. *International Journal of Quality Assurance in Engineering and Technology Education*. **3**(3), pp. 88-109. https://doi.org/10.4018/ijqaete.2014070104

Hill, G. J. (2016) Review of a problems-first approach to first year undergraduate programming. In: Kassel, S. and Wu, B. (eds.) *Software Engineering Education Going Agile: 11th China–Europe International Symposium on Software Engineering Education (CEISEE 2015)*. Springer, pp. 73-80. https://doi.org/10.1007/978-3-319-29166-6_11