



Industry Practice as a Source of Potential Values for Computing Education
– What Can Teachers and Pupils Learn from Industry Professionals

Submitted for the Degree of
Doctor of Philosophy
At the University of Northampton

2024

Aleksandra Dziubek

Acknowledgements

The completion of this thesis would not be possible without the help, support and guidance of many people.

First and foremost, I wish to express my sincere gratitude to my supervisory team – Dr Scott Turner and Dr Mark Johnson for their continued support and guidance during difficult moments, and their professional advice and constructive criticism throughout the duration of this research.

Secondly, I would like to thank all people who participated in my surveys, trials and interviews, especially computing teachers from Northamptonshire secondary schools who found time in their extremely busy schedules to talk to me.

Lastly, special thanks go to my dear family and friends, especially my parents and my husband, for their inexhaustible patience, understanding and support they have shown me throughout the years.

Abstract

This study explores the potential values coming from the practices of computing industry in the UK that could be incorporated in British schools in order to improve the quality of computing education and provide a more engaging experience for pupils and encourage more young people to pursue a career in this field, thus helping to bridge the skill gap on the job market.

In 2014, computer science returned to British schools after many years of absence – a change awaited by computing professionals, who criticised the previous ICT subject. The new, demanding curriculum brought a number of challenges that the education system needs to deal with: unprepared teachers, low popularity of computing among pupils and lack of established pedagogies in this field.

A report by The Royal Society from 2017 highlights areas of computing education in schools that need improvements and provides recommendations, including further research in this field and providing support to teachers with the help of industry, government and non-profit organisations. Findings and suggestions of this report became the main rationale for conducting this study.

An exploration of computing-specific pedagogies and educational support tools revealed a variety of both. However, what became apparent was the large number of tools for block-based programming and limited support for textual programming. This suggested a potential area of development.

Further research consisted of collecting quantitative and qualitative data from several groups of people whose input was seen as valuable: computing tutors from universities, computing industry professionals and secondary computing teachers.

The results coming from two surveys, conducted among computing tutors in 2014 and 2018, are in line with findings presented in “Next Gen” report and show that despite recent changes in schools the quality of computing education is still seen as low among people from higher education institutions.

A survey with computing professionals working in the industry helps to understand their original motivation to learn computer science, explores their current learning and problem-solving habits, and offers some advice to schools.

An analysis of #CASchat discussion on Twitter provides an insight into the world of computing educators, focusing on problems with transferring to text-based programming, ways to overcome them, as well as educators’ views on the practice of industry as a potential source of values for schools.

Finally, a series of semi-structured interviews with computing teachers from secondary schools in Northamptonshire has allowed the researcher to understand the challenges of their everyday work, see what tools and methodologies they use, and find out how much they know about industry practice and whether they see it as a source of values for computing education.

Findings coming from the activities outlined above are in line with the literature and allow to identify a number of key values based on the practice of industry professionals: teamwork and collaboration, doing practical projects to solve real-life problems, using internet resources and IDEs for textual programming and problem-solving. These values should be promoted and encouraged in schools.

Another direction that was investigated during this research was the experimental development of an IDE specifically designed for beginners. While a detailed description of the process and its evaluation is provided in Appendix 10, preliminary results from trials with computing students and teachers suggest significant promise for this approach in enhancing learning within computing education.

Table of Contents

Acknowledgements.....	i
Abstract.....	ii
List of Figures and Tables.....	vi
Chapter 1: Introduction	1
1.1. Introduction	1
1.2. Rationale	1
1.3. Aims and Objectives.....	2
1.4. Thesis Overview	2
Chapter 2: Literature Review - Computing Education in British Schools: Current Situation, Teaching Methods and Challenges.....	5
2.1. Introduction to Computing Education in Schools	5
2.2. History of Computer Science in British schools	5
2.2.1. Introduction of Computer Science in Late 60s.....	5
2.2.2. Information and Communications Technology (ICT)	6
2.3. Current Situation of Computer Science in British Schools.....	7
2.3.1. Introduction of the New Curriculum.....	8
2.3.2. Content of the New Computing Curriculum	8
2.4. New Challenges.....	9
2.4.1. Low Popularity Among Pupils	10
2.4.2. Unprepared Teachers	10
2.4.3. Royal Society Report	11
2.4.4. Non-Exam Assessments Problem (NEA)	12
2.5. Computer Science Teaching Approaches – A Literary Perspective	13
2.5.2. Converting from Visual to Text-based Programming	13
2.5.3. Use-Modify-Create.....	15
2.5.4. Rubber Duck Debugging.....	18
2.5.5. Educational Software and Courses	19
2.5.6. Support for Teachers.....	22
2.6. Successful Computing Teaching Strategies – As Per Teacher’s Perspective.....	22
2.6.1. Common Teaching Pedagogies in Literature	22
2.6.2. Teaching Pedagogies Identified by Teachers	23
2.5.1. Visual and Blocks Programming.....	24

2.7. Insight into the Practice of Industry Professionals	25
2.7.1. Stack Overflow Surveys.....	25
2.7.2. Professional Software Development Environments	26
2.7.3. Successful Computer Programmers' Stories.....	27
Chapter 3: Methodology.....	29
3.1. Problems of modern computing education in British schools.....	29
3.2. Project Aims and Objectives	30
3.3. Research Design	31
3.4. Research Activities as per Ethics Policy	32
3.5. Data Collection.....	35
3.5.1. Higher Education Lecturer Surveys.....	35
3.5.2. Teacher Interviews.....	36
3.5.3. Twitter #CASchat.....	36
3.5.4. Computing Professionals Survey.....	37
3.6. Research Analysis.....	37
3.6.1. Survey Analysis.....	37
3.6.2. #CASchat Analysis	39
3.6.3. Interview analysis.....	41
3.7. Ethical Considerations.....	41
3.7.1. Ethics Policy for Data Collection through Surveys & Trial Participants	41
3.7.2. Ethics Policy for Data Collection through Interviews.....	41
3.7.3. Ethics Policy for Data Collection through Twitter #CASchat.....	43
Chapter 4: Data Analysis	44
4.1. Introduction	44
4.2. Tutor Surveys	44
4.3. Higher Education Entry Level Computing Skills Assessment	45
4.4. Quality of Computing Education within Schools.....	47
4.5. Future Improvements to Computing Education within Schools.....	48
4.6. Respondents' Comments	49
4.6.1. The 2014 Survey.....	49
4.6.2. The 2018 Survey.....	50
4.6.3. Summary of Respondents' Comments	50
4.7. Computing Professionals' Survey.....	51
4.7.1. Your Beginnings	52
4.7.2. Learning as a Professional.....	57
4.7.3. Your Recommendations.....	59

4.7.4. Comments	60
4.7.5. Advice to Yourself	61
4.7.6. Summary of Computing Professionals' Survey	62
4.8. #CASchat	63
4.8.1. Answers' Word Clouds	67
4.9. Teacher Interviews.....	68
4.9.1. Answer Analysis	70
4.9.2. Summary of Teacher Interviews	75
Chapter 5: Reflection, Conclusions and Recommendations.....	77
5.1. Conclusions	77
5.1.1. Computing Education in Schools – Changes, Challenges and Teaching Methodologies	77
5.2. Further Research Findings	78
5.2.1. Computing Tutors' View of Computer Science Education in Schools.....	78
5.3. Implications.....	82
5.4. Limitations.....	82
5.5. Recommendations	83
5.5.1. GCSE's Practical Project	83
5.5.2. Research Ways to Promote Teamwork and Collaboration in Classroom	83
5.5.3. Research How Solving Real-Live Problems Can Improve Student Engagement and Enhance Learning Process	83
5.5.4. Develop IDE Dedicated to Use in Classrooms	84
5.5.5. Establish Closer Cooperation with the Industry	84
5.6. Concluding Summation	84
6. Appendices.....	85
Appendix 1 – The Professional Practitioners View of Computing Education in Secondary Schools (2014) – email to tutors	85
Appendix 2 - The Professional Practitioners View of Computing Education in Secondary Schools (2014) – questionnaire.....	86
Appendix 3 – The Professional Practitioners View of Computing Education in Secondary Schools (2018) – email to tutors	89
Appendix 4 - The Professional Practitioners View of Computing Education in Secondary Schools (2018) – questionnaire.....	90
Appendix 5 – Professionals' opinion about computing education – initial message	93
Appendix 6 – Professionals' opinion about computing education – questionnaire.....	94
Appendix 7 – Computing Teacher Interviews – questions	99
Appendix 8 – Computing Teacher Interviews – initial email	100
Appendix 9 – Computing Teacher Interviews – consent form	101

Appendix 10 – Integrated Development Environment for Students	102
1.1. An Integrated Development Environment (IDE) for Beginners	102
1.2. Requirement Specifications	104
1.3. System Analysis and Design	105
1.4. Implementation	109
1.5. Evaluation and Testing	112
Appendix 11 – Software Trials – list of tasks	126
Appendix 12 – Software Trials – initial email to teachers	127
Appendix 13 – Software Trials – email to teachers with further information.....	128
Appendix 14 – Software Trials – teachers’ questionnaire	129
Appendix 15 – Software Trials – students’ questionnaire	132
7. References	134

List of Figures and Tables

Table 1.1 - Thesis overview	4
Table 2.1 - Summary of Computing Curriculum on Every Key Stage	9
Table 2.2 – A summary of recommendations made by The Royal Society.....	12
Figure 1.2 - Droplet Editor: Code Created Using Blocks	14
Figure 2.3 - Droplet Editor: Blocks Converted to Text (JavaScript).....	14
Figure 2.4 - An Example Programme Written in Greenfoot's Stride Editor.....	15
Figure 2.5 - The "Use-Modify-Create" Model	16
Figure 2.6 - PRIMM approach model.....	17
Figure 2.7 - An Example "Conversation" Recorded Using Duckie.....	19
Table 2.3 - Summary of Existing Software Frameworks/Tools for Teaching Computing	22
Figure 2.1 - An example programme created using Scratch.....	25
Table 2.4 - Comparison of Professional IDEs	27
Figure 3.1 - Problems in Computing Education	30
Figure 3.2 - Project's Aims and Objectives Summary	31
Table 3.1 – Groups of people for data collection with the rationale.....	32
Table 3.2 - Research Activities that Require an Ethics Policy	34
Figure 3.3 - A Screenshot of Summary Report on eSurveyPro.com	38
Figure 4.1 – Respondents’ Fields of Computing (2014)	45
Figure 4.2 - Respondents’ Fields of Computing (2018).....	45
Figure 4.3 - Most Common Answers to Question about First-Year Students’ Skills 2014.....	46
Figure 4.4 - Most Common Answers to Question about First-Year Students’ Skills 2018.....	46
Figure 4.5 - Higher Education Entry Level Computing Skills Assessment (Score Comparison).....	46
Figure 4.7 – Most Common Answers to Statements Regarding Aspects of Computing Education in Schools 2018	47
Figure 4.6 – Most Common Answers to Statements Regarding Aspects of Computing Education in Schools 2014	47

Figure 4.8 - Opinions on Different Aspects of Computing in Schools	48
Figure 4.9 - Rating of Improvement Ideas	49
Figure 4.11 - Age of Respondents (Computing Professionals' Survey).....	52
Figure 4.10 - Gender of Respondents (Computing Professionals' Survey).....	52
Figure 4.13 - Respondents' Primary Programming Language (Computing Professionals' Survey)	52
Figure 4.12 - Respondents' Years of Experience (Computing Professionals' Survey).....	52
Figure 4.14 – Professionals' Initial Interest in Computing	53
Figure 4.15 - Who Introduced Professionals to Computing?.....	53
Figure 4.16 - How Did Professionals Learn Computing?.....	54
Figure 4.17 - What was Professionals' First Programming Language?	54
Figure 4.18 - Did Professionals have Computing Lessons at School	55
Figure 4.19 - Professionals' Rating of their Computing Lessons	56
Figure 4.20 - Popularity of educational tools.....	56
Figure 4.21 - Rating of Educational Tools	57
Figure 4.22 - Ways of Expanding Knowledge.....	57
Figure 4.23 - Popular Websites for Developers	58
Figure 4.24 - Seeking Support at Work	58
Figure 4.25 - Recommended Programming Languages	59
Figure 4.26 - Recommended Educational Tools	60
Table 5.1 - Professionals' Advice: How to Improve Computing Education.....	60
Figure 4.27 - Word Cloud Generated from Comments, Showing Most Common Words	61
Table 4.2 - Professionals' Advice: Message to Past Self.....	62
Figure 4.28 - Word Cloud: Professionals Advice to Themselves.....	62
Figure 4.30 - Chart with #CASchat Participants' Locations	64
Figure 4.29 - Map of #CASchat Participants' Locations	64
Table 4.3 - Twitter #CASchat: Questions and Summary of Answers	66
Figure 4.33 - Word Cloud of Answers to Q3	67
Figure 4.34 - Word Cloud of Answers to Q4	67
Figure 4.31 - Word Cloud of Answers to Q1	67
Figure 4.32 - Word Cloud of Answers to Q2	67
Figure 4.35 - Word Cloud of Answers to Q5	68
Figure 4.37 - Interviewees' Formal Education in Computing (Teacher Interviews).....	68
Figure 4.36 - Interviewees' Gender (Teacher Interviews).....	68
Figure 4.39 - Interviewees' Industry Experience (Teacher Interviews).....	69
Figure 4.38 - Interviewees' Teaching Experience in Years, where T-Number Represent Each Teacher (Teacher Interviews)	69
Table 4.4 - Interviewees' Profiles.....	70
Figure 4.40 - A List of Teaching Aids Enumerated by Interviewees.....	72
Table 5.1. - System Functionality and Features	105
Figure 5.1 - IDE Student Welcome Window.....	106
Figure 5.2 - IDE Create Project Window	107
Figure 5.3 - IDE Main Editor Window.....	107
Figure 5.4 - IDE Duck Debugging Window	108
Figure 5.5 - IDE Email to a Friend Window	108
Figure 5.6 - IDE email to teacher window.....	109
Figure 5.7 – Logo of the Software.....	109
Figure 5.8 - Workspace Window.....	110
Figure 5.9 - Create New Project Window	110

Figure 5.10 - Main Editor Window with Correct Compiled Code	111
Figure 5.11 - Talk to the Duck Window.....	111
Figure 5.12 - Send Email (to Friend) Window	112
Figure 5.13 - Send Email (to Teacher) Window	112
Table 5.2 - IDE's Testing Summary.....	113
Table 5.3 - A Summary List of the Tasks in Test Scenario	115
Figure 5.14 - Is Software Like this Suitable for School Education.....	116
Figure 5.15 - Can Software Like this Significantly Improve Learning Processes	116
Figure 5.16 - Can Software Like this Encourage Students to Learn Programming	117
Figure 5.17 - Rating of IDE's Features	117
Figure 5.18 - Students' Favourite Feature	118
Figure 5.19 - Is Industry Experience Valuable in Learning Programming	118
Figure 5.20 - Can Software Like this Give Students Industry-like Experience	119
Figure 5.21 - Can Software Like this Give Students Industry-like Experience - Average Answer	119
Table 5.4 - Student participants' comments.....	120
Figure 5.22 - Would Teachers Consider Using Software Like this in Their Classroom.....	121
Figure 5.23 - Can Software Like this Significantly Improve Learning Process.....	121
Figure 5.24 - Can Software Like this Encourage Students to Learn Programming?	121
Figure 5.25 - Teachers' Rating of IDE's Features.....	122
Figure 5.27 - Pupils' Favourite Feature, as Predicted by Teachers	122
Figure 5.26 - Teachers' Favourite Feature	122
Figure 5.29 - Can Software Like this Give Students Industry-like Experience	123
Figure 5.28 - Is Industry-like Experience Valuable in Learning Programming	123
Figure 5.30 - Teacher participants' comments	124

Chapter 1: Introduction

1.1. Introduction

A new study was commissioned by the Royal Society in the year 2016 by Microsoft and Google in order to determine the challenges that educators face while delivering the studies of computer science and computing, with an aim to discovering best practices that are widely accepted and have the capability of being adopted with ease. This shows the effort that Britain is making in creating strong teaching and learning pedagogies for computing education to strengthen the curriculum dissemination practices that are currently prevalent in British Secondary schools. For the first time, the new computing curriculum for students between 5 and 16 years old in England has been established as a foundational subject in addition to Mathematics, Science and English. Moreover, the government has undertaken several initiatives to keep computer science as an integral part of the STEM discipline (Crick, 2017).

In such a case, the input of industry professionals and computing education teachers becomes quite necessary to explain the potential this new curriculum holds and how it can enhance ICT education in the UK (Crick, 2017).

Following the publication of two reports: “Next Gen” (Hope and Livingstone, 2011) and “Shut down or Restart? The Way Forward for Computing in UK Schools.” (The Royal Society, 2012), which discussed the skill gap in computing industry and criticised computing education offered in schools, the British government decided to re-introduce computer science to the curriculum (Department for Education, 2014), after years of absence. Since then, pupils in primary and secondary schools have had the opportunity to learn this rigorous discipline, including things like: computer hardware, networking, security, programming using blocks and later using a textual language.

This awaited revolution, however, brought a number of challenges that schools need to face in order to meet the needs of the demanding curriculum. Reports (BCS, The Tech Partnership, 2016; The Royal Society, 2017) published in the years after the introduction of computer science allowed to identify these challenges:

- low popularity of the subject among pupils
- unprepared teachers, without the right skills to teach the subject
- lack of established pedagogies

Bringing computer science back to British schools was undoubtedly a necessary move, but it should be considered only as the first step towards better computing education and closing the skill gap which arose in the industry. More research is required to address the challenges and help to improve the quality of education offered in schools.

1.2. Rationale

As outlined in the previous section, computing industry and its problem with finding skilled candidates for jobs, was a key factor that triggered the revolution in schools and began to change how computing education is perceived. Many positive changes were made during the last 10 years, especially in British schools, yet computing education still requires more development to meet all expectations (Wohl, 2017).

The 2017 report “After the Reboot: Computing Education in UK Schools” (The Royal Society, 2017) calls for help from computing industry and organisations, as one of the ways to support teachers who struggle with delivering the contents of the new curriculum. Authors of the report also see the need for further research in this field in order to address the issues that they have described.

In the light of these facts, it seems justified to research and identify the values coming from industry practice that could be used to help computing teachers to deliver high standard education and encourage more pupils to learn this subject and perhaps enter a computing career path in the future.

1.3. Aims and Objectives

This research aims to answer the following question: *Can the computing industry assist in improving the quality of computing education taught in British schools?*

In search for possible answers, the following essential project objectives have been formulated:

- 1) Understand the situation of computing education in British schools and observe the changes taking place in the years following the introduction of the new curriculum; explore current educational practice within schools for the computing discipline to identify any existing deficiencies and strengths present within the current practice.
- 2) Undertake a comprehensive exploration of methodologies and contemporary technologies that aim to support more effective learning of computer science.
- 3) Conduct qualitative research to gain in-depth insight into computing teachers’ everyday problems and the methodologies they use; to determine their outlook on the practice of the industry and its potential value to schools.
- 4) Gather and analyse quantitative data on how industry professionals used to learn in the past, as well as how they learn and solve problems on everyday basis; to identify practices that could be incorporated into this work.
- 5) Review findings and provide a set of recommendations

1.4. Thesis Overview

The overview of the thesis is outlined in a table format for simple and clear presentation. The left column contains a summary of chapters 2 to 6, with their key findings enumerated in the right column.

Chapter Summary	Key Findings
<p>Chapter 2: Explores the situation of computing education in British schools, starting from a brief historical background leading to the recent transformation and challenges that it brought. A number of teaching methodologies and educational tools aimed at supporting computing education are reviewed. Finally, the chapter provides an insight into the practice of the industry based on Stack Overflow surveys, reviewing tools</p>	<ul style="list-style-type: none"> • There is a number of challenges that schools need to face after the introduction of the new computing curriculum. • Computing industry was one of the triggers behind the transformation of computing in schools. • It is expected that the industry will continue to support computing education. • There is a number of teaching methodologies and educational tools available, however support for text-based programming seems to be very limited. • A great number of professional programmers use advanced IDEs in their work.

<p>used by professionals and examining the stories of successful programmers.</p>	<ul style="list-style-type: none"> • Famous computing programmers were initially driven by curiosity and learnt by solving real-life problems.
<p>Chapter 3: Formulates the aims and objectives of the research, based on the findings from the literature review. Provides rationale for further research and outlines the methodology that was used in order to find answers to the key research questions. Lastly, it explains the quantitative and qualitative methods used to analyse the data that was collected.</p>	<ul style="list-style-type: none"> • Aim of the research becomes to find what value can be transferred from industry practice to schools. • Research activities are defined as: surveys among computing tutors and industry professionals, analysis of #CASchat, interviews with secondary computing teachers and software trials. • Suitable methods of analysing the data are chosen.
<p>Chapter 4: Presents quantitative analysis of data collected during surveys with computing tutors and computing professionals from the industry, and qualitative examination of #CASchat discussion on Twitter and a number of semi-structured interviews with secondary computing teachers.</p>	<ul style="list-style-type: none"> • Computing tutors have a negative outlook on the education delivered in schools, however, they see the new curriculum as a positive change, allowing to better prepare pupils for further education or work. • Tutors seem to recognise the role of computing industry in school education, but they do not agree on what value it could bring. • Industry professionals were also driven by curiosity when they started learning about computers. One of their main sources of education is university, rather than school. • Professional software developers rely mainly on the internet (especially Stack Overflow) and collaboration with others in solving problems at work and learning. • Industry professionals recommend that schools should make computing more fun, by doing practical projects based on real-life problems. They also encourage learning from different resources (websites, other programmers) and using support tools. • Educators participating in #CASchat agree that pupils struggle with transition from block- to text-based programming and name a couple of reasons behind it: more demanding syntax, poor typing skills and text being less appealing and entertaining than blocks. • Computing educators believe that what schools could learn from the industry is teamwork, collaboration and using real-life problems. • Educators named a few features that could help pupils with text-based programming: syntax checking, friendly error messages and switching between blocks and text. • Computing teachers struggle with lack of time to learn and deliver the demanding curriculum. • They complain about the curriculum putting too much focus on theory, rather than practice, especially after the changes to non-exam assessments. • Teachers named a few tools that they use in the classroom, mainly Raspberry Pi, but admitted that they do not use any particular methodology. • They agree that a user-friendly IDE with syntax highlighting and meaningful feedback could be a great help to their pupils.

	<ul style="list-style-type: none"> • Teachers with no industry background have little knowledge of industry practice, but are very eager to meet professionals and learn from them. • Most of the teachers see using the internet and collaboration as important part of problem-solving, as this is what is done in the real-world.
<p>Chapter 5: Reflects on the main findings of the project and presents the conclusions that have been made based on them. It attempts to answer the key questions of the thesis, namely, what is the value in the practice of computing industry that could be transferred to schools in order to improve the quality of education. The chapter concludes with providing several recommendations for further research in this field and for improving computing education in schools by learning from the industry practice.</p>	<ul style="list-style-type: none"> • There are several values that schools can learn from the practice of industry: <ul style="list-style-type: none"> – teamwork and collaboration, – doing practical projects based on real-life problems, – allowing pupils to use internet resources. • An educational IDE for beginners could not only help to remove the syntax barrier in learning text-based programming, but also promote the values identified in this research. • Recommendations include: <ul style="list-style-type: none"> – improving GCSE Practical Project and finding ways of fair assessment that could re-enable it to count towards the grades – researching methods to promote collaboration and teamwork in classroom – developing an IDE dedicated for beginners – looking for ways to enable closer collaboration between schools and computing industry

Table 1.1 - Thesis overview

Chapter 2: Literature Review - Computing Education in British Schools: Current Situation, Teaching Methods and Challenges

2.1. Introduction to Computing Education in Schools

This thesis discusses problems within the field of computer science in British schools – a field that is not only very vast, but also rapidly changing and evolving, especially in the recent years, making it a “hot topic” (Department for Education, 2013; The Royal Society, 2017; Robertson, 2018). Therefore, there are many aspects of it that require initial research and introduction, allowing the reader to gain a better understanding of the current situation of computing education in British schools.

The aim of this chapter is to provide a good overview of computing education in British schools – starting with a brief historical outline, leading to the current situation, including recent reforms and their implications.

Furthermore, this chapter also contains a summary of the most popular learning theories, teaching methodologies and various educational software programmes designed to help in teaching computing education. In modern education, these elements play a key role in introducing young people to computer science.

The final section provides an insight into the industry-based professionals’ world: how they learned coding and other aspects of computing education, what tools they use in everyday work and how they continue to learn. It also includes the stories of several well-known computer professionals, creating an understanding of what made them learn computing and become experts in this field.

2.2. History of Computer Science in British schools

Learning about the past is very important as it helps in gaining a wider perspective of the subject, understand what factors contributed to the present situation and what teaching methods were originally used. Therefore, it is essential to trace the development of this subject. By learning when and how computing became a part of the British education system we can gain valuable insights into the challenges currently faced by schools and educators.

The following sections will provide a concise overview of key milestones in the history of computing education within British schools.

2.2.1. Introduction of Computer Science in Late 60s

The early history of computing in Great Britain was outlined by Professor Roger Boyle and Doctor Martyn Clark from the University of Leeds in their conference paper “Computer Science in English High Schools: We Lost the S, Now the C Is Going.”. This section provides a summary of this history based on their paper.

As computer science began its dynamic post-war development, it was also introduced in British universities. At first it was only accessible to pioneer computer building projects in Manchester and Cambridge. Then in the year 1964, obtaining a degree in computing became generally available to students outside the pioneering projects at Manchester and Cambridge. Only five years later, in 1969, the computer science subject was introduced in British high schools. The first syllabus contained, among others, the following topics:

- number systems
- problem specification and analysis
- computer hardware
- analogue computing
- numerical methods
- data processing

At that time, this was very consistent with university curricula which contained similar topics. Another very positive fact about the original school syllabus, is that it required pupils to not only study theory, but also put their knowledge into practice by writing simple programmes and properly documenting them.

The main problem that schools faced was the lack of equipment for the students to be able to do their practical work and prepare projects. Therefore, in 1981 a new instruction sent to the headmasters was introduced to the syllabus: they were required to gain approval for their computing facilities/arrangements before they could enter students for this subject. As per of an evaluation of the course, a common practice was to send programmes written by pupils to local universities or local government facilities. Despite the process being very slow – running the code and getting the output could take days – it was assuring that the pupils’ experience was consistent with high university standards.

Another problem was shortage of staff with relevant computing qualifications and experience, which was noted in 1982 at a meeting of computer science examiners in Cambridge. In the same year, the first personal computers were introduced in British schools, which began the era of Information Technology (Boyle and Clark, 2006).

2.2.2. Information and Communications Technology (ICT)

With the development of new technologies and especially the invention of microprocessor in 1971, universities were able to gradually replace their large and slow machines with small, relatively inexpensive and much more powerful computers (History.com Staff, 2011). Soon, they also became available to schools. One of the first microcomputers – the BBC Micro – was introduced in British schools in early 1980s by the joint initiative of BBC and UK Government Computer Literacy project. Within a decade it laid the foundations for the British computer gaming industry and contributed to the establishment of ARM Ltd – world’s leading processor manufacturer (The Royal Society, 2012; Norbury, 2016).

Boyle and Clark (2006) point out that when computers started to become available to non-professional users, especially in office work, a skill gap slowly became visible. As the computing industry was blooming, in early 1990s microcomputers were replaced by Personal Computers, designed mostly for business and office use. Soon the skills that pupils were able to acquire since 1970s were no longer enough, as the need for skills in the use of office packages (e.g. spreadsheets and databases) began to grow. The government responded in introducing a new school subject – Information Technology (IT), which was enthusiastically received. It was much easier to find teaching staff for IT subject, because it did not require a degree in computing. The number of pupils registering for an A-level in IT began to rapidly grow (Boyle and Clark, 2006), while the numbers for computer science were decreasing.

As the popularity of IT was going up, in the general opinion the border between computer science and IT slowly began to fade and the two disciplines were seen as strongly overlapping. Agencies offering IT qualifications contributed to spreading this incorrect belief, by using the phrase 'qualification in computing' in their promotional material (Boyle and Clark, 2006). This led to even higher popularity of IT in schools, where it was taught under the title of computing by people with strong IT skills, but little knowledge about computer science.

Universities, which continued to teach computing, have been trying to change the general opinion. Their efforts, however, remained unsuccessful (The Royal Society, 2012) as most of the new students had the wrong idea about computing and were unprepared for what they would need to learn during the course. This was one of the reasons why more computing students never finished their course, in comparison to other subjects (Boyle and Clark, 2006; The Royal Society, 2012)

2.3. Current Situation of Computer Science in British Schools

In the recent years, more and more key computing industry figures have begun to publicly raise their concerns about the problem with computing education in British schools and requesting a change. During a lecture at Edinburgh Television Festival, Google's Chairman Eric Schmidt pointed out that "[the UK's education curriculum] focuses on teaching how to use software, but gives no insight into how it's made." (Edinburgh International Television Festival, 2011).

In 2010 Alex Hope, the CEO of VFX studio Double Negative, and Ian Livingstone, a key figure in the British gaming industry, were asked by the government to review the current needs of the industry. Their report "Next Gen" (Hope and Livingstone, 2011), released a year later, revealed that there are not enough skilled candidates available for jobs in this sector and blamed the national curriculum and schools for the situation. In the report Ian Livingstone states clearly:

"The games industry is highly competitive and the industry needs the best talent with hard skills: world-class computer scientists and artists. Quite simply, there are not enough of them in the UK. In recent years the UK has slipped from 3rd to 6th in the world development league. The education system is failing to produce the talent of the calibre required. There is a generation of young people who are passionate about playing games, yet they don't know that a development industry is well established in the UK, or which subjects they need to pursue a career in the industry. Now is the time to invest in talent by equipping them with skills for the digital age. This is not about additional funding. It's about re-directing existing resource to have the right mix of subjects to prepare our children for a digital world and its creative and commercial opportunities." (Hope and Livingstone, 2011)

The authors also believe that "(...) it is schools which equip young people with the knowledge and skills foundation on which universities build." Unfortunately, according to their data, "(...) school leavers are poorly prepared [for university courses]" and "(...) STEM skills of school-leavers are inadequate". (Hope and Livingstone, 2011).

The report ends with a set of recommendations that in the opinion of its creators, would help to change the situation. At the top of their list was the request to bring back computer science to schools as an obligatory subject. Other recommendations included recruiting better teachers, providing them various forms of support, encouraging pupils to follow the computing career path and engaging with STEM (Hope and Livingstone, 2011).

In 2012 another report – "Shut Down or Restart? The Way Forward for Computing in UK Schools." - produced by The Royal Society came up with similar conclusions. It was stated that there was clearly a problem with delivering good quality computing education in British schools. Although the ICT subject was quite broad and did allow the teachers to introduce pupils to computer science, the way

it was usually delivered was seen as boring and discouraging. Young people would usually finish school with only so-called 'digital literacy' skills (e.g. using word processor or databases), as there were not enough qualified teachers who would be able to teach them anything more beyond that.

Another important finding of this report was that the British education system did not understand what computer science really is and how important it is to introduce pupils to this discipline. It was stressed that every young person should be able to learn Computing Science at school.

This report also provided several recommendations, marked as urgent. The very first point said that the National Curriculum should be revised in order to split the ICT into two different subjects, one of them being computer science. The updated curriculum should ensure that pupils will learn all of the most important aspects of computer science. Other recommendations focused on the teachers – encouraging computing graduates to become teachers, employing staff with adequate skills and helping them to improve their qualifications. Schools should also provide more suitable teaching resources (The Royal Society, 2012).

2.3.1. Introduction of the New Curriculum

The British government listened to these voices and in January 2012 the Education Secretary announced the plans to work on a new National Curriculum that would bring back computer science to schools. In the press release the ICT subject was criticised as being boring and irrelevant. The Education Secretary also promised to support the teachers' development in order to help them deliver better content (Department for Education and The Rt Hon Michael Gove MP, 2012).

Prior to releasing a new curriculum, the government launched a set of public consultations. In February 2013 a consultation began on replacing the ICT subject with computer science in the new National Curriculum (Department for Education, 2013a). The participants were asked whether the name of the subject should be changed from ICT to Computing in order to reflect the latest changes to the module. The largest number (39%) of the respondents answered "Yes". Other responses were "no" – 35% and "not sure" – 26%. The change was also motivated by the 'bad reputation' of ICT. Following the results, the government decided to rename the subject to Computing.

Draft programmes of study for Computing were prepared in cooperation with industry experts from the British Computing Society and Royal Academy of Engineering, and based on the report "Shut Down or Restart?" by the Royal Society. The subject's content was divided into three domains:

- computer science – a rigorous academic discipline, which includes programming
- digital skills – ability to effectively use computers
- information technology – the design and application of digital systems to meet users' needs

(Department for Education, 2013b).

The new National Curriculum was completed and introduced to schools in September 2014 (Department for Education, 2014).

2.3.2. Content of the New Computing Curriculum

Since the release of the new curriculum, Computing has become an obligatory subject during all 4 key stages. The following table summarises the teaching content which is laid out in the current curriculum for each stage (Department for Education, 2014):

Key stage	
1	<ul style="list-style-type: none"> • Introduction to algorithms • Creating and debugging simple programmes • Predicting the behaviour of simple programmes • Digital content • How information technology is used beyond school • Basics of internet safety and security
2	<ul style="list-style-type: none"> • Creating and debugging programmes that accomplish specific goals; solving problems by breaking them into smaller parts • Using sequence, selection, and repetition; variables, input and output • Explaining simple algorithms; detecting and correcting errors in algorithms and programmes • Computer networks, including the internet; web services, such as the World Wide Web; communication and collaboration • Effective search, understanding how results are selected and ranked; evaluating digital content • Using a variety of software on a range of digital devices to design and create programmes, systems and content that accomplish given goals, including collecting, analysing, evaluating and presenting data and information • Internet safety and security: recognising unacceptable behaviours
3	<ul style="list-style-type: none"> • Creating computational abstractions that model the real-world problems and physical systems • Understanding several key algorithms that reflect computational thinking; comparing different algorithms for the same problem • Using minimum 2 programming languages (at least one textual) to solve computational problems; using data structures; • Designing and developing modular programmes that use procedures or functions • Simple Boolean logic and its uses in circuits and programming; binary numeric system and binary operations • Hardware and software and communication between them • How instructions are stored and executed within a computer system; how data can be represented and manipulated digitally, in the form of binary digits • Creative projects using multiple applications to achieve challenging goals, including collecting and analysing data and meeting the needs of known users • Creating, reusing and modifying digital artefacts for a given audience • Internet security – data protection, recognising inappropriate content
4	<ul style="list-style-type: none"> • Developing capability, creativity and knowledge in computer science, digital media and information technology • developing and applying analytic, problem-solving, design, and computational thinking skills • how changes in technology affect safety, including new ways to protect online privacy and identity, and how to report concerns

Table 2.1 - Summary of Computing Curriculum on Every Key Stage

2.4. New Challenges

The new curriculum undoubtedly was a revolution to British schools. Re-introducing computer science after decades of absence was perhaps the most awaited change that it brought. However, the beginnings are very often not easy. Something that was seen by the industry as a solution to poor computing skills problem, introduced many new challenges that the schools and the government will need to face.

2.4.1. Low Popularity Among Pupils

In December 2016, another research was published by BCS and the Tech Partnership. Their “Women in IT Scorecard” report (BCS, The Tech Partnership, 2016) illustrates the current situation of Computing in schools and in the industry, focusing on the problem with attracting girls to this field of science.

The economic statistics show that the computing industry’s performance is excellent: the number of digital enterprises increases almost two times faster than business enterprises, resulting in a much higher employment growth, compared to all industry sectors.

Despite these optimistic statistics, computing is not a popular subject among pupils and students. The report admits that there are many initiatives aiming to promote IT as an academic or career choice, particularly for girls, however, not much has changed since 2015 and this field remains unattractive to young people. The problem is especially serious among girls: the percentage of female pupils taking IT-related GCSE exams continues to drop, only 15% of applicants of computing courses at universities are female (BCS, The Tech Partnership, 2016).

Several months later, BCS published an article (BCS, The Tech Partnership, 2017) expressing their concern over stagnation of the number of applications for computer science GCSE. According to the information released by Ofqual (The Office of Qualifications and Examinations Regulation) there is a significant reduction in the growth of young people willing to take computer science GCSE in 2017, compared with 2016. The Director of Education at BCS, Bill Mitchell, described this as “deeply worrying”, as he would expect the popularity of computer science GCSE to rapidly increase (BCS, The Tech Partnership, 2017).

2.4.2. Unprepared Teachers

In August 2014, just a month before the release of the new curriculum, an article was published in *Computer Weekly* online magazine (Bateman, 2014) – “The UK’s New Computing Curriculum Is Here: Are Teachers Ready?”. The author, Kayleigh Bateman, refers to the results of a survey that was conducted a year back showing that the secondary teachers were unprepared for the new curriculum. 74% of the teachers were certain they did not have the right skills required to teach computer science. What was even more worrying was that over half of the teachers believed that their pupils had more knowledge in this field than them.

The article lists several initiatives to support and prepare the teachers for the new curriculum that were started during the preceding year. The teachers were offered help by Microsoft, Computing at School (CAS) or Codecademy, as well as many others. However, despite a huge interest in training sessions for teachers, it was stated that “It will take some time for them to get up to speed with IT and learn how to keep up to date. It will take some time to embed the new curriculum.” (Bateman, 2014).

This situation did not seem to change even after 3 years since the introduction of the new curriculum. In his article, Benjamin Wohl stated that children are confused with the new curriculum’s contents which is due to the fact that their teachers, who formerly taught ICT, are now required to teach different content without much training and support (Wohl, 2017).

As mentioned before, BCS expressed their concern over stagnation of number of computer science applications. They also believed that there is an issue with the teachers who do not have the relevant skills and who struggle to effectively teach the subject. Bill Mitchell stressed the need for professional development for teachers, as even 70% of them may be unable to teach computer science at GCSE level (BCS, The Tech Partnership, 2017).

A 2017 research from BJSS (IT and business consultancy) and YouGov (BJSS, 2017) showed that teachers still did not feel confident about teaching computer science. According to the survey, 67% of them stated they do not have the right skills or tools to do their job effectively and 83% believe that more training from the Department of Education is required. Many of the teachers (39%) also reported that they do not have the access to adequate IT and software enabling them to teach programming. 76% of the respondents would like to see more engagement from technology companies, which could help them with that issue (BJSS, 2017).

2.4.3. Royal Society Report

In November 2017, the Royal Society released a report entitled “After the Reboot: Computing Education in UK Schools”. The authors identified several problems that are present in the reformed computing education and presented a list of new challenges which arose following the release of the new curriculum in 2014. The table below summarises these issues and the authors’ recommendations how the government should address them.

Challenge	Recommendations
The highest gender imbalance of all STEM subjects	<ul style="list-style-type: none"> • Research projects should look for ways to increase female participation in computing • The government and industry-funded interventions should evaluate their impact on improving the gender balance and prioritise it
Low uptake of computing by 14-16-year-old pupils	<ul style="list-style-type: none"> • School governors and Ofsted should monitor if and how computing education is delivered to all pupils • Government in cooperation with other parties should ensure that the available qualifications include pathways that are suitable for all pupils • The learning societies should establish a curriculum committee that would advise the government
Not enough computing teachers	<ul style="list-style-type: none"> • The government should provide quality-assured conversion courses for existing teachers • The government, higher education providers and the British Computer Society (BCS) should work together to develop pre-service subject content courses, allowing more people from different backgrounds to become computing teachers • Higher education providers should promote teaching careers • Industry and academia should support so-called braided careers (simultaneous work in 2 sectors)
Not enough support for teachers	<ul style="list-style-type: none"> • The government and the industry need to improve continuing professional development for computing teachers and provide financial support for schools allowing the teachers to attend professional development opportunities. • The industry and non-profit organisations should work together with BCS and STEM Learning to offer teaching support to teachers and schools.
Availability of computing education research	<ul style="list-style-type: none"> • Researchers, research funders, teachers and policymakers should develop a plan to achieve several key goals: <ul style="list-style-type: none"> ○ creating a long-time research agenda for computing education, ○ stakeholders’ commitment ○ strengthening of UK research capacity ○ knowledge sharing between teachers and researchers

- | | |
|--|--|
| | <ul style="list-style-type: none"> • Education research funders (primarily the Economic and Social Research Council) should address the research priorities identified in the report. |
|--|--|

Table 2.2 – A summary of recommendations made by The Royal Society

In chapter 5 of the report “Improving computing education through research” the authors emphasise the role of research in computing education. So far, researchers’ main focus was higher education and now there is a need for school education research as well.

The authors noticed that pedagogies for computing are less developed because this discipline is relatively new. They believe that new teaching methods could enable teachers to do their job more effectively and as a consequence increase the popularity of this subject among pupils (The Royal Society, 2017).

2.4.4. Non-Exam Assessments Problem (NEA)

Originally, when GCSE computer science qualifications were introduced to schools, there were 2 ways of evaluating students’ knowledge and skills:

- exams, worth 80% of the marks
- non-exam assessment (NEA), worth 20% of the marks

During the NEA, students were required to complete a practical task set by the exam board, under controlled conditions and within maximum 20 hours. Unfortunately, following the incidents in 2017, when the tasks were leaked to the Internet, a decision had been made by Ofqual (Office of Qualifications and Examinations Regulation) to change the assessment process. According to the new regulations, the NEA remains a part of the computer science assessment, however, it no longer counts towards the final grade, which will be based on exam results only (Office of Qualifications and Examinations Regulation, 2018).

This decision was subjected to criticism among software engineers and educators. A news article in *Schools Week* (Robertson, 2018) quotes several people of different backgrounds who have a negative opinion about the new regulations and computer science GCSE in general. A software engineer was especially disappointed as he believed that exams were “a useless way to assess people’s programming skills”. Educators from both schools and universities were unhappy about the way the subject was designed and assessed, suggesting that it needs a complete redesign (Robertson, 2018).

Ofqual defended its decisions by stating that the NEA was not a fair and manageable way to evaluate students’ knowledge. Despite running a consultation on this topic, they were unable to find a better solution than removing the NEA in order to prevent malpractice. At the same time, the chief regulator at Ofqual, Sally Collier, stressed that practical programming skills were considered as very important aspect of computer science GCSE, but they need to be assessed in a valid and effective way (Whittaker, 2018).

NEA first came to light as a programming project and was presented in a news article on Computing at one of the school’s websites. The experiences of CAS members showed that some students and their parents had a negative attitude towards the NEA, believing it was pointless (Woollard, 2018). The author published the article in the hope to help convince them otherwise. The programming project should be considered as a valuable contribution to the learning process and a way of preparing for the exams. Practical work is essential for the students to gain a real understanding of the subject and can improve their performance during the exams because there is a considerable overlap in the knowledge required for both teaching and learning techniques (Woollard, 2018).

2.5. Computer Science Teaching Approaches – A Literary Perspective

Computing is a relatively young, yet rapidly developing field of science. As a school subject it is even less mature – it lacks well-established pedagogies and struggles to keep up with the latest changes. Furthermore, many teachers do not have relevant skills and are simply confused about how they should teach this subject.

Addressing these challenges requires a comprehensive understanding of available pedagogical tools and techniques. Through critical evaluation, we can identify areas requiring additional support for both educators and pupils.

This section explores existing pedagogies, strategies, and approaches for teaching computer science in secondary schools. It will also highlight the how programming is deeply embedded in the computing education learning. It is an integral part that teachers must incorporate as per the new curriculum.

2.5.2. Converting from Visual to Text-based Programming

Block-based programming languages have got many advantages that make them a popular learning tool for beginners. However, as blocks are unsuitable for complex projects (Kölling et al., 2017) and therefore not used by professionals, it is necessary to eventually move to text-based programming. The new computing curriculum in Britain requires pupils to learn at least one textual programming language, starting from Key Stage 3 (Department for Education, 2013).

Unfortunately, switching from blocks to text is usually challenging for students and teachers (Kölling et al., 2015). A series of experiments from 2005 and 2006 showed that students who began their programming education using VPL Alice struggled with the transition to Java or C++. Both weak and strong students were overwhelmed by the syntax and found it hard to map Alice code to textual code. They also lost their confidence and began to question learning with Alice in the first place as they no longer saw it as ‘real programming’ (Powers et al., 2007).

2.5.2.1. Blocks and Text View

One of the popular approaches to overcoming this problem and make the transition easier for students is using a development environment that allows to toggle between blocks and text. Users can start by building their programmes with blocks that they are accustomed to. Then they can click a button to view the text version of their programme and edit it (e.g. in JavaScript). There are several editors that offer this functionality – for example, Blockly from Google and Droplet (Bau, 2015).

Researchers from the Victoria University of Wellington developed and evaluated a graphical editing environment for Grace programming language that supports the transition from blocks to text. The editor, called Tiled Grace, at first glance resembles Scratch and other VPLs. However, in this case the blocks contain full textual syntax of Grace, allowing the users to gradually become familiar with it. Using the full textual syntax has got one more advantage – the creators were able to implement a smooth visual transition, where blocks gradually fade out, leaving only the text that was written on them. Other functionalities include: error reporting, overlays with information, type checking and hints. An experiment conducted by the authors showed that the majority of users enjoyed using the tool and liked the new way of reporting errors. They also found it very helpful that the blocks were an exact equivalent of the text version, allowing them to easily switch between both views (Homer and Noble, 2014).

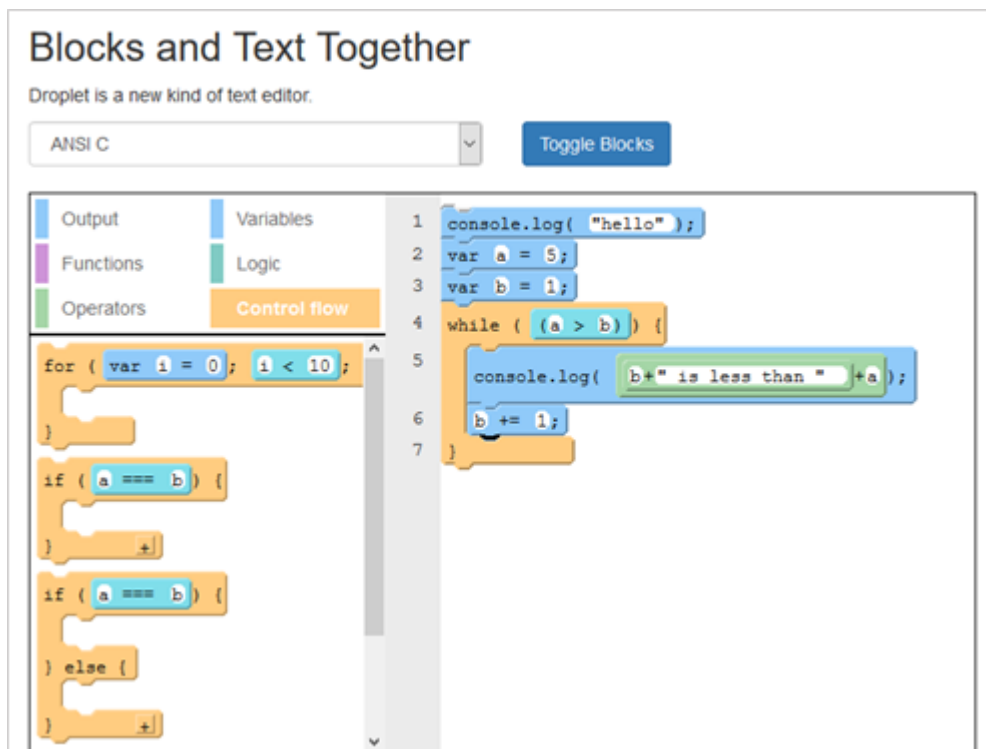


Figure 1.2 - Droplet Editor: Code Created Using Blocks

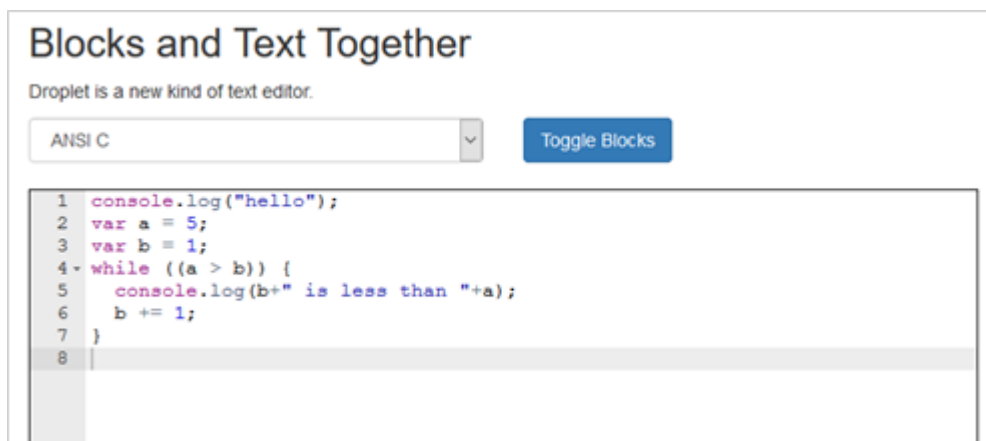


Figure 2.3 - Droplet Editor: Blocks Converted to Text (JavaScript)

2.5.2.2. Frame-based Editing

Researchers from the University of Kent proposed a different approach to ease the transition. They analysed the pros and cons of both block- and text-based languages; then developed the concept of *frame-based editing* that would combine their best features: error avoidance and discoverability of block, as well as flexibility, manipulation efficiency and keyboard control of text. Moreover, code written using the frame-based approach should remain readable on both small- and large-scale (Kölling et al., 2015).

The implementation of this concept is a new, Java-like programming language called Stride that was incorporated into Greenfoot IDE. In a Stride editor statements, such as methods or if-statements, are displayed as frames (boxes with distinct background colour) that represent their scope. This can help in reducing the number of syntax errors made by novices, for example forgetting the closing bracket. The authors conducted an experiment with two groups of students in order to evaluate the concept

of frame-based editing. All participants were given the same task (creating a video game), but one group had to complete it using Java and the other – using Stride. While there was no difference in user satisfaction, the results showed that the Stride group’s progress was significantly faster and they spent less time on fixing syntax errors (Price et al., 2016).

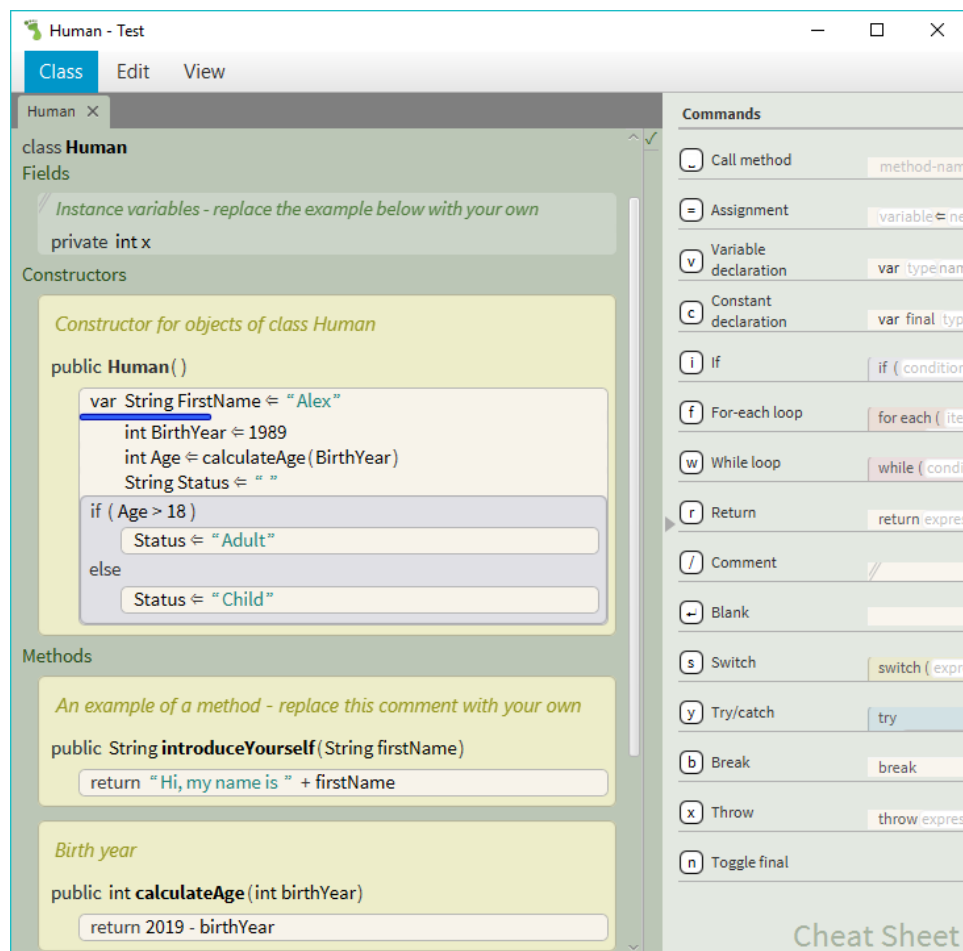


Figure 2.4 - An Example Programme Written in Greenfoot's Stride Editor

2.5.3. Use-Modify-Create

Teaching computational thinking is an important part of computing education. The term was created by Janette Wing in order to describe a set of thinking skills that are required to formulate and solve problems (Wing, 2006) – something especially useful in computer science. An article in ACM Inroads magazine (Lee et al., 2011) explores how computational thinking looks like in middle and high schools and how it can be supported. The authors came up with the “use-modify-create” (UMC) framework to engage young people in computational thinking. The framework divides student progression in learning to programme into three stages (Lee et al., 2011):

1. USE – at the beginning, students are just users of a programme written by someone else (e.g. a computer game or a programme to control robots). They learn how the programme works from a user’s perspective and experiment with its features.
2. MODIFY – students gradually begin making their own modifications to the programme, starting from a fairly simple level (e.g. changing colours) to more advanced levels (e.g. modifying

behaviour). Through a cycle of modifications and improvements, they understand more things and gain new skills.

3. CREATE – finally, students reach the stage where they are capable of creating their own programme, using the process of refine, test and analyse. In this final step, they should be able to present all key aspects of computational thinking.

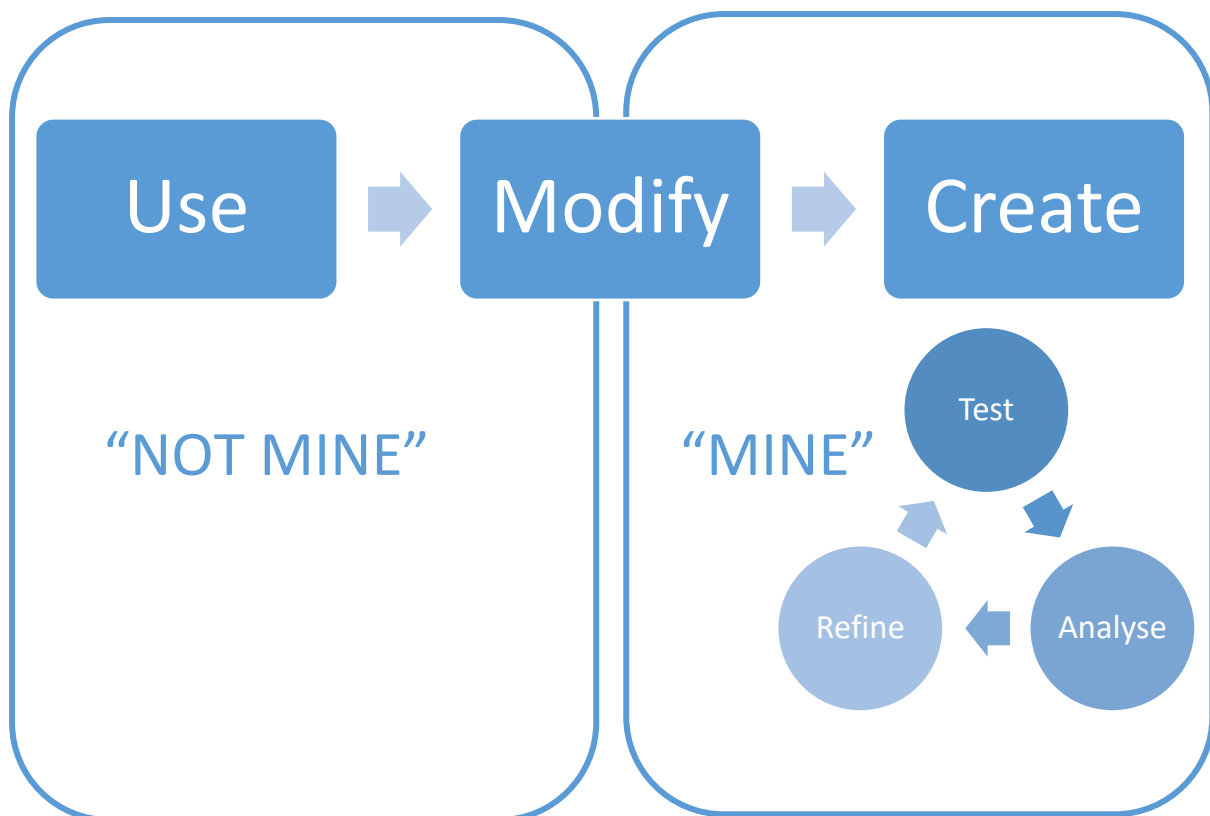


Figure 2.5 - The "Use-Modify-Create" Model

2.5.3.1. PRIMM

PRIMM is a new approach to teaching programming, based on the UMC framework and the research of Raymond Lister. It extends the original model by splitting the first stage into three more precise steps:

- **Predict** what a given programme will do
- **Run** the programme to verify the prediction
- **Investigate** what each line of the programme does
- **Modify** the programme to alter its original behaviour
- **Make** a new programme based on the original one

As can be noted, the new framework puts more emphasis on understanding and investigation of the code, before students can attempt to make their own modifications. This change was primarily inspired by the work of an Australian researcher, Raymond Lister. His experiments showed that

students should be able to trace code with over 50% accuracy before moving to the next step – making their own changes (Lister, 2011).

As PRIMM is a relatively new approach, it is still being tried and evaluated. In their paper, the authors presented a plan of a three-phase study that they have started. In Phase 1, the framework was introduced to non-specialist teachers during a course on teaching computer science. Feedback was gathered from 15 participants and showed that the majority found it helpful and stated that they would like to use it. In Phase 2 of the study, which was under implementation, 7 experienced teachers were asked to test PRIMM in their classrooms. In the final phase, the approach was planned to be tested during trials in more schools over an academic year (Sentance and Waite, 2017).

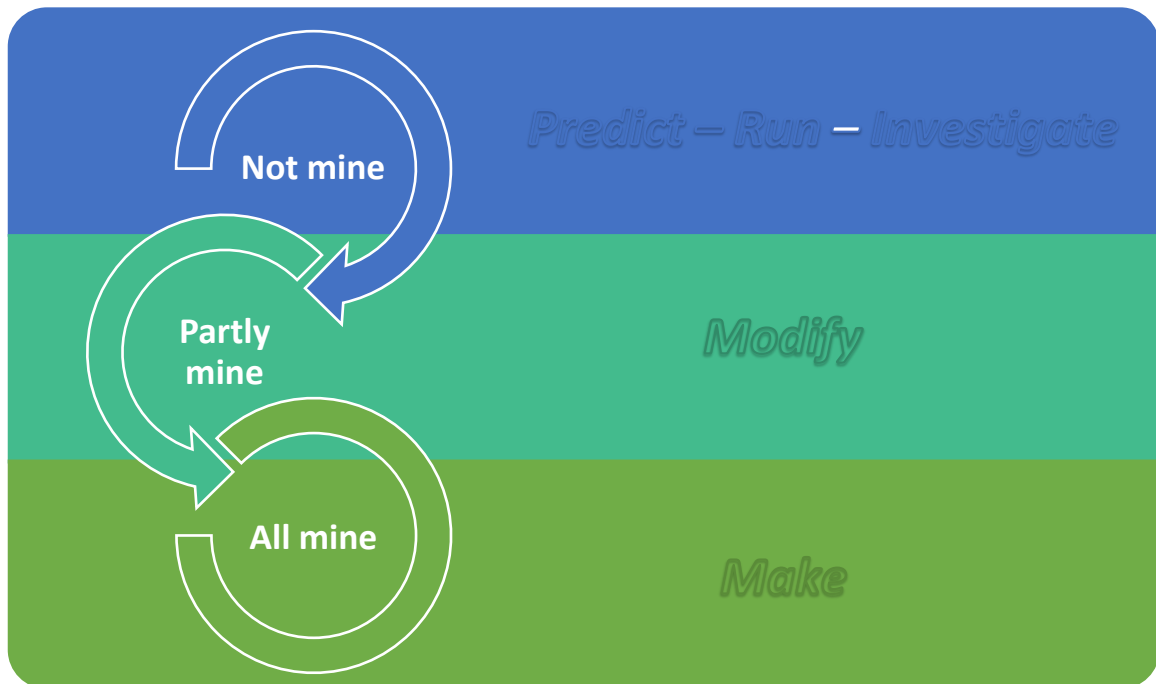


Figure 2.6 - PRIMM approach model

In 2019, Sentence et al. (2019) published a new article presenting the details of their study that focused on evaluation of PRIMM. It was highlighted that the framework was influenced by Vygotsky’s sociocultural learning theory and that the aim of the study was to analyse it from this perspective.

Vygotsky’s work emphasises the importance of social interaction during the learning process – discussing with peers and seeking guidance from more experienced individuals (such as teachers) (Vygotsky, 1978). An important part of the PRIMM framework that draws from sociocultural theory, is the first stage where pupils try to understand a programme (predict-run-investigate). It gives them the opportunity to talk with each other about the given programme and learn together.

The framework was evaluated in 13 schools, among 11 to 14-year-old pupils. After 8-12 weeks of learning with PRIMM, they were asked to fill out a test. Then the results were compared against a control group that was taught using a mix of other methods. The experiment proved that the framework helped to improve pupils’ performance and make better progress than the control group, regardless from their abilities. PRIMM contributes to developing more effective pedagogies and encourages to use language and dialogue in computing education (Sentence et al., 2019).

2.5.4. Rubber Duck Debugging

As the new computing curriculum was introduced to British schools, Computing at School and Naace prepared a guide for secondary teachers, containing a lot of tips and ideas. One of the recommendations was to encourage pupils to solve problems by doing so-called *rubber duck debugging* (Kemp, 2014).

Rubber duck debugging or simply *rubber ducking* is a technique of solving problems by talking to a rubber duck (or any other object) and was first introduced in The Pragmatic Programmer book. The idea behind this is to break down a problem by saying it loud to someone else (Hunt and Thomas, 1999). As explained in a blog post “Rubber Duck Debugging: The Psychology of How it Works”, this technique allows a person to slow down with their thoughts and become more exacting. As rubber ducks “don’t know how to programme” it forces people to assume the role of a teacher and precisely explain the problem step-by-step, which ultimately helps them to understand it (Hayes, 2019).

According to Hankin (2017) a British secondary school teacher explained how he implements this technique in his classroom using a screen recording tool called Screencastify. He asked his students to record a video in which they present their programming work and explain it out loud to a real or imaginary rubber duck. This allowed them not only to easily spot mistakes, but also to better understand their code and consolidate their skills.

Rubber duck debugging can also be done online. A popular implementation of this idea is a bot called Cyberduck, based on ELIZA (early natural language processing computer programme). The website contains a terminal-themed chat box where a user can type down the details of their problem and receive a “reply” from Cyberduck. The reply appears once the Enter Key is pressed and it refers to the user’s previous message, imitating a real conversation (Rubber Duck Debugging, 2017).

Another, more recent and under-development project is Duckie by John Eckert. This web-based implementation allows users to actually talk about their problem. The application uses speech recognition APIs (application program interfaces) to transcribe human speech and automatically pick up keywords, which are displayed in circles of different sizes, depending on relevancy (Eckert, 2018).

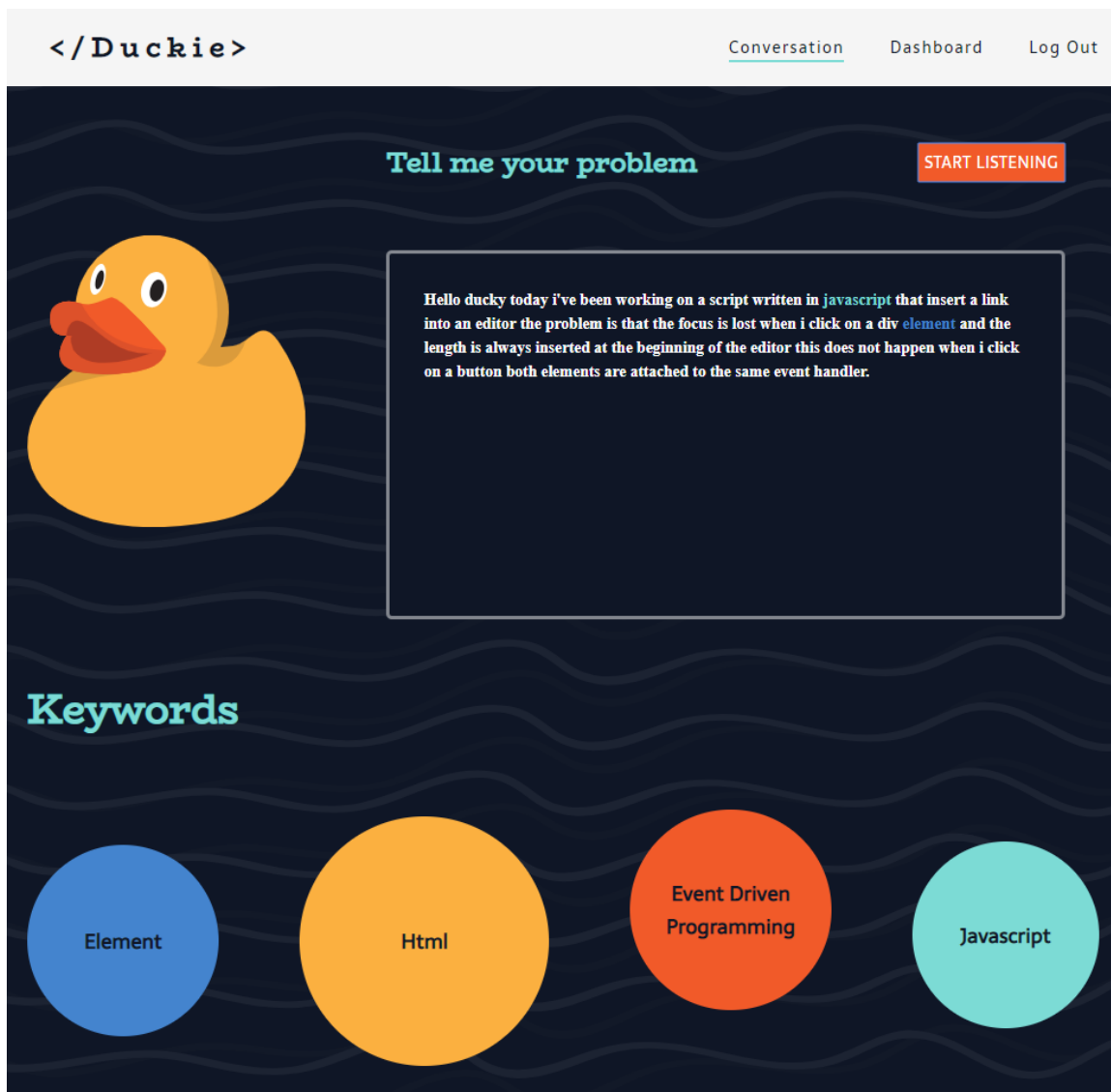


Figure 2.7 - An Example "Conversation" Recorded Using Duckie

2.5.5. Educational Software and Courses

As the importance of computing education continues to grow, new solutions are being developed in order to support teachers in their work. Their aim is to make computer programming easier to understand and more enjoyable so that the students become interested in the subject.

However, this type of software is not the invention of the 21st Century. One of the first tools created for educational purposes was Logo programming language, designed in 1967. The language provides simple commands to draw geometric shapes. The most common implementation of the language is so-called *Turtle graphics* where the students can write commands to move a turtle on a white space. The turtle 'holds' a pen, so as it moves it draws lines (Logo Foundation, 2015).

The table below lists some of the most popular educational software/platforms for computing education:

Author(s)/Date	Summary of conclusions from work
Code, 2014	<p>CODE (Code.org) is an integrated online teaching environment designed to help introduce students (from a variety of ages, 5-16) to the fundamental concepts of programming. It consists of a series of static web-pages with key content including:</p> <ol style="list-style-type: none"> Simple code concepts tutorial videos. Curriculum suggestions for the US school system. Some elementary tutor educational resources. Suggestions on how to motivate young learners to become interested in computing/code. A limited number of assessment tools embedded within the framework.
University of Kent, 2014	<p>Greenfoot is an interactive Java programming environment, very popular in teaching high school students and undergraduates. It has got a simple GUI (Graphical User Interface) that makes learning programming easier and more attractive than just writing code in a text editor. The software is used to create 2D graphical applications, such as games and simulations. A typical application consists of:</p> <ol style="list-style-type: none"> World – a space where objects exist Actors – objects that populate the World and can perform different actions
Bobrow et al., 1967	<p>Logo is a programming language that was developed for educational purposes. Students can draw shapes using simple commands or combine these commands into more complex methods – e.g. instead of drawing a square line by line each time, it is possible to create a Square method. The Logo programming software is mostly famous for the turtle icon which moves on a white space, drawing a desired shape.</p>
Carnegie Mellon University, 2014	<p>Alice is a 3D programming environment and an educational tool that allows students to create animations for telling stories, playing interactive games or creating videos. It was designed to teach object-oriented programming and basic programming concepts. Similarly to Greenfoot, it allows to create applications with:</p> <ol style="list-style-type: none"> Virtual World 3D objects (actors) that populate the Virtual World and can be animated
Microsoft Research, 2014	<p>Kodu is a free software that lets children and adults create PC or Xbox games using a simple visual programming language. According to the creators, it develops many skills, such as: creativity, storytelling, problem-solving and programming. It does not require any previous experience in programming, therefore it can be used for educational purposes.</p>
MIT Media Lab, 2014	<p>Scratch is a programming language and a free multimedia authoring tool, created for educational and entertainment purposes. It uses event-driven programming and multiple active objects, called “sprites”. The software has got a user-friendly GUI and allows to create simulations, visualisations of experiments, animated presentations and stories.</p>
Lego and MIT Media Lab, 2013	<p>Lego Mindstorms are programmable robots built using Lego bricks. The robots typically consist of:</p> <ul style="list-style-type: none"> RCX – a brick with a mini-computer Sensors, such as bumpers (physical contact detection) or light sensors (differentiation between light and dark) Other parts, such as motors and wheels <p>The robots are fully customisable to perform different tasks. They can be programmed in various languages, such as: Ada, Java, Visual Basic or C-like languages.</p>
Google, 2012	<p>Blockly is an open-source JavaScript library developed by Google. It uses the concept of blocks, known from Scratch, for building code. Blockly is a purely client-side solution, which means that it runs in a browser and the user does not need to install it. The programme built using the blocks can be easily converted to real code (e.g. JavaScript or PHP) at any time.</p> <p>It is used by many educational applications and platforms, such as Micro:bit or Code.org.</p>
Zach Sims and Ryan Bubinski, 2011	<p>CodeCademy is an online platform which provides programming tutorials in 12 different programming languages (e.g. Java, PHP, SQL) as well as markup languages (HTML). The student is presented with a simple text editor and step-by-step guide, each step containing certain tasks to complete.</p>

	The platform is generally free to use, however, a paid 'pro' version is also available.
BBC, 2015	Micro:bit , according to the manufacturer "is an award-winning programmable device that allows students to get hands-on with coding and digital making." (Micro:bit Educational Foundation, 2023) Micro:bit offers a range of features, such as light and temperature sensors, sound input/output, LEDs and buttons, and many others. It can be programmed using Microsoft's MakeCode or Micro:bit Python Editor.
Brian Harvey and Jens Mönig, 2011	Snap! is a graphical programming language, an extended reimplement of Scratch. It is browser-based and available for free. Just like Scratch, it uses blocks to build a programme, but is far more advanced and powerful. The new features include: Build Your Own Blocks, first class lists, first class procedures and continuations. The users can also easily convert their block programme into code written in one of the most popular programming languages such as Python and JavaScript.
Phillip Guo, 2010	Python Tutor – a web-based tool allowing the learner to visualise their code written in one of the popular programming languages: Python, Java, JavaScript, TypeScript, Ruby, C and C++. The user is able better understand what is happening "behind the scenes", as the computer executes the code that they have written. Another feature of Python Tutor is Codechella – a real-time collaborative mode. It allows multiple users to view and work on the same visualisation. This way users can exchange ideas, support and learn from each other. Other features of the environment include: <ul style="list-style-type: none"> • Codeoption – a real-time activity monitoring dashboard that allows a single user to watch the progress of many learners. • Omnicode – live programming environment that visualises the history of all programme values to give users a "bird's-eye view of execution". • Happy Face – a "pain scale" allowing users to report their frustration level.
Phillip Guo and Jeremy Warner, 2010	CodePilot – a web-based, collaborative programming environment designed for novices. Since the software does not need to be installed, the users can quickly and easily begin learning programming in pairs and test-driven development. Project repositories can be imported from GitHub and then two users can work on a project simultaneously, helping each other and learning together.
Raspberry Pi Foundation, 2018	Scratch to Python: Moving from Block- to Text-based Programming – an online course that aims to help students who have already mastered block-based programming to move on to a higher level – text-based programming in Python. It was developed in response to the feedback from educators, who reported this problem. The course takes 4 weeks to complete and covers transferring programming skills from Scratch to Python, programming and debugging in Python, understanding what can be created using text-based programming. It is taught by Raspberry Pi educators and is available for free (and upgraded, paid version with extra features is also available).
Guido van Rossum, 1998	IDLE – Python's Integrated Development and Learning Environment. It is a cross-platform software for creating programmes in Python. The IDE consists of a text editor and Python shell window. The main features are: colour coding, smart indent, tips and auto-completion. Users can also search for and replace text. Debugging is also available.
Amjad Masad and Haya Odeh, 2016	Repl.it – is a cloud development environment and collaboration tool. It offers a text editor with colour coding and suggestions, debugger as well as hosting and deployment service. Furthermore, it is a powerful collaboration tool, featuring real-time collaborative programming, live chat, embedded code and examples. Finally, it allows to create a classroom environment, where students and teachers can collaborate and build on exercises. Repl.it supports a wide range of programming languages, including: C-family, Java, PHP, Python, Ruby, JavaScript and SQLite.
Vitaly Pavlenko, 2012	Snakify – is an online programming course for beginners. It offers lessons covering basic principles of programming, such as different types of variables, conditions, loops and arrays. Each lesson contains exercises with an in-browser code runner, helping students with practice. The course covers Python, HTML5 and CSS, JavaScript and its library – jQuery.

<p>Joshua Lowe, 2016</p>	<p>EduBlocks – is a project started by a 12-year-old student who wanted to help his friends with the transition from blocks to text and today his online solution is used in more than 120 countries (AbilityNet Tech4Good Awards, 2019). Its main features are: displaying Python code on blocks, switching to Python editor and compatibility with Micro:bit and Raspberry Pi.</p>
-------------------------------------	---

Table 2.3 - Summary of Existing Software Frameworks/Tools for Teaching Computing

With such a wide choice of educational support software and tools it could seem that the needs of students, teachers and educators are fully satisfied. However, Roffey (2019) disagrees with that. In his article published in the computing education magazine “Hello World” he argues that there is a need for a new development environment for teenage pupils (Roffey, 2019).

2.5.6. Support for Teachers

In order to provide more support for computing teachers, a new government-funded initiative was established: National Centre for Computing Education (NCCE) (Henson, 2019). This consortium, made up of STEM Learning, BCS (British Computing Society) and RPF (Raspberry Pi Foundation), launched a web portal dedicated to computing teachers, called “Teach Computing”.

The website contains a news feed with the latest information about computing education and a bank of resources for teachers. It also allows them to book face-to-face and online CPD (Continuing Professional Development) courses run by NCCE. On top of that the consortium offers bursaries to state-funded schools in England that can cover the cost of courses (National Centre for Computing Education, 2019).

This initiative is seen as a positive step towards better computing education by helping to upgrade the skills of teachers, especially those who used to teach ICT and do not have any computer science background.

2.6. Successful Computing Teaching Strategies – As Per Teacher’s Perspective

The new curriculum for computing teaching in England has created a challenge for many teachers who are developing a number of teaching pedagogies to make their jobs easier. Some teachers have proposed that they are able to apply their existing teaching pedagogies to computing teaching as well. A recent study has determined that keeping one’s focus on a variety of strategies that have the ability to make the teachers feel more confident with computing teaching.

2.6.1. Common Teaching Pedagogies in Literature

The literature on computing teaching has identified several methods that make the subject of computer science more fun, engaging and accessible to the teachers and the students. Several researches conducted by Dewey (1938), Burner (1996) and Piaget (1950) suggest that when learning, the student has to be active and have the capacity to construct knowledge and context for the content they are learning. They must also be able to connect what they are learning with the knowledge they have acquired.

This is a part of the constructive learning theories that are applicable to computing learning that requires active attention to detail, and both constructive and subjective knowledge while placing students at the centre of the entire learning process. It is a well-known fact that students who study actively have higher critical thinking and problem-solving skills (Ben-Ari, 1998).

Another aspect that literature has made evident is that of experiential learning that also rises from constructivism and focuses on activities that engage learners directly. It is very similar to the idea of

Papert's (1993) constructionism, and involves working with real-world tangible objects. It also builds on the concept of constructivism. It has also been observed that kinaesthetic activities that promote active participation enhance classroom teaching of computing education (Bell et al., 2009; Nishida et al., 2009). Such activities explain takeaways in a practical and concrete manner. As per research carried out by Curzon et al. (2009), it was identified that many engaging kinaesthetic activities were generated by CS4FN (Computer Science for Fun). Grover and Pea (2013) have determined that it is essential for students to gain a real-world context for learning so they can have a firm grasp on their concepts.

Moreover, students must have excellent computational thinking skills in order to excel at computing education and learning. Wing (2006) popularised computational thinking in her research but these skills were being implemented for a long while before this research. As per Curzon et al. (2014), with the release of the new curriculum, guidelines have been developed for teachers to help them understand how to teach computational thinking explicitly in classrooms. The aspect that is most difficult to teach in computer science is programming. To teach programming, several activities can be arranged to facilitate student learning. For producing viable examples of the activities, you can make use of, Van Gorp and Grissom introduced the following for the benefit of computing education teachers in England:

- Walkthroughs of Code
- Writing of algorithms in groups
- Practice of adding comments to existing code in pairs
- Developing code by using algorithm in pairs
- Identifying bugs in code

As per research conducted by Lopez et al. (2008), in order to support programming teaching and learning, it is essential to read and trace code. It is an important step for the problem-solving stage of writing code (Lister et al., 2004). A major pre-requisite of writing a programme is that the students first need to trace code with a minimum of 50% accuracy.

2.6.2. Teaching Pedagogies Identified by Teachers

Teachers in England were selected to participate in a study that examined some of the top pedagogies that teachers use in their daily classrooms for teaching computing at schools. Before teaching methodologies were discussed, the teachers identified some challenges that students frequently face in classrooms while learning computer science. These include the students not understanding or having difficulty understanding complex computing concepts. At times, there are gaps in the teachers' knowledge of the subject. There are challenges of differentiation, a lack of resources and other technical problems that also make it difficult for students to grasp computing education with ease. Then the teachers described some of the most common techniques they use frequently while teaching computing, mentioned as below:

2.6.2.1. Contextualisation of Learning

One of the ways of adding context to computing teaching is by relating the computing content to other types of content being taught in the curriculum. They also use examples from the real life or home setting to clarify all the student concepts.

2.6.2.2. Collaborative Working

Many teachers have introduced collaborative activities to accommodate teaching of computing education. These activities include but are not limited to peer-mentor checks, paired collaboration and programming and teamwork. As per Kafai and Burke (2014), such strategies are based on the computational participation concept and are developed in the classroom. Many teachers believe that

collaborative working has the power to motivate students individually and the entire class as well as a unit.

2.6.2.3. Computational Thinking

Teachers also use computational thinking as a primary skill to add to the students' potential to learn computing education successfully. This type of thinking includes decomposition, logical thinking, abstraction and problem-solving (Brennan & Resnick, 2012; Curzon et al., 2014).

2.6.2.4. Tracing and Scaffolding the Code

Through this technique, the teachers help the students in understanding programme code. By tracing the code, students are able to identify the code they require to write their own programme. Through scaffolding, students learn to debug and extend programmes. By debugging programmes, students are able to learn how to write programmes as well (Diethelm et al., 2012).

As it can be seen from the above discussion, teachers feel confident when they utilise multiple strategies to teach computing. The same is corroborated by research conducted by Sentance and Csizmadia (2015). It seems a better approach than sticking to just one strategy.

2.5.1. Visual and Blocks Programming

Visual programming is writing programmes by arranging and combining graphical elements, often called "blocks". The main advantage of using visual programming languages (VPL) is their simplicity. First of all, VPLs remove the problem with correct syntax, as syntax errors cannot be made – there are no semicolons, brackets or quotes to worry about. Secondly, the shapes of the blocks give users a hint on how to combine them, e.g. a Boolean block is angular and fits in the angular blank space of an "if" statement. Furthermore, development environments for VPLs are usually user-friendly. They not only provide a catalogue of blocks available to the users, but also allow them to easily execute their programmes and immediately verify the output, which increases their satisfaction. These features make VPLs a good choice for beginners with no programming background, as they provide them with all they need to build a programme and allow them to focus on the logic instead of syntax. From the visual side they are also more appealing and less intimidating than textual programming languages. (Dehouck, 2015; Kölling et al., 2017).

Visual programming also has limitations, which is why it is not used by professional programmers. Programmes created by using blocks take a lot of space, making them unsuitable for bigger and more complex projects. The visual side can also become problematic – while the colourful blocks make simple programmes look more readable and appealing, it is the exact opposite with large programmes. Another disadvantage is poor navigation and organisation of code, which make it more difficult to edit existing programmes in VPLs when compared with text-based languages (Kölling et al., 2017).

With the growing number of pupils taking Computer Science in schools, learning to programme with the help of block-based programming environments for beginners is becoming increasingly popular. Currently, VPL tools with the highest popularity are Scratch, Blockly and Snap! However, it is not clear whether block-based learning is better than text-based. In 2017 two American professors, Weintrop and Wilensky, conducted a study in high schools to understand the differences between using block- and text-based programming in education. For five weeks they observed two groups of students, where one group was learning to programme using block-based language and the other used a traditional text-based language. The results showed that both groups made progress during the experiment and there was no significant difference in confidence or enjoyment. However, students who were using blocks seemed to gain more knowledge and showed more interest in doing computing

courses in the future. On the other hand, text-based students described their experience as more effective and closer to what professionals do (Weintrop and Wilensky, 2017).

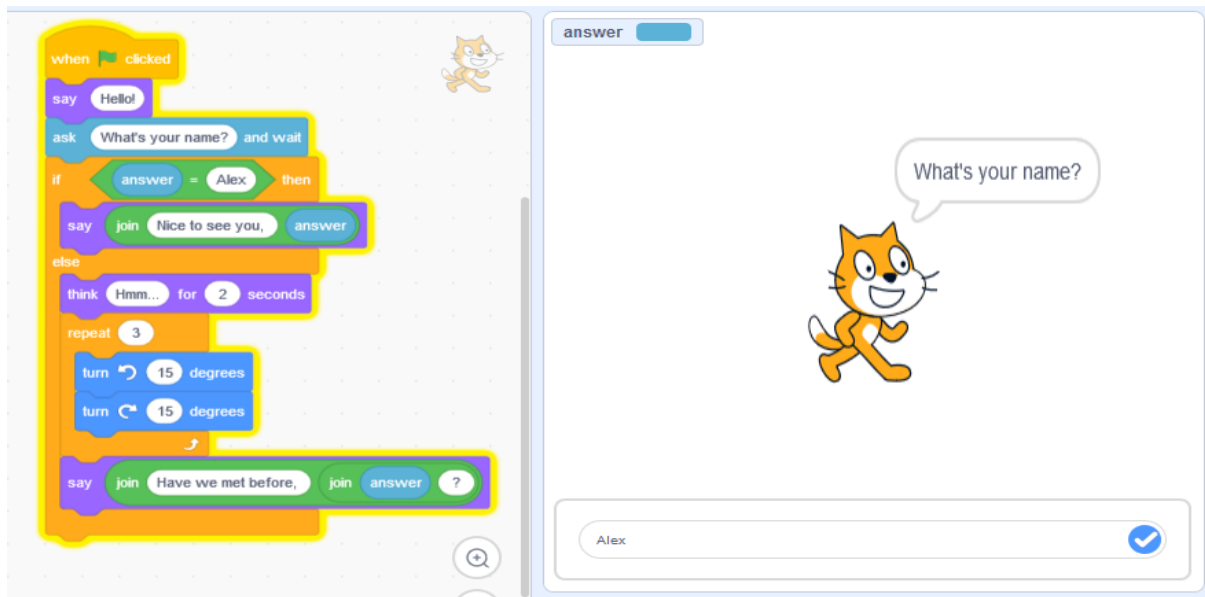


Figure 2.1 - An example programme created using Scratch

2.7. Insight into the Practice of Industry Professionals

To gain a complete understanding of effective learning approaches in computing, it is crucial to explore how professionals acquire and utilize these skills within the industry. This investigation aims to address several key questions:

Contrasting Learning Environments: How does the process of learning to program differ between industry professionals and pupils in secondary schools?

Leveraging Industry Tools: What practical tools and technologies employed by professionals could be potentially adapted for use in schools?

Understanding Motivational Factors: What factors initially motivated individuals working in the computing industry to pursue programming? Can these insights be used to build similar enthusiasm among young learners?

Researching these questions can help to identify valuable strategies and resources from the industry that can inform and potentially improve computing education within schools.

2.7.1. Stack Overflow Surveys

Stack Overflow is a popular question and answer website for programmers, with about 50 million people visiting it every month. Each year, the portal organises a survey among its users, asking them questions about their education, jobs, skills, programming preferences and many others. With around 90,000 to 100,000 responses from all over the world in each survey, it is a rich source of data that can give a bit of insight into the learning practices of people from the industry.

The Stack Overflow (2019) survey shows that most of the users from the UK wrote their first line of code when they were 14 years old. The majority of professionals – about three-fourths – have a bachelor's or master's degree, with over 60% of them majored in computer science, computer or software engineering. However, they have not stopped their education after that.

In terms of informal education, nearly 90% of professionals stated that they learnt something new (language, framework or tool) without taking a formal course. Still, over half of them also took an online programming/software development course. In the 2018 survey, professional developers had a chance to say how they learn on their own. The most popular ways are ex aequo: reading official documentation/standards and using the Stack Overflow website. The second most popular methods of learning were books and other online developer communities, followed by technology's online help system.

This shows that while most professional programmers obtained formal education, a great portion of their knowledge and skills come from other sources. It is worth noting how Stack Overflow and other developer communities play an important part in their learning (Stack Overflow, 2018).

It could be argued that since these surveys were conducted by Stack Overflow, its high popularity as a learning tool may not be objective. However, in Alexa's website ranking Stack Overflow has reached position 36 (as of October 2019), making it the most popular website in the category of Programming (Alexa, 2019).

2.7.2. Professional Software Development Environments

Writing code has become much easier, not only thanks to online resources and support communities, but also thanks to IDEs – Integrated Development Environments. The Stack Overflow surveys show that most programmers rely on them in their work, regardless of the programming language they use or the type of software they develop (Stack Overflow, 2018; Stack Overflow, 2019).

IDEs are advanced applications used to write programmes. They offer many useful features that can simplify and speed up everyday tasks, increasing productivity and reducing number of errors (Codecademy, 2019). These applications have many features that provide support in writing, compiling and debugging code. A research specialist, Aaron Walker names several benefits of using IDEs:

- Version control (tracking changes)
- Debugging tools
- Code completion (intelligent suggestions and automatic code insertion)
- Refactoring (safe renaming)
- Maintaining development cycle
- Increased efficiency and satisfaction

Walker (2018) also lists the most common features that are present in most IDEs:

- Text editor (with syntax highlighting)
- Debugger (a tool that helps to identify and fix issues)
- Compiler (a component that translates programming language into machine code)
- Code completion (prediction and automatic insertion of code)
- Programming language support (one or more languages)
- Integrations and plugins

Based on two rankings – “Top IDE index” (Carbonnelle, 2019) and “The Top 20 Integrated Development Environment (IDE) Software” (G2 Crowd, 2019), as well as the results of Stack Overflow's Developer Survey (Stack Overflow, 2018), the author chose the following three highest-rated and most popular IDEs: Visual Studio from Microsoft, Eclipse from Eclipse Foundation and IntelliJ from JetBrains.

The following table shows a comparison of the most common features of the three IDEs:

Feature	Visual Studio	Eclipse	IntelliJ
Text Editor	Yes – with syntax and error highlighting	Yes – with syntax and error highlighting	Yes – with syntax and error highlighting
Debugger	Yes	Yes	Yes
Compiler	Yes	Yes	Yes
Code Completion	Yes (IntelliSense)	Yes (Code Recommenders Eclipse)	Yes (Smart completion, Chain completion and others)
Database Tools	Yes (MySQL, SQL Server, Azure)	Yes (MySQL)	Yes (MySQL, SQL Server, Azure, Oracle and others)
Version Control	Yes (supporting Git and TFS, more available via extensions)	Yes (supporting Git and SVN, more available via plugins)	Yes (supporting i.e. Git, SVN and TFS)
Programming Language Support	<p>Multiple languages. Built-in include:</p> <ul style="list-style-type: none"> • C • C++ • C# • Visual Basic .NET • JavaScript <p>More available via plugins, including:</p> <ul style="list-style-type: none"> • Python • Ruby 	<p>Multiple languages. Primary: Java.</p> <p>More available via plugins, including:</p> <ul style="list-style-type: none"> • C/C++ • C# • JavaScript • Perl • Python • PHP • Ruby 	<p>Multiple languages. Primary: Java.</p> <p>More available via plugins, including:</p> <ul style="list-style-type: none"> • Python • Perl • C/C++ • C# <p>Ultimate Edition:</p> <ul style="list-style-type: none"> • JavaScript • PHP • Ruby • SQL <p>And others.</p>
Integrations and Plugins	Yes	Yes	Yes

Table 2.4 - Comparison of Professional IDEs

2.7.3. Successful Computer Programmers' Stories

Nobody is a born genius. All people learn and work hard during their lives in order to acquire new knowledge and a variety of different skills. At some point, everyone chooses their own path in which they specialise and which will allow them to utilise their best skills. Some people are especially talented – their outstanding achievements not only bring them wealth and fame in their field, but also grant them a global recognition. An interesting fact is that according to the Forbes magazine, four out of ten of the richest people in the world are computer programmers (Forbes, 2019).

For many years, the top position of the Forbes rank has been occupied by Bill Gates – the co-founder of Microsoft and undoubtedly one of the most famous programmers in history (Forbes, 2019). However, in the era of computing, more and more young talents gain fame and fortune thanks to their exceptional programming (and business) skills.

As a part of this research, it is worth learning their stories and understand how they manage to learn computing as it may help to provide a better education for today's children. The list below shows the beginnings of several successful programmers, who stand behind the technologies used by millions of people every day:

- **Bill Gates and Paul Allen** – the founders of Microsoft share the same story, as they first met and became friends in secondary school. That was also the place where they were introduced to computers in the form of a teletype terminal. They became interested in learning how the machines work and taught themselves to programme BASIC (Biography.com, 2017; Nolen, 2017). In a short promotional video created by Code.org, Bill Gates said that one of his first programmes was a simple ‘Tic-Tac-Toe’ game (Code.org, 2013).
- **Mark Zuckerberg** – the creator of Facebook was first introduced to a computer by his father who was a technology enthusiast. Mark learned BASIC programming from his father and quickly became fascinated by the world of computing (Vargas, 2010). In the video clip by Code.org, he admits that he just wanted to ‘make something that was fun for myself and my sisters’. He learned gradually by reading books and looking for answers on the Internet (Code.org, 2013). After some time, he was able to create a simple messenger programme, called ZuckNet, for his father’s dental clinic. At some point, his parents decided to hire a software developer to teach Mark (Biography.com, 2017).
- **Drew Houston** – known as the inventor of Dropbox. When he was only 5 years old, his parents bought their first computer – a PCjr. His father, an electrical engineer, soon began to teach Drew BASIC programming (Massachusetts Institute of Technology, 2013). In the video clip by Code.org Drew says that his first programme asked the user simple questions (Code.org, 2013). Like most children, Drew enjoyed playing computer games, however, he was also fascinated by how they worked. His curiosity led him to learning C programming language by studying the source code of an online game. As a teenager, he signed up for testing a game and discovered a series of security bugs. He was then offered to work for the company, being only 14 years old (Massachusetts Institute of Technology, 2013).
- **Jack Dorsey** – the creator of Twitter. His family moved many times when he was young, due to his father’s job as an engineer of medical equipment. Because of that, Jack became interested in maps and communications (Biography.com, 2015). When he was 8 years old his parents bought him a Macintosh, which was his first introduction to a computer (Code.org, 2013). He managed to teach himself programming and later joined a computer club in his high school. Jack’s passion for communication combined with his coding skills led him to writing a programme for dispatching taxis when he was just 15 years old (Biography.com, 2015).

It can be spotted that all these stories share a few interesting facts:

- The great programmers started their journey at a young age, very often thanks to their parents.
- They were driven by curiosity, step by step they taught themselves how to write very simple programmes that were primarily designed to entertain or solve a real-life problem.

Computer science is all about solving problems and this should always be the starting point of computing education. Mark Zuckerberg once said: *“Learning how to programme didn’t start off as wanting to learn all of the computer science or trying to master this discipline or anything like that. It started off because I wanted to do this one simple thing”*. (Code.org, 2013).

Chapter 3: Methodology

3.1. Problems of modern computing education in British schools

In the recent years, a great progress has been made to improve computing education in the United Kingdom. The long-awaited curriculum changes bring hope that the new generation of children will be able to not only use software, but also create it. However, this improvement also introduced a number of new challenges, which were pointed out in The Royal Society's 2017 report.

The following list of problems were identified after a thorough review of literature that modern computing education stands facing nowadays:

- **Not Enough Qualified Staff**

With the arrival of the new curriculum, a number of people and organizations expressed their concerns that the majority of teachers were not prepared for the changes. Many existing ICT teachers, tasked with delivering the more demanding computer science curriculum, face significant challenges due to a lack of necessary knowledge and experience (Bateman, 2014; Wohl, 2017; BCS, The Tech Partnership, 2017). This finding aligns with the report by Royal Society from 2017, which similarly highlights the substantial gap between existing teacher skillsets and the demands of the new curriculum.

- **Different Strategies of Teaching Computing**

The study by Sentance and Csizmadia (2015) shows that teachers who are members of the Computing at School association are confident in using and combining multiple pedagogies and approaches in their classrooms. However, the report suggests that this may not be true for all the teachers. Many teachers who previously taught ICT can find it hard to identify which teaching strategies are the best methods to introduce their students to computing.

- **Lack of Support for Text-based Programming**

Literature review revealed that there are many free educational applications for students available. However, most of them rely on colourful graphics and look appealing to younger children. As the children grow up and make progress, the time comes for them to learn real-life text-based programming. Moving from a graphical programming software to text-based code editor can be intimidating, as the number of support tools at this stage seem to be limited.

- **Low Popularity among Pupils and Prominent Gender Imbalance**

Studies by BCS (The Tech Partnership, 2016, 2017) highlight a concerning trend: a decline in pupils' interest in computing following the introduction of the new curriculum. This decline extends to attracting girls to the subject, further increasing existing gender-gap. The Royal Society's 2017 report supports these findings, revealing a low uptake of computing in secondary schools, particularly among young girls. This gender imbalance in computing is the highest among all STEM subjects.

- **Connecting with the Industry**

The Next Gen report emphasizes that it is important to equip young people with industry-relevant computing skills to prepare them for university education and help to bridge the talent gap (Hope and Livingstone, 2011). One of the recommendations coming from The Royal Society Report (2017) was that the industry should continue to support teachers in their professional development. However, it is uncertain in what other ways the industry could support computing education in schools, as well as if and how to build a stronger connection between the two.

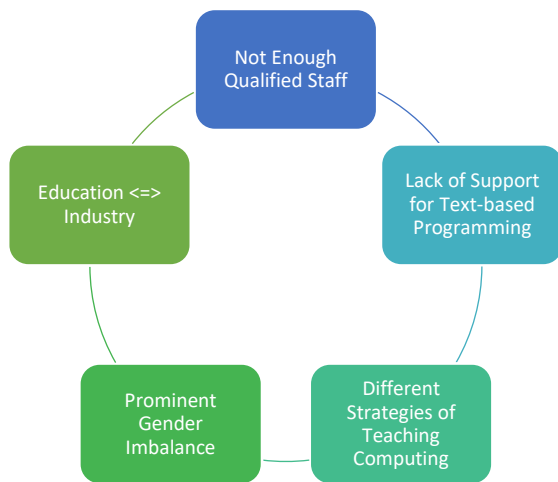


Figure 3.1 - Problems in Computing Education

A careful examination of the problems listed above allowed the author to define the aim and objectives for further research that can contribute to improving the quality of computing education in British secondary schools.

3.2. Project Aims and Objectives

The aim of this research is to investigate whether there is a value in the practice of the industry that could help in improving the quality of computing education in British schools.

The results will enable the author to develop a set of enhanced strategies for teaching the subject in secondary schools based on learning experience and everyday practice of professionals from the industry. This could enable computing teachers to do their jobs more effectively and encourage pupils to study this subject.

Moreover, the research outcomes can contribute to the identification of areas that require further investigation.

In order to achieve this aim, the following essential project objectives have been formulated.

- 1) Establishing the validity of the work by learning about computing education in Great Britain, observing the situation in British schools during the years after the new curriculum has been introduced; exploring current educational practice within schools, for the computing discipline, to identify any existing deficiencies and strengths present within current practice.
- 2) Undertaking a comprehensive exploration of the methodologies and contemporary technologies that aim to support students in computing education. Determining whether they meet the needs of pupils and educators.
- 3) Conducting qualitative research to gain in-depth insight into computing teachers' everyday problems and the methodologies they use; to determine their outlook on the practice of the industry and whether they see it as a potential source of values to school education.
- 4) Gathering and analysing quantitative data on how industry professionals became interested in computing and learnt in the past, as well as how they learn and solve problems on everyday basis; to identify practices that could be incorporated into this work.

- 5) Using a mixed-method research that involves agile quantitative and qualitative techniques for meeting the aim and objectives of this study.
- 6) Reviewing and presenting findings, as well as proposing a set of recommendations.

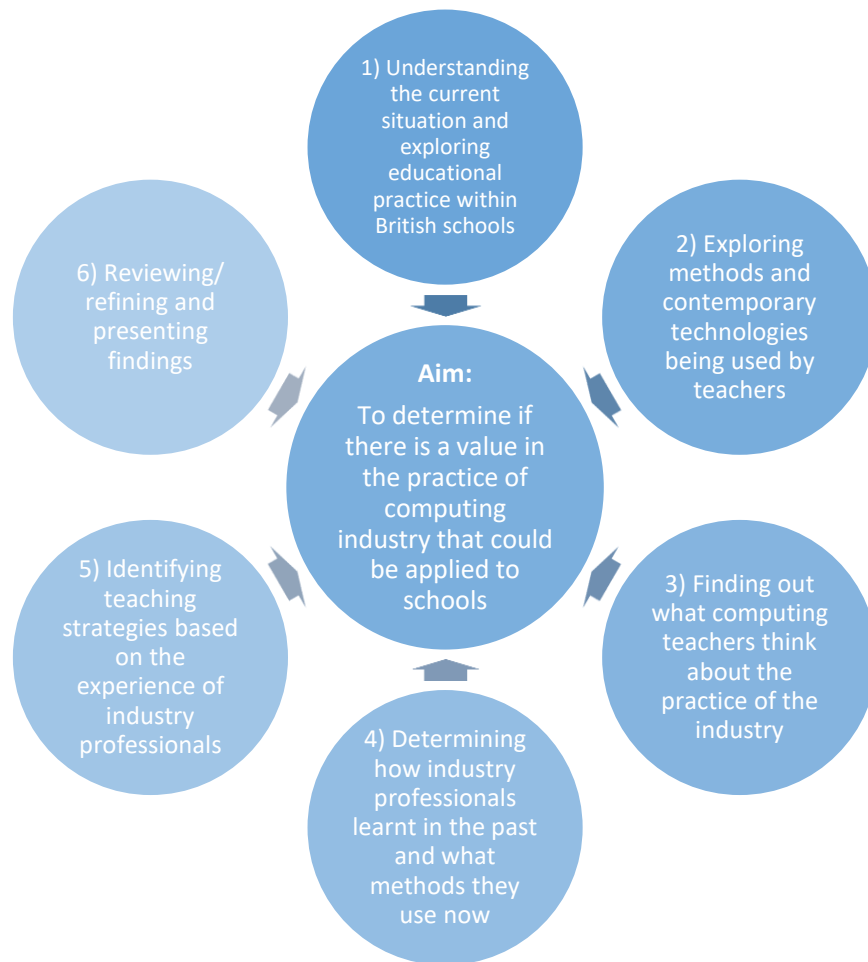


Figure 3.2 - Project's Aims and Objectives Summary

3.3. Research Design

This research study has utilised an exploratory sequential design, starting out with collecting the quantitative data and build up to the qualitative data collection and analysis and the proceeds to interpreting all data collected. In this design, the data results create or develop a new set of recommendations keeping in view the quantitative part of the research (Toyon, 2021). The qualitative part identifies key themes that help explain the teaching pedagogies in practice by computing educators in British schools as well as industry professionals who have learnt computing while working.

To obtain first-hand data from different groups of people involved in computing education, as well as people working in the industry, the primary data was collected from the following groups:

Group	Rationale
Computing lecturers	A course at a university is the next step of computing education for many young people who aspire to pursue a career in this field.

	Asking the lecturers a number of questions was helpful to paint a picture of secondary education and help to understand whether the current computing education in schools is sufficient to prepare young people for further education or work.
Computing professionals from the industry	These are the individuals who have made their educational journey – from school to work in the computing industry. They can be a source of practical advice and inspiration on how to improve computing education in order to make it more appealing to young people. Understanding what motivated them as well as what methods and tools they use to learn and work, was crucial in formulating the final recommendations for school education.
Computing teachers from secondary schools	Talking directly to secondary computing teachers was essential to understand the problems of modern computing education from their perspective. Interviews were also an opportunity to find out what educational tools are actually used in classrooms as well as how could the industry practice potentially help schools.

Table 3.1 – Groups of people for data collection with the rationale

Their views and opinions have been an important addition to what has been found through the literature review and have allowed the author to acquire a more complete understanding of the topic. Furthermore, the analysis of the data has allowed to identify innovative teaching/learning strategies that will meet the needs of both teachers and pupils.

3.4. Research Activities as per Ethics Policy

The table below summaries the research activities the author has conducted, with the underlying rationale for the each.

Research Area	Rationale/Justification
Exploring the current attitudes of Computing Lecturers (educators) in Higher Education Institutions, with respect to their perspective of the recent changes in computing education and the skill sets of students enrolling on computing courses within the UK.	<p>It has been found from the 2010 “Next Gen” report that school-leavers were unprepared for computing courses at universities and had poor skills.</p> <p>The 2014 and 2018 surveys were meant to confirm whether these findings were still true in the face of recent changes to the curriculum. The surveys were sent to a number of computing tutors from different British universities, asking them several key questions:</p> <ul style="list-style-type: none"> • What do the tutors think about the recent changes? • Has the quality of computing education improved or is expected to improve? • Do they believe that the experience of computing professionals could be used to enhance the quality of computing education? <p>Comparing the answers from both surveys allowed to see if and how their views changed after the introduction of the new curriculum.</p>
An investigation into the learning methods deployed by current Computing	The recent changes in UK computing education were triggered by the concerns raised by industry experts and the need to address the skills gap in this field (Hope and Livingstone, 2011; The Royal

professionals that enabled them to develop computer relevant competences. They were asked about their educational journey and experience, as well as current learning methods/practices and recommendations on how to improve the computing education.

Society, 2012). To effectively bridge this gap, it is crucial to better understand the perspectives of those who managed to successfully complete their educational journey and start a career in the computing industry.

An initial study of prominent computing professionals' biographies provided valuable insights into their educational experiences. However, to gain a more comprehensive understanding, a survey was conducted among professional programmers. This survey explored their early beginnings in computing, their school experiences, and how they continue to learn and solve problems in their daily work. The gathered information was an interesting input that, confronted with teachers' perspective, led to the development of recommendations on how to improve computing education.

The survey focused on three key aspects to gain a well-rounded understanding of industry professionals' viewpoints:

- **Early exposure and learning:** How did their interest in computing develop, and what were their school experiences like?
- **Continued learning:** How do they maintain and enhance their skills in the ever-evolving field of computing?
- **Educational recommendations:** What advice would they offer to educators and pupils to improve computing education?

Computing teachers in secondary schools are the main conduit from which students receive their computer science knowledge. An understanding of the attitudes of teachers towards the quality of education they currently provide, lays a logical foundation for the development of subsequent improvement strategies.

Several authors have already identified some potential limitations in the computing knowledge of the educators, the availability of relevant resources and delivery formats used within schools (BCS, The Tech Partnership, 2016; The Royal Society, 2017). However, a study by Sentance and Csizmadia (2015) showed that many members of the Computing at School association successfully use a variety of pedagogies and educational resources in their classrooms.

Interviews with secondary school teachers of different backgrounds and experiences allowed to obtain valuable first-hand information about how they deal with the new challenges and what pedagogies/resources they use. This also presented an opportunity to find out what is the attitude of teachers towards the computing industry.

To gain insights into teachers' experiences with the new computing curriculum, a set of interview questions was developed (Appendix 7). These questions focused on several key areas:

- **Teacher Background:** Educational and professional background of the interviewed teachers.
- **Curriculum Opinions:** Teachers' opinions about the strengths and weaknesses of the new computing curriculum.
- **Teaching Practices:** The methods and tools used by teachers in their classrooms to deliver the curriculum.
- **Programming Transition:** Exploring the challenges and opportunities associated with moving from block-based to text-based programming.

- **Essential Skills:** Investigating how teachers integrate using IDEs, finding online resources, and fostering collaboration among students.
- **Industry Relevance:** Examining the connection between industry practices and their potential value for enriching school curriculum.

The information gathered through these interviews was crucial in formulating a set of recommendations for improving the quality of computing education.

Table 3.2 - Research Activities that Require an Ethics Policy

3.5. Data Collection

3.5.1. Higher Education Lecturer Surveys

The methodology for the Higher Education (HE) questionnaires distribution is outlined in the list below - The Professional Practitioners' View of Computing Education in Secondary Schools in the UK Questionnaire Process.

- 1) The questionnaires (Appendix 2 and 4) were created using a free tool called eSurveysPro available online. The tool was chosen because it has got an intuitive interface as well as features which completely meet the needs of the researcher – not only does it allow to create an online survey, but also automatically generates graphical overview of the responses.
- 2) In an attempt to seek a set of representative results across the HE sector, the sampling frame to be used was extracted from the current Guardian League table of computing departments within the UK (e.g. 102 University Computing departments).
- 3) Using a non-random sampling method, universities were selected by employing a constant skip (e.g. selecting every 4th element from the sampling frame) starting at the first placed institution. This provided a representative sample of 25 intuitions.
- 4) Once the sample institutions had been selected, the research was carried out to identify 20 academic staff, from each Computing department (where departmental members were allowed). Where small academic staff members were noted, all available staff were selected for questionnaire distribution. In larger departments, preference was given to academic staff teaching on first-year degree or HND provisions at each institution, to help provide a more accurate picture of lecturer's perceptions of the students as they arrived in higher education.
- 5) The title, name and e-mail address of each identified member of the staff was noted and the data collected was used to issue appropriately addressed e-mails (Appendix 1 and 3), inviting the selected academics to take part in the completion of the survey.
- 6) The electronic deployment of the questionnaire was seen as appropriate, with a list of 475 academics currently selected from 25 institutions.
- 7) A period of two weeks was allocated to allow candidate participants to complete the questionnaire, with a reminder e-mail being sent to potential responders who haven't engaged with the survey after 1 week.
- 8) The researcher wanted to achieve a response rate between 10%-15%, to provide approximately 50-75 sets of responses, for subsequent data analysis.

Almost identical questionnaires were used in 2 surveys conducted in 2014 and 2018. The only notable changes in the second questionnaire were:

- Removing questions about introduction of the new curriculum in the future, as by 2018 the curriculum had been released
- Adding a neutral answer in the questions with scale rating, due to the feedback received in the first survey
- Changing the questions about the respondents to more relevant ones

The process of creating the questionnaire, selecting the sample and distributing the questionnaire remained the same. Therefore, the same ethic policy was applied to both surveys.

3.5.2. Teacher Interviews

Another group that delivered a significant input for the research, by highlighting any problems with the existing computing educational provision, are the teachers. Due to the importance of the views of professional educators in schools, it has been decided that a suitable approach for the collection of teacher viewpoint data would be the utilisation of suitably designed semi-structured interviews.

The process was started by contacting computer science teachers from several secondary schools in Northamptonshire (Appendix 8). Typically, they were teachers who had already worked with the institution, e.g. via STEM events. The preferred method of communication was email or telephone. The teachers were informed about the purpose of the research and asked if they would be willing to take part by sharing their opinions via an interview. Those who expressed their agreement were contacted again to agree on a date and time for the interview, a face-to-face meeting or a telephone call. All willing participant teachers were informed about the interview process and asked to complete the relevant consent form.

During the interview the teachers were asked the pre-formulated questions. It was estimated that each interview would take approximately 30 minutes, with the researcher aiming to interview about 10 teachers from schools throughout local counties. Each interview was recorded (with the consent of the subject) and the meeting outcomes were transcribed and analysed after the event.

3.5.3. Twitter #CASchat

The Computing at School (CAS) organisation organises a public discussion on Twitter under the hashtag #CASchat. The discussion concerns computing education and takes place every Tuesday at 8:00 PM. It is open to everyone with a Twitter account, though the participants are usually computing teachers, educators and enthusiasts. During the chat, members of CAS publish a set of questions (usually about 5) and let other participants answer them and share their opinions and ideas (Humphreys, 2016).

The #CASchat is therefore a rich source of information voluntarily provided by people involved with computing education.

The #CASchat that was chosen for analysis was held on 18th September 2018 and contained the following set of questions:

- How do you think the practice of industry professionals (particularly approaches to learning and problem-solving) differs from those used in the classroom?
- As educators, what lessons do you think we can learn from industry that could be applied to the classroom?
- What are the main problems/challenges faced by children when moving from blocks to text-based programming?
- When is the best time to move from blocks to text-based programming? What are your tips for making the transition easier?
- What features do you think a beginners' text-based programming language should have?

As can be noted, the above questions are highly relevant for this research which was the reason for choosing this particular chat.

3.5.4. Computing Professionals Survey

The proposed methodology for the computing professionals' questionnaire (Appendix 6) distribution is outlined in the list below - The Computing Professionals Educational Background Questionnaire Process.

1. Similarly to the first questionnaire, this one was also created using the eSurveysPro tool, as it has proven itself to be a good choice.
2. In an attempt to seek a set of representative results across the software sector, a part of the sampling frame was extracted from the members of LinkedIn social network.
3. As LinkedIn does not allow unconnected users to communicate, the author used a trial version of the recruiter's account. This type of user has got more permissions and the trial version allows contacting up to 30 unconnected users.
4. The sample group was found on LinkedIn using the portal's search feature. The search criteria were set as follows:
 - a. Connections: 2nd and 3rd+
 - b. Keywords – Title: software
 - c. Locations: United Kingdom
 - d. Industry: Computer Software
5. The author then browsed through the search results and chose users who stated on their profiles that they attended British schools and now worked in the computing industry.
6. The selected LinkedIn members received a short message explaining the purpose of the research and with a link to the survey (Appendix 5).
7. Since the number of LinkedIn profiles that can be contacted is relatively small, the author decided to ask her personal connections to participate in the survey as well.
8. In this case the participants were selected from the author's colleagues and former university classmates who attended British schools and now worked in the industry.
9. The participants were contacted using Facebook or LinkedIn. The author sent them messages asking for help in the research by filling out the questionnaire (a link was provided).

3.6. Research Analysis

3.6.1. Survey Analysis

As mentioned before, all surveys were created using a free online tool – eSurveysPro.com. This service also automatically generates a Summary Report for every survey, containing appropriate data charts for each question.

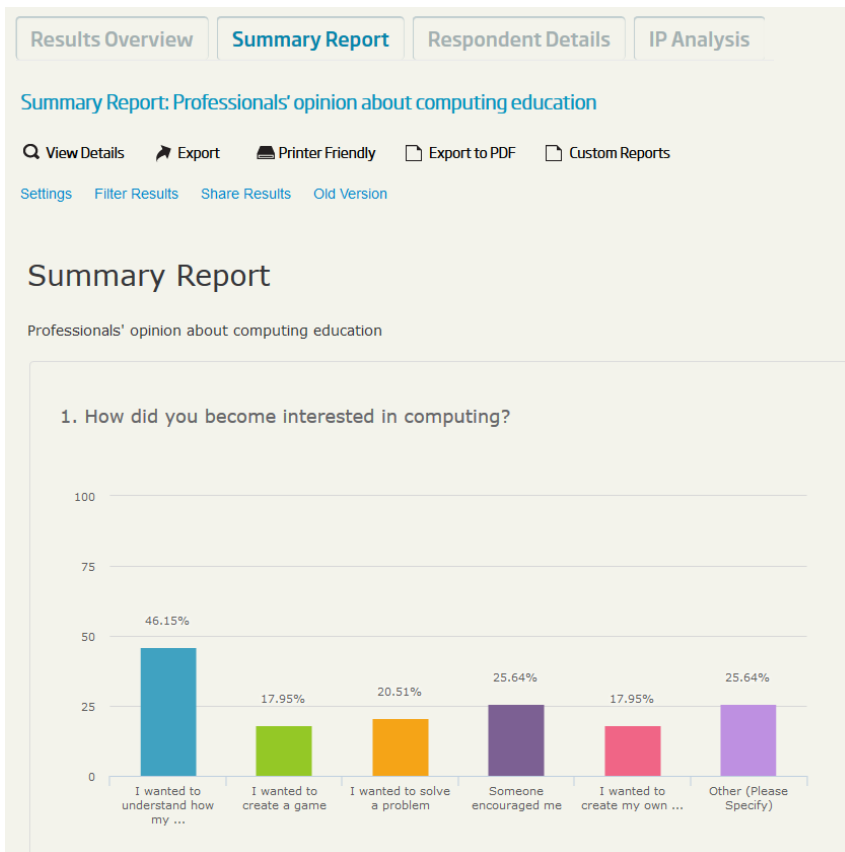


Figure 3.3 - A Screenshot of Summary Report on eSurveyPro.com

The data from each survey was then manually transferred to a spreadsheet for further processing, details of which are presented in the sections below.

The results were then represented on graphical charts: column format for scale/multiple choice questions and pie format for single choice questions. The charts were generated using the Excel functions.

Finally, the results from the first and second Tutor Survey were copied to a single Excel spreadsheet in order to compare the answers given in 2014 and 2018. The comparison was also shown in a graphical format using column charts.

3.6.1.1. Single/Multiple Choice Questions

Each query contained several closed questions with a set of possible answer options, sometimes including an “other” option to allow respondents to provide an alternative answer. In this type of questions, a percentage of respondents who chose each of the options was calculated as follows:

$$\frac{ns}{nr} \times 100\%$$

ns – number of times an option was selected
nr – total number of respondents who answered the question

The results were then presented on either column or pie charts with percentage scale, depending on the number of answers to display.

3.6.1.2. Questions with a Rating Scale

Questions with a rating scale are quite special, e.g. from *Strongly Agree* to *Strongly Disagree*. In this case, a weighted arithmetic mean was used to calculate an *average score* for every statement within the question.

First of all, every answer option was given a 'weight': numeric value from 1 to 4-5 (depending on the number of options), where 1 is the worst opinion/lowest level of agreement. For each option its weight was multiplied by the number of times the option was selected. All outputs were then summed and divided by total number of respondents who answered the question.

$$\frac{\sum_{i=1}^{no} (i \times ns_i)}{nr}$$

no – number of options

ns – number of times an option was selected

nr – total number of respondents who answered the question

The results were presented on column charts with a decimal scale.

3.6.1.3. Open Questions

In the case of open questions where respondents could write down their answer, the analysis was a more manual process.

To begin with, from all available answers only those that were relevant were chosen – for example all good luck wishes were omitted. Then the answers were copied to a Word document, where similar statements were highlighted with the same colour, e.g. “don’t give up” and “never give up”. This allowed to see if the respondents agreed on certain topics and find the most common ideas, which were later presented in a table along with the percentage of respondents who shared them.

A visual representation in the form of so-called *word clouds* were generated as well. The tool used for this purpose was WordClouds.com website, which creates a word cloud based on inserted text and selected criteria. The copy of all answers to each open question were copied to a Word document, where any spelling mistakes were corrected and all words were changed to lower case, in order to make sure that the programme will group them correctly. Then the whole text was transferred to the online tool and the following options were selected:

- words which appear only once were omitted
- some words, such as pronouns and prepositions were omitted
- fonts, shape and size were chosen.

3.6.2. #CASchat Analysis

The #CASchat discussion on Twitter was analysed using online tools developed by a software engineer Martin Hawksley: TAGS and TAGSExplorer.

The TAGS is a tool created using Google Spreadsheet that allows the user to automatically generate a spreadsheet in real time, containing the details of Tweets and their authors. The user can specify search terms (e.g. hashtags), time period and limit the number of tweets (Hawksley, 2012).

For the purpose of this research, the filter criteria were set as following:

- Search criteria: #caschat OR #CASchat
- Tweets limit: 3000

- Period: -1 days (the #CASchat was held on 18th September 2018 and the data was collected on the following day)

The other tool used was the TAGSExplorer that enables users to create an interactive visualisation of Twitter conversations that were previously saved to a Google Spreadsheet using TAGS. The visualisation illustrates the connections between conversations in the form of an animated graph. The application also offers statistical data in a user-friendly format: Top Tweeters, Top Hashtags and Top Conversationalists. Another very useful feature is Search Archive that allows the user to easily find relevant tweets using keywords (Hawksley, 2018).

The Search Archive feature was used to find all answers to each of the 5 questions that were posted during that #CASchat. As mentioned before, members of CAS who run the discussion require the question tweets to begin with the letter “Q” and the question number. Similar rule applies to the answers of every question – in this case the format is: letter “A” and the number of question that the answer refers to. This information was used to find the relevant tweets, for example answers to “Q1” could be found using the “A1” keyword.

As the tweets (including retweets) were collected in one place and filtered, they could be easily read and analysed to define a list of the most common answers and ideas. Similarly to the open questions in surveys, only the relevant tweets were chosen and similar ideas were highlighted with the same colour. What makes the difference in this analysis are retweets. People usually retweet messages that they find important or agree with. Therefore, it was decided to include them in the analysis. The results were presented in a table, including short title of the answer/idea, number of tweets and retweets that it appeared in, and a description including quotations.

A word cloud was generated from the tweets in the exact same way as in the open questions from the surveys.

Any personal data that could identify individual users has been removed and not used in the analysis. The locations of the participants were taken from their public profiles and listed in a spreadsheet without any other user data. Based on the UK locations a list of regions was created and saved to a CSV file, which was then uploaded to Google Maps in order to create a map with pinned locations. Apart from that, a list of all locations was used to produce a chart.

3.6.3. Interview analysis

The recordings of the interviews were listened to multiple times in order to familiarise with the data and make notes in the form of a table that would later allow to easily compare different recipients' answers. The first row of the table listed the names of the respondents and the first column – questions they were asked during the interview.

These notes contained the key points of each answer. Similar statements and ideas were highlighted using the same colour, allowing to see themes and patterns.

Personal details, such as gender, teaching experience and education were listed in spreadsheet and used to generate charts.

3.7. Ethical Considerations

3.7.1. Ethics Policy for Data Collection through Surveys & Trial Participants

Several sensitive areas have been identified, that may raise concerns from potential participants of the survey and the trial participants who were asked to fill a questionnaire:

Anonymity: A desire for anonymity is seen as a major consideration for survey participants, as potentially negative viewpoints about student knowledge levels and computing relevant skill sets could be seen as contentious or inflammatory, should the comments be attributed to a given academic institution.

Data Security: As the proposed collection vehicle for the survey is an online questionnaire distribution, potential participants maybe concerned about the security of the data they provide – how will the data be stored and who will have access to it?

Results Utilisation: Candidate participants implicitly want to know why the survey is being carried out and how the survey results output will be utilised.

As a response to these identified potential participant concerns, the following Ethical Policy was developed and employed for this aspect of the project:

- All candidate participants were informed about the research aims, survey purpose, estimated duration, potential consequences of the research, and the likely utilisation of the overall findings; when they were asked to participate in the study via email.
- As in this study there is no valid reason for collecting sensitive personal data that could be used to identify respondents, therefore they were not asked to provide any personal details, making the survey completely anonymous. All potential candidates were assured of anonymity in the initial invitation email.
- The data was collected and stored using the eSurveysPro online software. For this purpose, the author created an account on eSurveysPro portal that can be only accessed by logging in using email address and a strong password that meets all security standards.

3.7.2. Ethics Policy for Data Collection through Interviews

The potential ethical issues that may arise during this stage of the research process are outlined in this section; to this end, the researcher has formulated a list of likely participant concerns and questions that need to be considered:

1. Research topic – what is the research about and why is their input considered valuable?
2. Data protection – what kind of information will be collected and where will it be stored?
3. Date, time and location – when and where will the interview take place?
4. Answering questions – will they be able to come up with an answer straight away?
5. Interview authorisation – will they be able to read and correct a transcribed version of the interview?
6. During the interview – will they be able to stop at any moment if they feel uneasy or decide that for whatever reason they do not wish to continue the interview?
7. After the interview – will they be able to withdraw their consent for using the interview in the research?

In response to these considerations, the following Ethical Policy was developed and has been deployed throughout the interview process:

1. Prior to the interview, every teacher received a complete written information about the research topic and aims, as well as information about the interview process itself. Any questions that they had after reading it were answered instantly.
 2. The required personal data of the teachers were their first and last names and the names of the school in which they currently teach. These details were kept in a separate file and have allocated identification numbers. The numbers were used to link the personal details to the actual interviews. All the files produced during the interview process were stored in password-protected RAR archives.
 3. The researcher proposed a certain date and time range to the teachers and they chose the specific date and time that suited their preference. However, if for any reason they were not available in that period, they were able to agree on a more suitable date and time. The location was usually the school where they teach at. Although, if they wished, they could propose a different location within Northampton.
 4. If the teachers did not feel confident that they would be able to answer all questions straight away, they were able to receive them via email before the interview in order to prepare themselves.
 5. On request, the teachers would be able to obtain a transcribed version of the interview for authorisation. It would be sent to them via email in a password-protected, with the password being sent separately.
 6. At the beginning of the interview, the teachers were informed that if for any reason they needed to pause or stop the interview they could do so at any moment.
 7. If the teachers changed their minds about participating in the research or did not like the result of it, they were able to withdraw their consent during the first 2 weeks after the interview.
- Once the teachers learned all necessary information listed above, they were asked to sign a consent form agreeing to participate in the study (Appendix 9).

3.7.3. Ethics Policy for Data Collection through Twitter #CASchat

Due to the fact that #CASchat discussion is publicly available to any person with a Twitter account and it is run by Computing at School organisation, using it as a source of data does not seem to require an ethics policy.

However, it is worth stating that the data used and quoted in this paper has been anonymised, as personal details of the participants are not relevant to the research.

Chapter 4: Data Analysis

4.1. Introduction

This chapter presents a comprehensive analysis of the data that was collected through the surveys with tutors and computing professionals, interviews with teachers and #CASchat. A close examination of the dataset will help to illuminate key themes and patterns that address the research objectives outlined in the previous chapter and ultimately allow to find what industry practices are relevant and could be adapted in school environment.

4.2. Tutor Surveys

The aim of these surveys was to gain a better understanding of computing tutors' opinion about the secondary education in this field, based on their experience with first-year students. Due to the introduction of the new computing curriculum in September 2014, the survey was conducted twice, in order to be able to observe the changes.

The participants were tutors that teach computing at selected British universities and they are referred to as "respondents" or "tutors" in this section (4.2. Tutor surveys).

The first survey was conducted in April 2014 and it was completed by 52 respondents, an 11% response rate. The second survey was conducted four years later, in 2018 and was completed by 31 respondents, a lower response rate of 6.5%.

In both surveys, in the "About you" section tutors were asked to specify their field of computing. This was an important information that affects the analysis of the main questions – a greater diversity of backgrounds guarantees a more general view of computing education, without focusing on a specific field.

The data presented in the charts below shows which fields of computing the participants of each survey identified themselves with. The largest group in each survey were tutors from software engineering, however, their number was small in comparison to all other fields put together. This confirms that there was a great diversity among the participating tutors, which is important for the accuracy of the results.

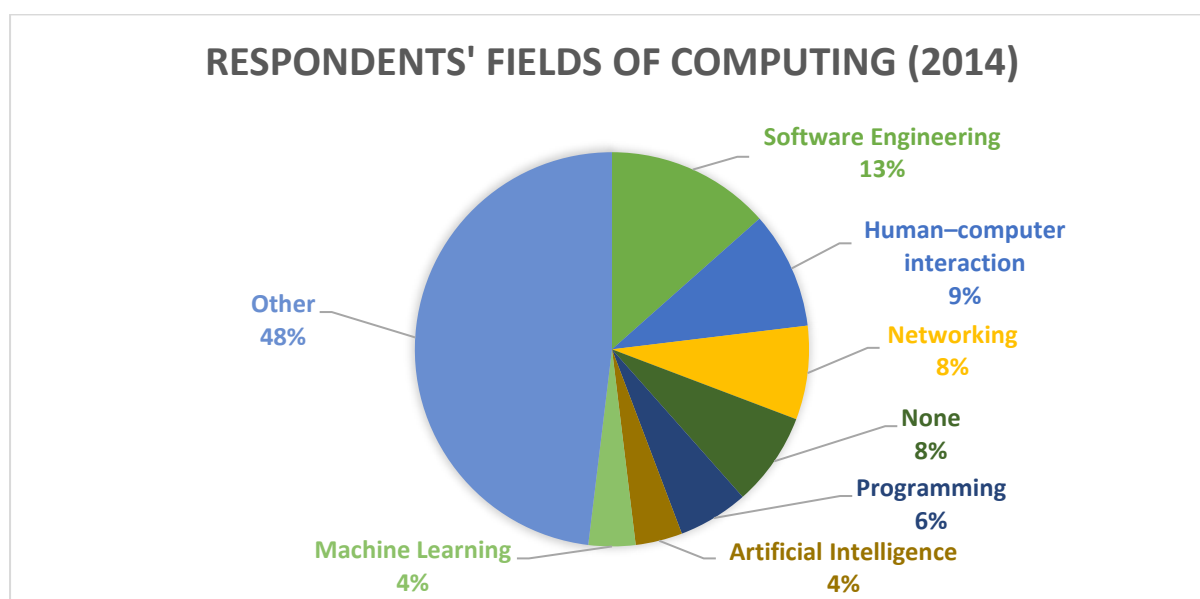


Figure 4.1 – Respondents’ Fields of Computing (2014)

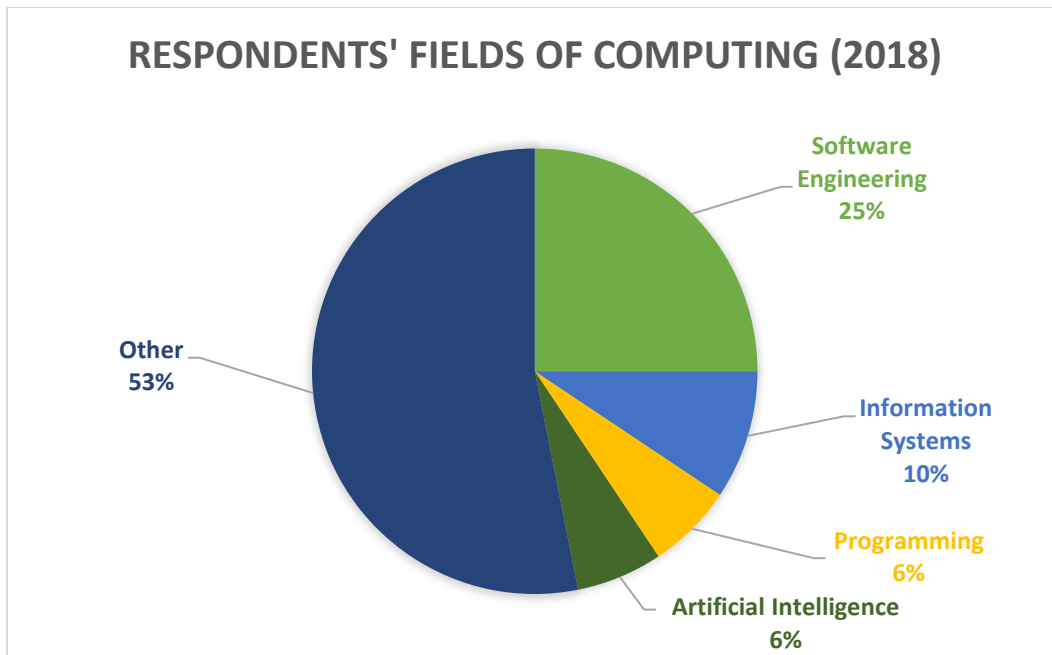


Figure 4.2 - Respondents’ Fields of Computing (2018)

4.2.1. Higher Education Entry Level Computing Skills Assessment

In this question, tutors were asked to assess their students’ knowledge and skills in 9 different fields of computing, by selecting an option on scale from “no expertise” to “considerable competence in the area”. As mentioned in Chapter 3: Methodology, for the purpose of analysis, the options were assigned numbers from 1 to 4 or 5, allowing to calculate the “score” for each statement.

The following two charts present what were the most commonly selected answer options for all skills within this question in 2014 and 2018. This allows to understand the participants’ general attitude, before moving to more precise analysis.

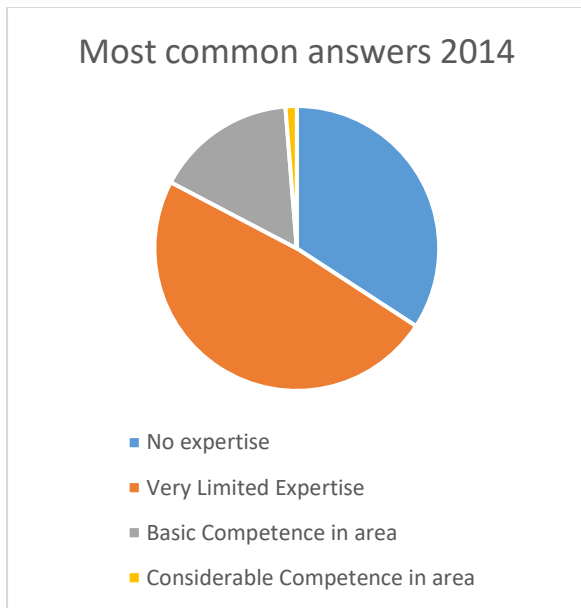


Figure 4.3 - Most Common Answers to Question about First-Year Students' Skills 2014

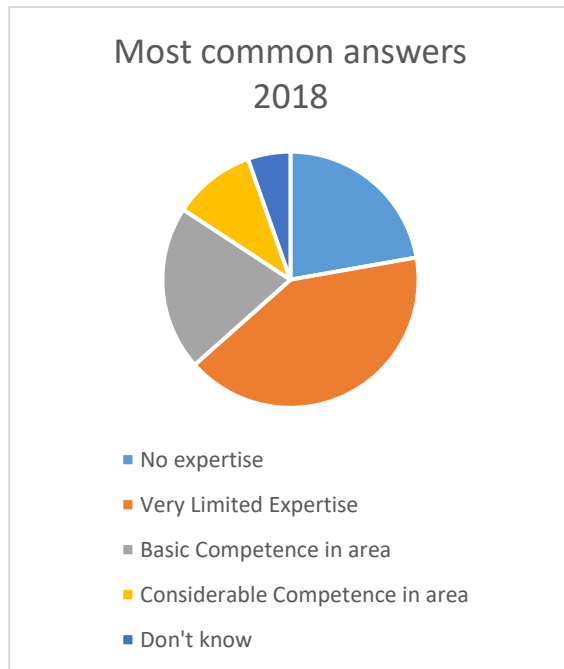


Figure 4.4 - Most Common Answers to Question about First-Year Students' Skills 2018

The answers from both surveys show that in most cases, first-year students have a “very limited expertise” or “no expertise”. However, within the 4 years the percentage of positive answers has notably increased, showing that more students seem to have “basic” or “considerable competence in this area”.

The next chart presents a comparison of scores gained by every field in each survey:

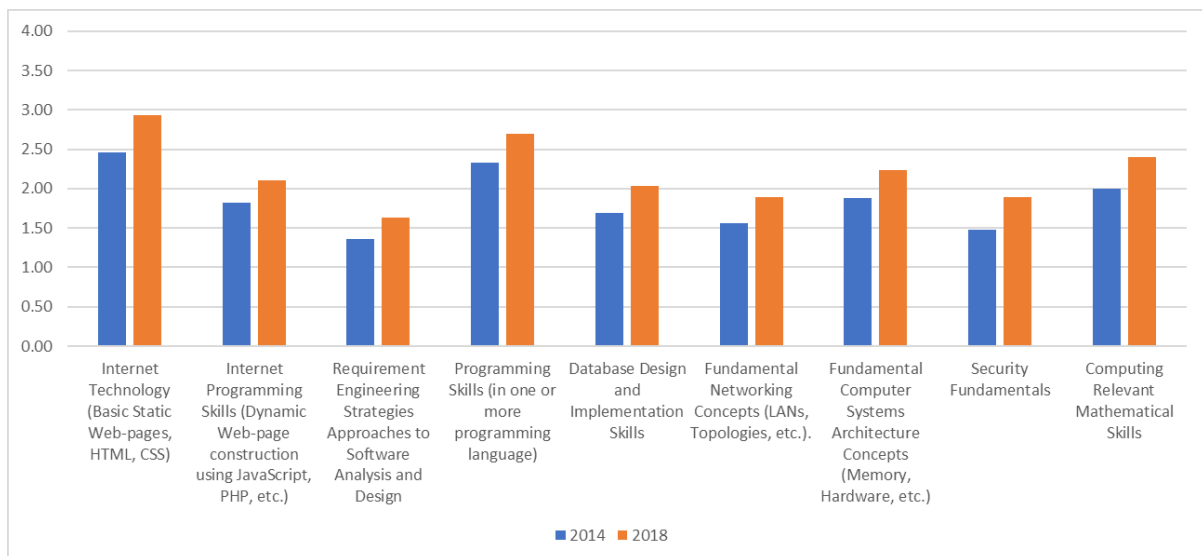


Figure 4.5 - Higher Education Entry Level Computing Skills Assessment (Score Comparison)

The scale on the Y-axis represents the score of each skill, expressed in “points” that correspond to the numbers assigned to each answer option, allowing to read the average rating of the skill.

The scores were very consistent in both surveys – the differences between skills did not change drastically. All of them noted a very similar increase within the 4 years, ranging from 15% to 28%.

In the 2014 survey, tutors stated that their students had “very limited competence in the area” (≈ 2 points) at best. In two of the fields: security fundamentals and software analysis and design, they did not have any expertise at all (≈ 1 point). The top 3 skills were in internet technology (2.46 points), programming (2.33 points) and mathematics (2 points).

Students’ skills seem to have improved over the 4 years – in 2018 scores for all fields have increased. Tutors reported that now their students had “basic competence” (≈ 3 points) in two areas: internet technology (2.93 points) and programming (2.70 points). In all other areas they were reported to have “very limited competence” (≈ 2 points).

As per research by the Royal Society (2018), although the new curriculum was the right step in bringing a change to the field of computer science in Britain, the entry level skillset for undergraduate pupils was limited in their first year, as also apparent from the results above. To cover this dearth of skillset, the new curriculum focuses on the following three aspects of computing, without which any computing knowledge is incomplete; information technology, digital literacy and computer science. To enhance this gap, it is essential that students begin to learn computing at an early age.

4.2.2. Quality of Computing Education within Schools

The aim of the next question was to understand what university tutors think about different aspects of secondary computing education. The participants were asked to select their level of agreement with 7 positive statements about computing in British schools.

In general, they disagreed with the statements, with the most common answer in both surveys being “disagree”. In 2018, however, tutors’ views seem less negative – the level of strong disagreement has dropped by 12%. Also, a lot of tutors used the “don’t know” option which was not present in 2014.

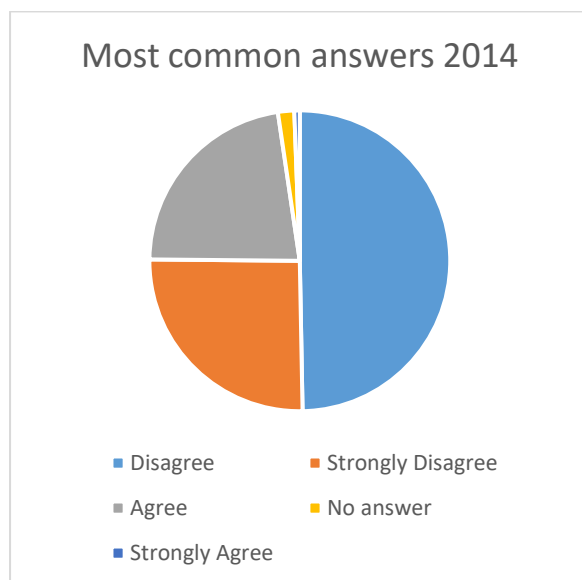


Figure 4.6 – Most Common Answers to Statements Regarding Aspects of Computing Education in Schools 2014

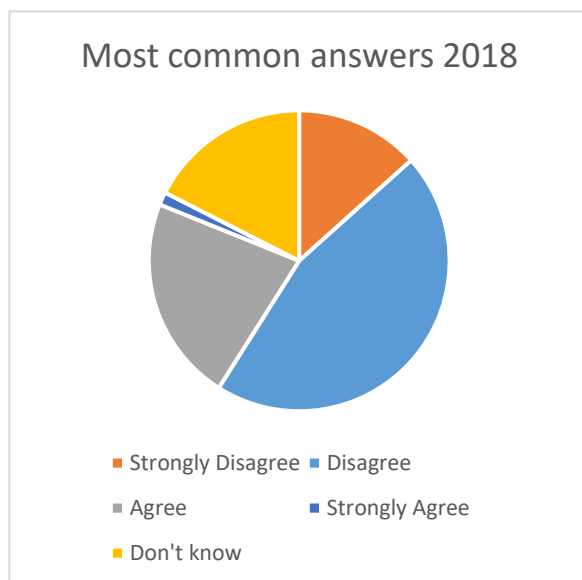


Figure 4.7 – Most Common Answers to Statements Regarding Aspects of Computing Education in Schools 2018

The next chart compares the levels of agreement – or rather disagreement – with each statement from both surveys. Tutors seem to have quite a negative opinion about current computing education in schools. All of the statements received a low score, under 2.5 which is the equivalent of “disagree”

(≈2 points). In 2014, tutors expressed a strong disagreement when asked if the curriculum was good enough to prepare pupils for industry work.

Within the 4 years, two areas remained unchanged: the quality of computing education and the quality of teachers. The answers show that the tutors could not see any visible improvements in these areas.

They did see, however, a slight improvement when it comes to the content taught in schools and first-year students' knowledge, which matches with the answers given in the previous question. A more outstanding improvement can be seen in the statements related to the computing curriculum. Tutors believe that the new curriculum allows to better prepare pupils for higher education or work in the industry. Since the introduction of the curriculum their agreement with the statements has grown by 22.9% and 28.9% respectively.

As the results clearly show, computer science in UK schools has been a roller-coaster ride. Corroborated by research conducted by Brown et al. (2014), it can be seen that in the 1990s and the early 2000s, computing education began to vanish from the UK schools. It was replaced by (ICT) Information and Communication Technology that was more focused on the use of technology rather than creating it and the computing basics required to do so. The research in this area also suggests that the tutors were barely able to tell the difference between using a computer and programming a computer. However, the past decade has brought a welcome change in the perspective of the government as well as computing educators. Computing has the power to develop transferrable skills such as computational thinking as well as other digital and modern skills.

The last statement noted a slight, but clearly visible decrease of level of agreement. This may suggest that while the resources used in the classroom were adequate for ICT, they are less adequate for computer science.

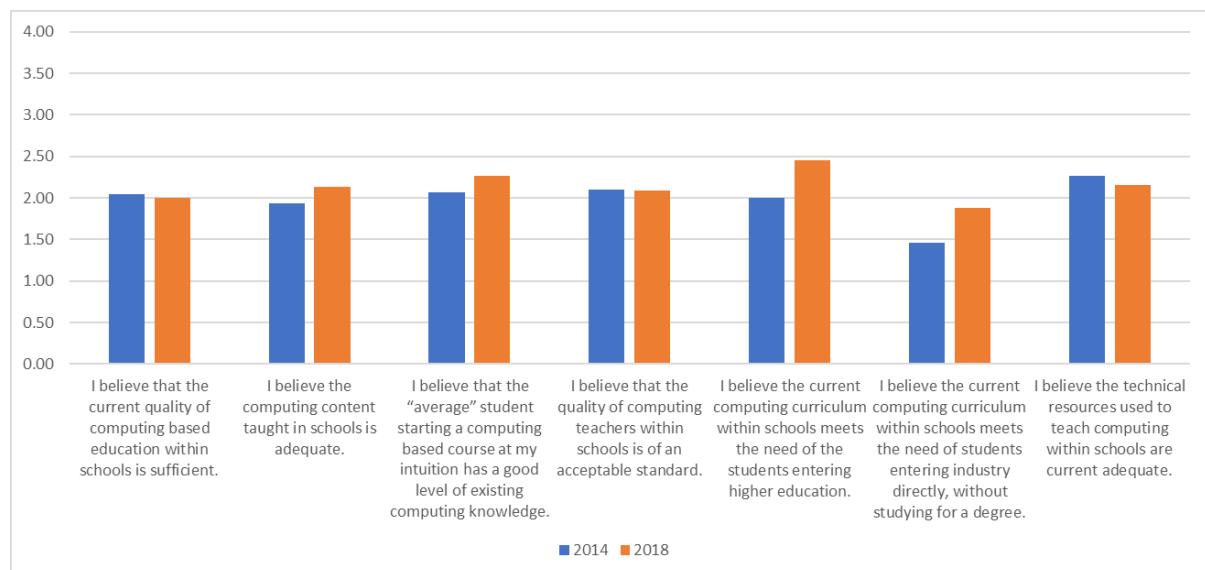


Figure 4.8 - Opinions on Different Aspects of Computing in Schools

4.2.3. Future Improvements to Computing Education within Schools

The final question presented 6 different ideas on how could computing in schools be improved. Participating tutors were asked once again to express their level of agreement with each statement.

The chart below presents a comparison of their answers:

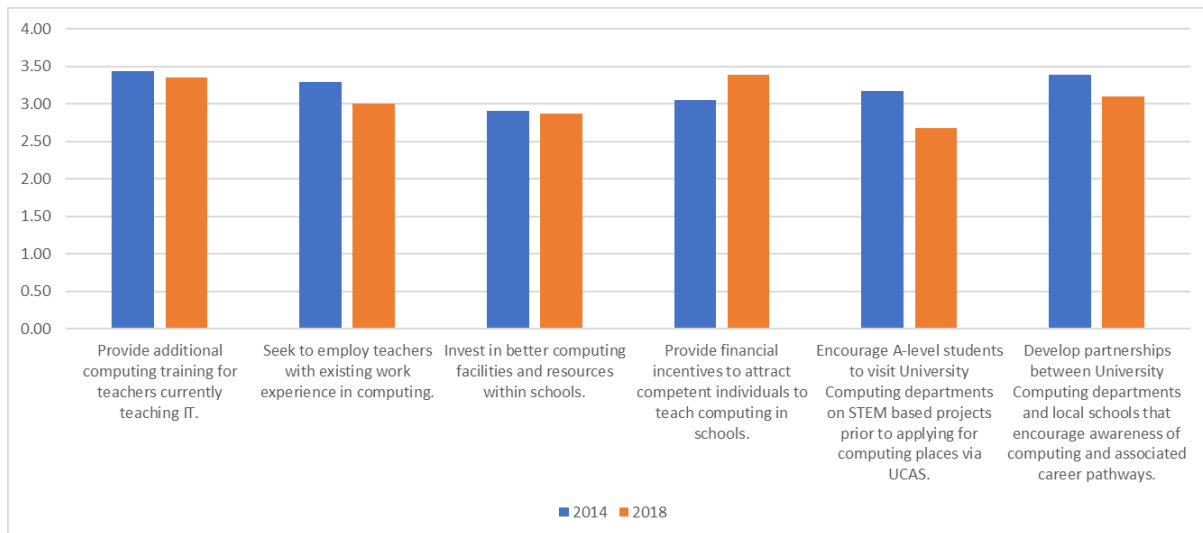


Figure 4.9 - Rating of Improvement Ideas

Tutors generally had a more or less positive attitude towards the proposed ideas. In general, it can be said that in 2014 and in 2018 tutors agreed with all ideas, as their level of agreement was within the “agree” range (≈ 3 points). In 2014 most of the statements, with just one exception, were rated above 3 points. The only idea with lower level of agreement was investing in better facilities, and their opinion remained unchanged after 4 years.

In 2018 most of the ideas received lower scores. Tutors were slightly less positive, especially about encouraging pupils to visit computing departments, developing partnership with universities, and employing teachers with industry experience. The drop was not drastic, varying from 2.4% to 15.6%.

Only one idea received a notably higher score than before – to provide financial incentives to attract more competent teachers (11.1% increase since 2014). This could be interpreted that there are not enough teachers that can handle teaching by the new curriculum.

As elaborated by the results, there is plenty of room for improvement for teaching computing education. Research by Sentance et al. (2013) reveals three major areas where there is definite need for improvement. The first one involves upskilling of the existing teachers. Existing teachers have only been teaching ICT without computer science. Therefore, they require upskilling in order to groom themselves in the field of computer science. The second area of improvement pertains to training new teachers. After completing the recruiting requirements, the next step is to train the new teachers onboarded to teach computing. The third area is curriculum and its understanding. The new curriculum needs explanation and clear comprehension of the curriculum (Sentance et al., 2013).

4.2.4. Respondents’ Comments

In both surveys, tutors were given the opportunity to leave a comment. 25% of them decided to use this option in 2014, and four years later – 22.5%. They shared their opinion on different aspects of computing education in schools and provided suggestions, as well as good-luck wishes. The sections below summarise several chosen comments that contained opinions seen as valuable for the research.

4.2.4.1. The 2014 Survey

One of the tutors pointed out that the level of computing education in British schools was variable – some of them were very competent, while others struggled. Because of that, weaker schools would struggle with delivering the material included in the new curriculum. The author of the comment

believes that “the biggest limitation is time and resources” – staff require more time to develop new material and schools need to buy new equipment.

Another respondent sees computer science as a very useful subject for all those who wish to study or find a job in the industry. On the other hand, “IT fits the needs of a lot of people”, therefore it is important to keep the balance between the two subjects.

An interesting opinion was shared by another tutor, who believes that the industry should play a bigger role in computing education: “If industry were able to be more directly involved with schools that would be great”. Partnerships with university are also seen as a positive thing, but they may use a lot of academic time.

The author of the last comment thinks that secondary computing education is in fact not particularly badly taught. According to the respondent, the problem lies with the curriculum, which focuses “too much on what it mistakenly thinks are the <needs of industry>” – such as “in-vogue” software, programming languages and architectures. The author believes that instead schools should be teaching core knowledge and skills that will not become obsolete and will help pupils with learning if they decide to follow this career path.

It is worth noting that some of the tutors used the comment section to complain about the lack of “don’t know” option in some questions. This was taken into consideration and the option was added in the next survey.

4.2.4.2. The 2018 Survey

As there were few participants in the second survey and even fewer of them decided to comment, only two opinions were chosen for analysis.

The author of the first comment shared their opinion about programming languages that are popular in school education – Java and Python. The respondent points out that people learn them on their own or at university, therefore schools should be teaching something different – a “functional language like Haskell or Scheme”.

Another tutor sees the “change in content” (of the curriculum) as possibly helpful to students. However, the respondent had not observed any improvements as “not enough students are taking these courses”. Therefore, the problem might lay with lack of engagement, rather than the content.

4.2.4.3. Summary of Respondents’ Comments

The responses from computing tutors show that the quality of British education in this field is far from satisfactory. Respondents have a negative view of secondary education both before and after the new curriculum revolution, as can be read from their assessment of student skills and low level of agreement with positive sentences about different aspects of the education. The curriculum itself is seen as a positive change – tutors believe that it allows to better prepare young people for higher education and work in the industry, comparing with 2014. However, this was still not enough to be able to say that school-leavers’ level of computing knowledge is adequate for university or industry.

A consistent improvement can be observed in tutors’ opinion about first-year students’ computing skills – all of them, without exception received visibly better ‘score’ after 4 years. This may indicate a positive trend in computing education, however at this moment it seems that schools cannot be credited for it.

Tutors agreed that there are many good ways of improving the situation in schools. Since the introduction of the new curriculum, they see an increased need for competent teachers who would

be able to deliver new content. In 2018, they were slightly less enthusiastic about interactions between pupils/schools and universities – the statements about school-university cooperation were rated from 2.4% to 15.6% lower, but the reasons behind this are unclear. Perhaps, the need for such interactions is not as urgent as before with the new curriculum in schools, but to answer this question more research would be required.

4.3. Computing Professionals' Survey

The computing professionals' survey was conducted in 2018 among people who worked in the industry as software developers. These people will be referred to as "respondents" or "professionals" in this section. The aim of this paper was to answer 3 key questions:

- How did they learn computing?
- How do they learn and solve problems these days?
- What are their recommendations for teaching programming in schools?

Therefore, the survey was divided into 3 sections, entitled "Your Beginnings", "Learning as a Professional" and "Your Recommendations". For improved clarity, a different colour was chosen for charts in each section: orange, blue and green respectively.

The respondents were chosen through social media (based on public data) and personal connections in order to make sure that they come from relevant background (obtained education in the UK and work or worked in computing industry). The four charts below present key information about the respondents, proving that they were diverse in terms of age, experience and programming language. Due to the gender-imbalance in computing industry, the representation of women in this survey was smaller (15.4%). The gender ratio is slightly better than the one in Stack Overflow survey (Stack Overflow, 2019), where 7.5% of the respondents were women. Other data is in-line with the industry statistics (Stack Overflow, 2018). Because of this, the respondents can be considered an accurate representation of professionals employed in the industry.

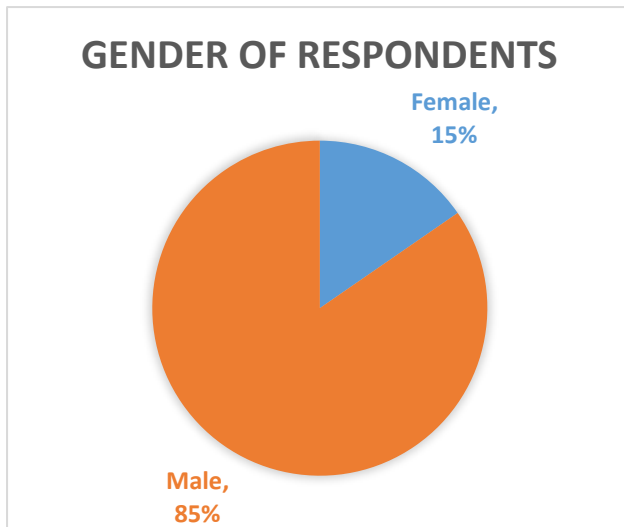


Figure 4.10 - Gender of Respondents (Computing Professionals' Survey)

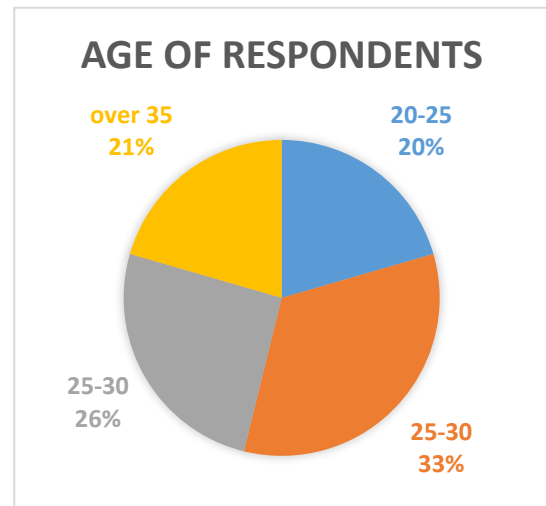


Figure 4.11 - Age of Respondents (Computing Professionals' Survey)

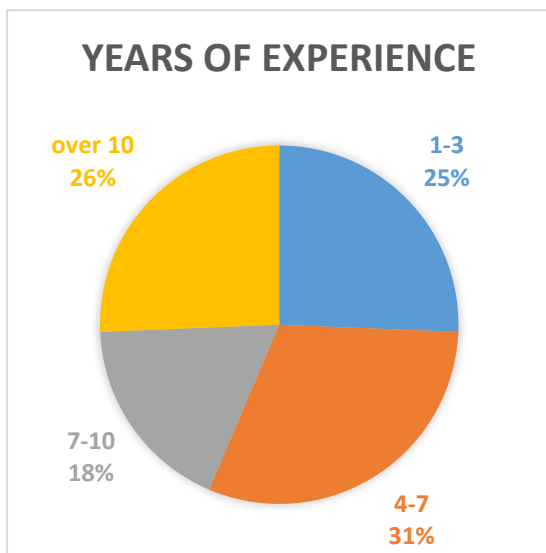


Figure 4.12 - Respondents' Years of Experience (Computing Professionals' Survey)

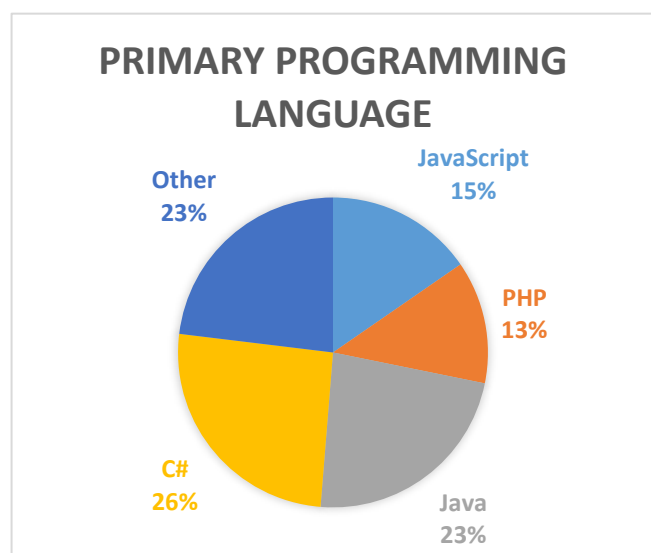


Figure 4.13 - Respondents' Primary Programming Language (Computing Professionals' Survey)

4.3.1. Your Beginnings

4.3.1.1. Initial Interest in Computing

In the first question, the respondents were asked about their initial interest in computing – what started it? The chart below presents their answers:

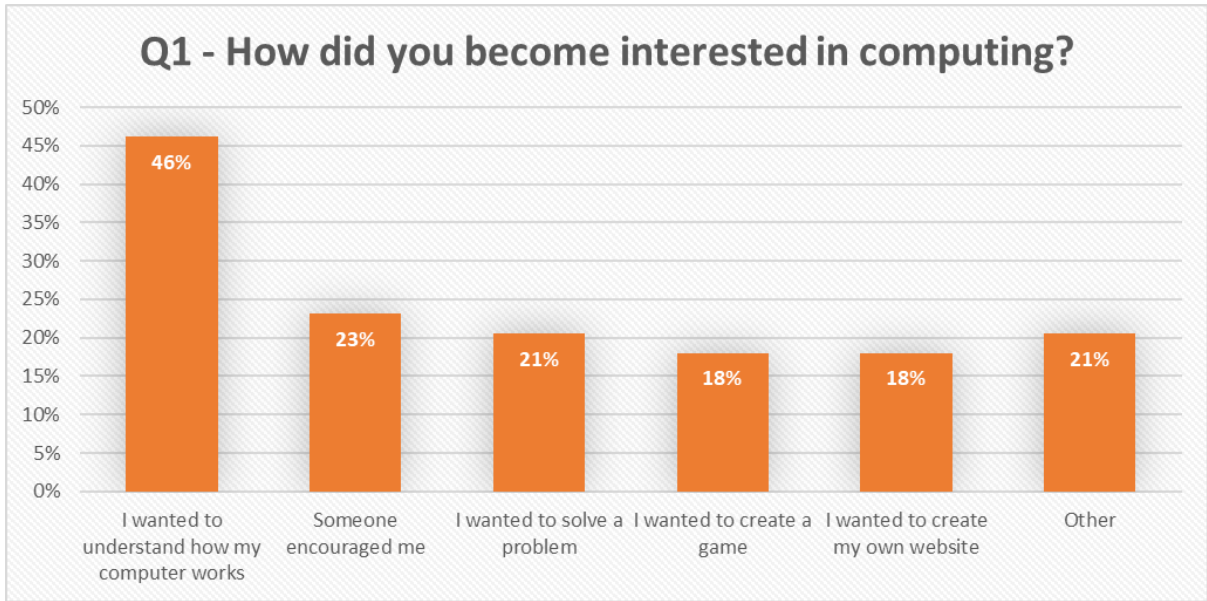


Figure 4.14 – Professionals’ Initial Interest in Computing

Nearly half of the respondents (46%) stated that it was their curiosity: they wanted to understand how their computer worked. 23% of them also had someone who encouraged them to learn about computers. A smaller number (18-21%) wanted to do something more concrete: solve a problem, create a website or a game. Other responses (21%) were surprisingly similar: most people said that their interest in computing began simply with their interest in technology. A few other people were looking for a good career path.

4.3.1.2. Were They Introduced by Someone?

The second question aims to find out whether there was a figure in the respondent’s life that played a role in introducing them to computing. Their answers are shown in the chart below:

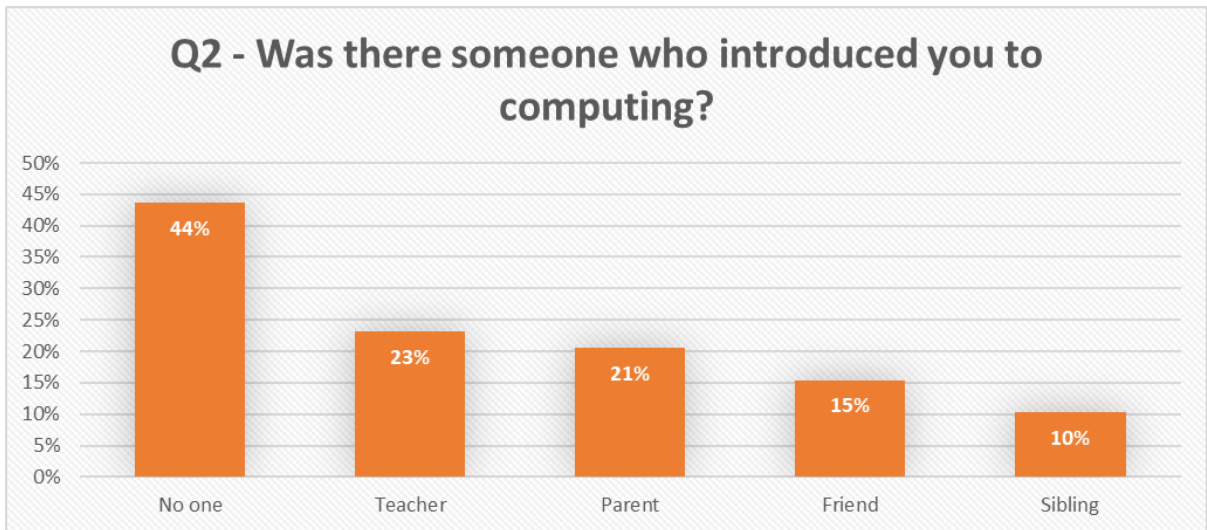


Figure 4.15 - Who Introduced Professionals to Computing?

The responses match the results from the previous question. Nearly half of them (44%) were not introduced by anyone. Some of the respondents (23%) were introduced by a teacher or parent (21%). An even smaller number of people selected ‘friend’ or ‘sibling’.

4.3.1.3. How Did They Learn?

This question seeks to find out how respondents obtained their skills and knowledge in the field of computing – what were their main sources of education?

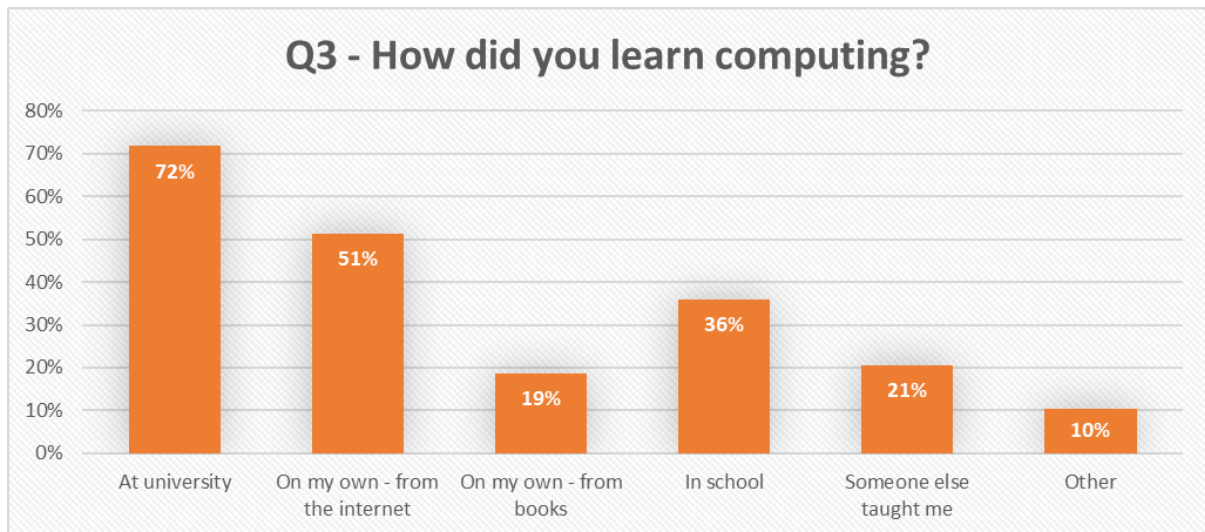


Figure 4.16 - How Did Professionals Learn Computing?

To most people (72%) it was during their time at university when they obtained computing education. Half of them stated that they learnt on their own using the internet. School education was chosen only by a third of the respondents. The least popular answers were: learning from another person (21%) and self-education from books (19%). Only 10% of respondents gave another answer, which included college, afterschool club, magazines or simply mixed method.

4.3.1.4. First Programming Language

Finding out what was the language in which respondents made their first programming steps can contribute to the discussion about what should be taught in schools.



Figure 4.17 - What was Professionals' First Programming Language?

The chart shows that 18% of the respondents declared their first programming language being Java. Java is an open-source, platform independent language with a well-established community, making it a very popular choice in the industry (Stack Overflow, 2018)

A slightly smaller number of respondents (15%) selected JavaScript – a programming language used in web development to create dynamic content. The main advantage of JavaScript is that it does not require installation or an IDE, making it easily available to anybody.

The C-family languages were less popular (8-10%), similarly to BASIC, PHP and Visual Basic (8%). Quite a large number of respondents chose to provide their own answer (18%), however, there was no common language in their responses.

The list of possible answers also contained Python, Pearl and Ruby, but these languages were not chosen by anybody. This could be due to the low popularity of these languages at that time. A comparison of Stack Overflow surveys in 2018 and 2019 (only one-year difference) shows the rapid growth of Python’s popularity among professionals (Stack Overflow 2018 and 2019).

4.3.1.5. Computing at School

The following two charts show what percentage of the respondents attended computing classes during their school education and how they rated their experience with it.

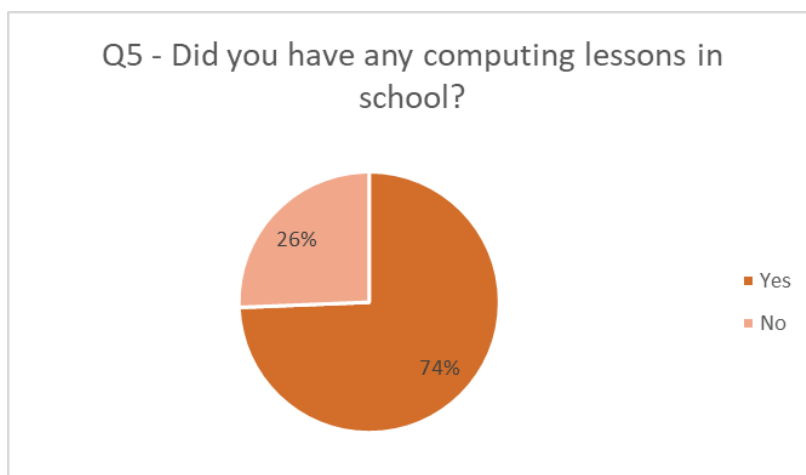


Figure 4.18 - Did Professionals have Computing Lessons at School

Despite nearly three-quarters of people stating that they did have computing at school, in one of the previous questions (Q3) only a third (36%) of them chose school as the place where they learnt computing. This proves that school education was ineffective.

The answers to the next question only confirm this – the average rating of the computing lessons they had was only 2.6 out of 5.

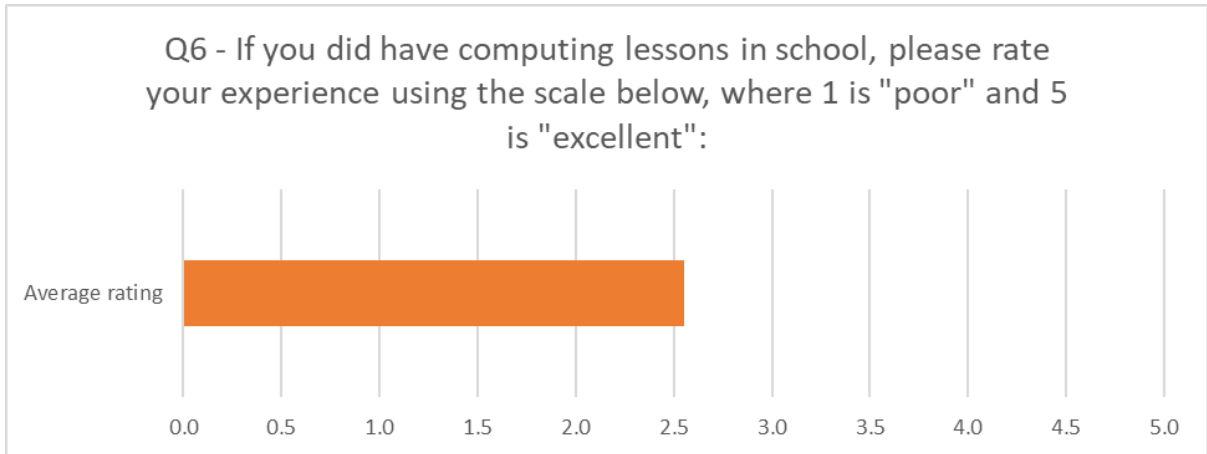


Figure 4.19 - Professionals' Rating of their Computing Lessons

4.3.1.6. Support Tools

The last two questions in the "Your Beginnings" section concerns popular support tools for learning programming. Respondents were asked if they ever used such tools and to rate their experience.

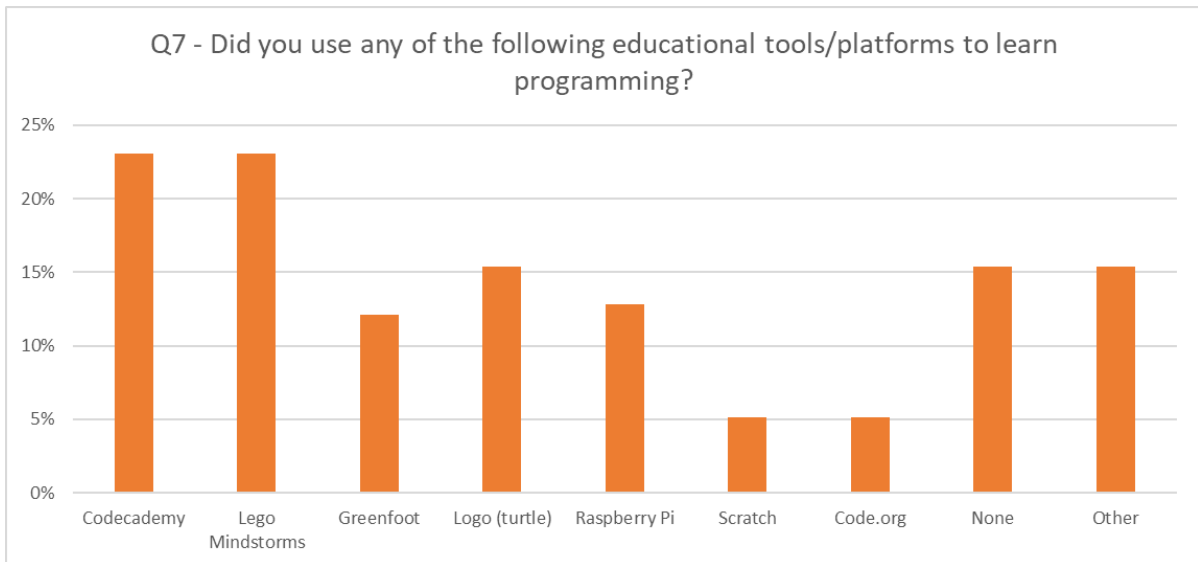


Figure 4.20 - Popularity of educational tools

Most of the respondents declared that they did use a tool during their computing education – only 15% of them skipped the question. The most popular answers were Codecademy and Lego Mindstorms, each of them was selected by 23% of respondents. Less popular were Logo (turtle) (15%), Raspberry Pi (13%) and Greenfoot (12%). The least popular from the provided options were Scratch and Code.org (5% each). Among 'other' answers were e.g. Alice, BBC microcomputer and Stack Overflow.

Respondents have found them quite helpful in learning computing – their average rating was 3.3 points out of 5.

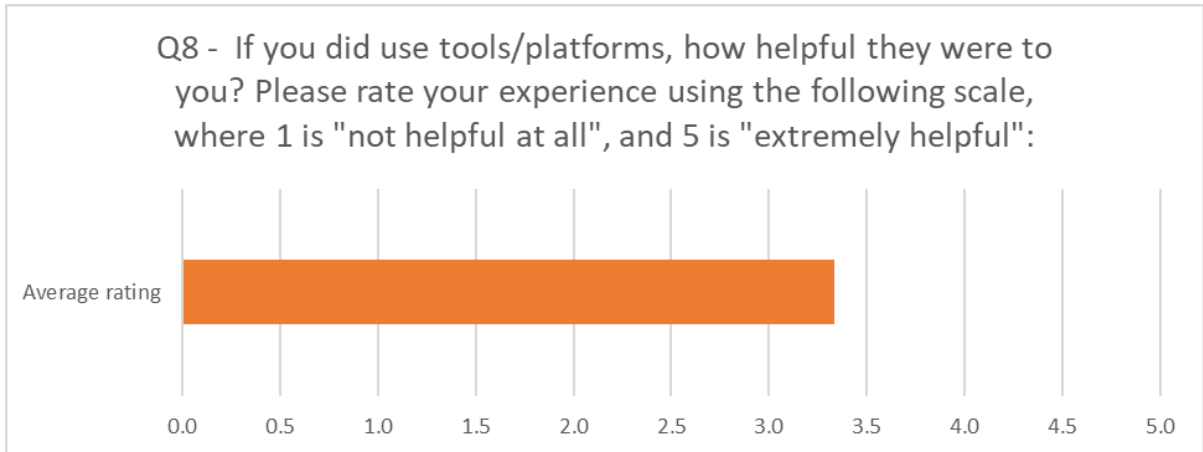


Figure 4.21 - Rating of Educational Tools

4.3.2. Learning as a Professional

4.3.2.1. Expanding Knowledge

The focus of the next section of the questionnaire, "Learning as a Professional", lays on respondents' everyday work and habits. The following chart presents their answers to the first question:

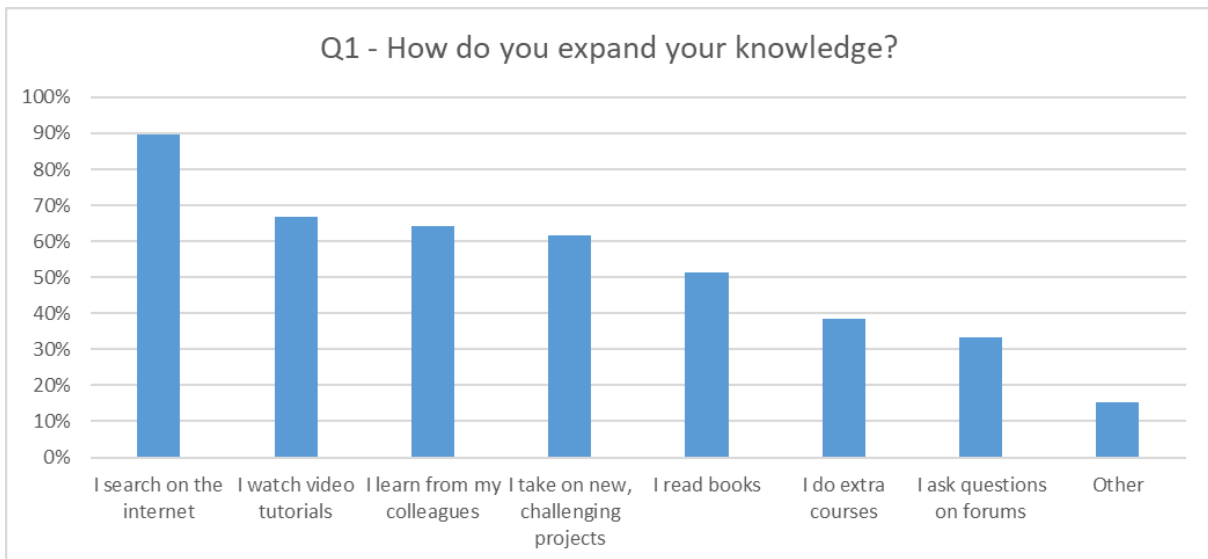


Figure 4.22 - Ways of Expanding Knowledge

Almost all respondents (90%) declared that they expand their knowledge by searching on the internet. A little less popular answers were: watching video tutorials (67%), learning from colleagues (64%) and taking challenging projects (62%). Half of the respondents (51%) stated that they read books.

A smaller number said that they do extra courses (38%) and use forums (33%). Among other answers (15%) were: reading source code, blogs and newsletters.

4.3.2.2. Popular Support Websites

The answers to the next question show what are the most popular websites that industry professionals use in their daily work to learn and solve problems.

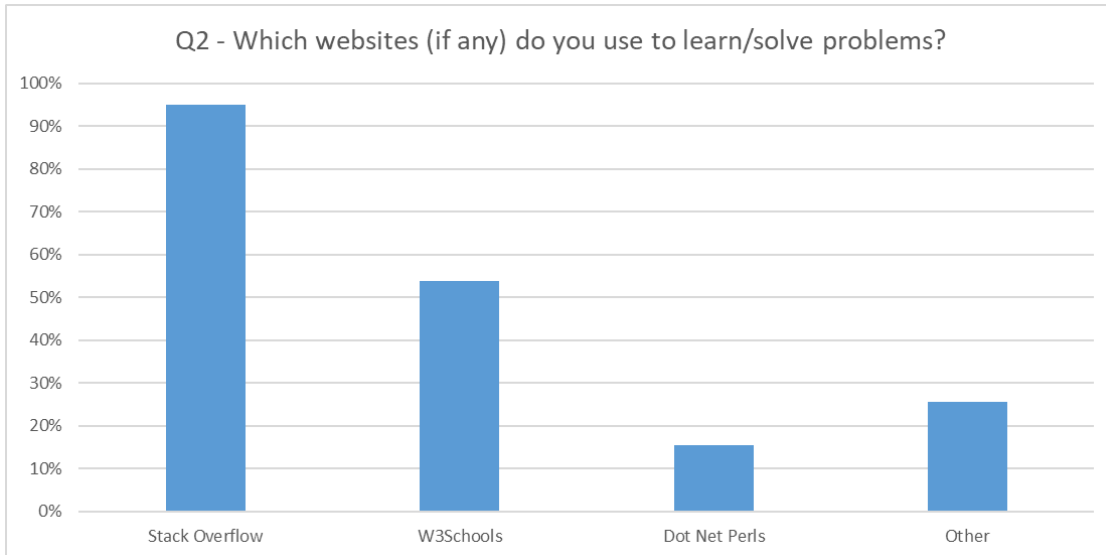


Figure 4.23 - Popular Websites for Developers

The unquestionable and unsurprising winner is Stack Overflow (95%) – a very popular ‘question and answer’ portal for software developers. Second came W3Schools, used by half of the respondents (54%) – a website with references, examples and tutorials focused on web development. Dot Net Pearls, a website with code examples in many programming languages was used by 15% of professionals.

Other answers (26%) included: Java documentation, Microsoft Virtual Academy or Mozilla’s MDN Web Docs. However, the answers were not repeated.

4.3.2.3. Everyday Support

The last question of the second section regards various ways of seeking support in everyday work as software developer.

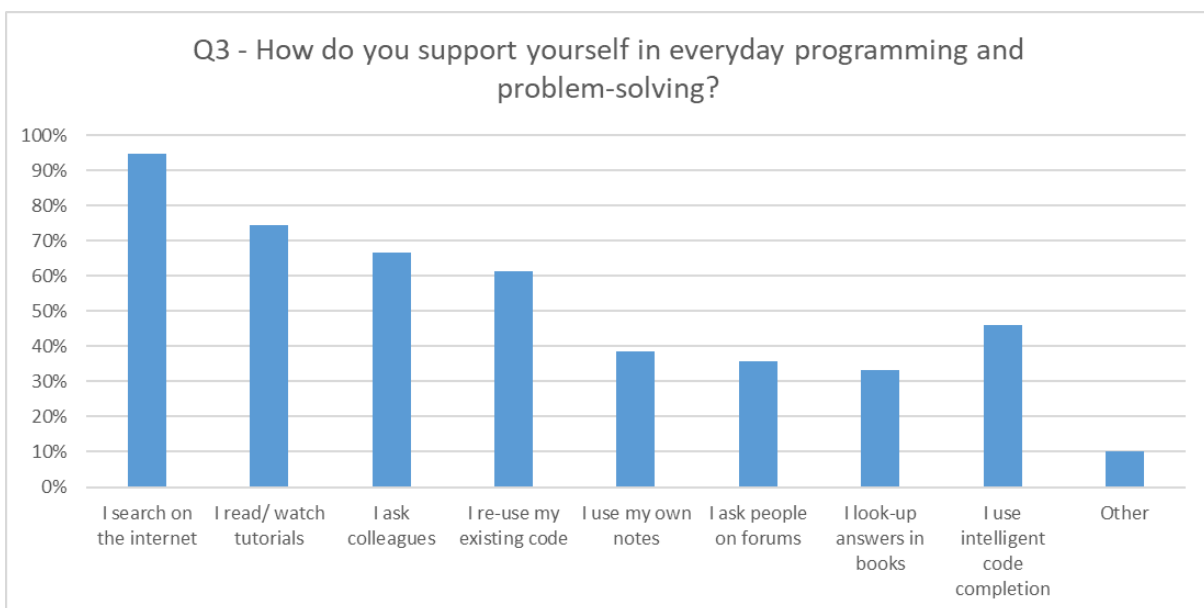


Figure 4.24 - Seeking Support at Work

As can be seen on the chart above, the top answer was “I search on the internet”, chosen by 95% of the respondents. A large number of them also stated that they use tutorials (74%), ask colleagues

(67%) or re-use existing code (62%). Almost half of the professionals (46%) use intelligent code completion.

Less popular methods were using notes (38%), forums (36%) and books (33%). Only a small number of respondents chose to provide other answers (10%), such as browsing Git and talking on IRC.

4.3.3. Your Recommendations

4.3.3.1. Recommendations

The last section titled “Your Recommendations” contains two questions where respondents had to select programming languages and support tools that should be used in computing education.

Professionals believe that the best languages to begin programming are JavaScript (49%) and Java (46%). These were also the first programming languages to most of the respondents, as was found in the first section of the questionnaire. It seems that learning experience with these languages was so good, that the respondents decided to recommend them to beginners. Also, as mentioned before, both languages are open-source and easily available.

The third place belongs to C# (41%), which has become open-source and increasingly popular with Microsoft’s new simple IDE – Visual Studio Code. It is closely followed by Python (36%). This programming language was chosen by quite a large number of respondents, despite none of them used it when they were learning to code. This aligns with the earlier review of educational software (Chapter 2), showing Python’s increasing popularity, especially in schools.

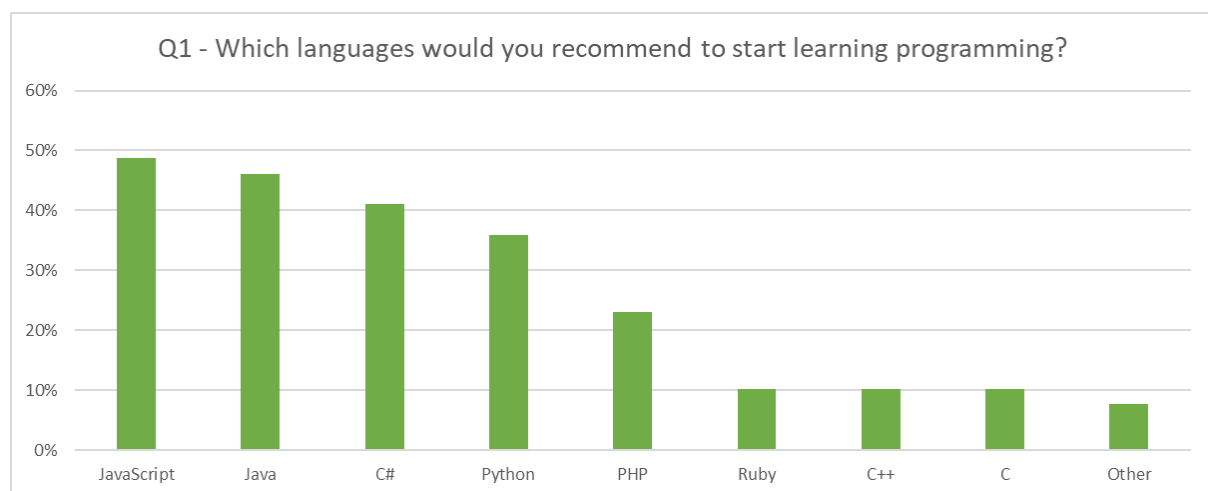


Figure 4.25 - Recommended Programming Languages

The best educational support tool according to the professionals is Codecademy (38%), which is an interactive platform with free coding classes in several languages. Raspberry Pi (31%) and Lego Mindstorms (26%) were also popular recommendations. Scratch, which is very popular in British schools, was chosen by 18% of respondents.

The remaining tools: CodePilot, Greenfoot, Logo, Code.org and Python Tutor were recommended only by a small number of professionals (5-8%). ‘Other’ answers (13%) included: Micro:bit and Kodu.

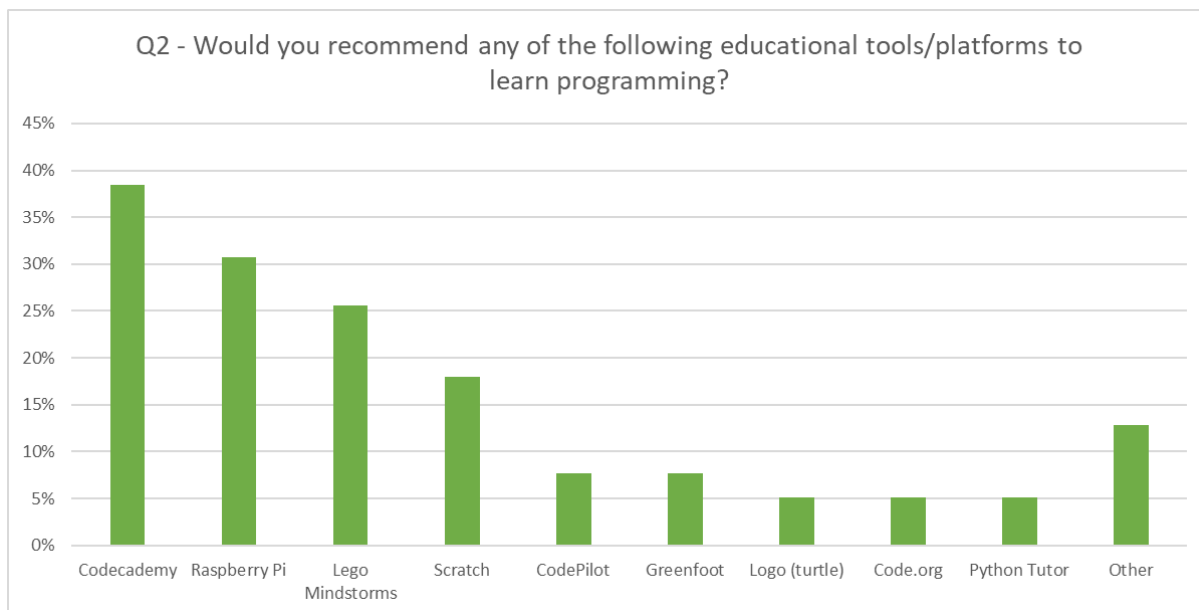


Figure 4.26 - Recommended Educational Tools

4.3.4. Comments

Professionals were also given the opportunity to provide their tips and advice on how computing should be taught in schools. Comparing to the tutor surveys, a large number of respondents (66.6%) decided to use this option to share their thoughts.

Many of the comments shared similar pieces of advice and ideas. The most common ones were grouped and counted. The top ones are presented in the table below:

Advice	Details
Use real-life examples	This was the most frequent advice, mentioned in 30% of comments. Professionals strongly suggest using real-life examples to teach and give pupils real-life problems to solve. They believe that asking pupils to create something usable will build their interest in computing.
Make it fun	19% of respondents who left a comment said that it is important to make computing lessons fun and evoke pupils' curiosity. This can help to "naturally develop their skills and ability".
Use tools	19% of comments contained suggestions to use various tools "which can improve learning experience", such as Lego Mindstorms, Raspberry Pi or Greenfoot.
More practice	Another popular advice was to do more practical exercises – mentioned in 15% of the comments. Professionals see programming as "a practical process/art" and suggest that there should be "more practical assignments than questions/answers exams".

Table 4.1 - Professionals' Advice: How to Improve Computing Education

Other interesting advice included:

- "Ensure that the technologies and techniques taught by the teachers/lecturers are up to date and not 10+years old"
- "Hire teachers who have worked in the industry"
- "Weekly after school coding nights and all that jazz should be done more"

- “(...) soft skills, how to build and maintain networks and relationship building are key ways to learn”

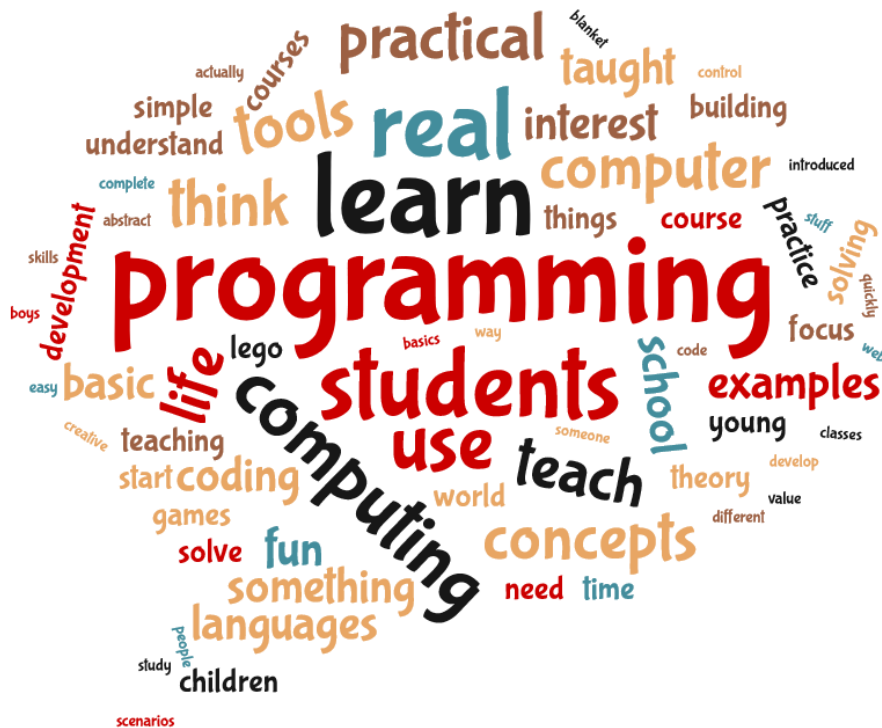


Figure 4.27 - Word Cloud Generated from Comments, Showing Most Common Words

4.3.5. Advice to Yourself

Another open question for the professionals was:

“Imagine that you can send messages to the past. What practical tip would you send to your younger self to make the learning process easier?”

The aim of this was to find out what were the most common problems that they once had to face, what was difficult or discouraging, and what would they do about it from today’s perspective, with the knowledge, skills and experience they have now. The question was answered by 82% of the respondents.

The most common pieces of advice that they wrote were grouped and counted. The top ones are presented in the table below:

Advice	Details
Start early and make small steps	A lot of respondents (31.2%) advised to “start earlier, and start simpler” – “little pieces here and there when young”. According to them it is important to “start working on small projects” and “get it right, don’t rush the basics”. Some suggested to create a “very small simple piece of software” and to “break things down into the smallest single units”.
Learn from different resources	21.8% of respondents suggested using various method to learn and solve problems. One of them was to “speak to other people who have started programming”, because “you can learn from people who have made the mistakes already”. Someone even stated that “the best way to learn and problem solve is to reflect with others”.

Despite most professionals having some type of computing lessons at school, it is the university that was considered to be their most common 'source' of education in this field. The quality of school education was considered average. They had a noticeably more positive opinion about educational programming tools and most of them admitted that they did use one or more during their education. Respondents' recommendation for such tool was mainly Codecademy online platform, followed by programmable devices – Raspberry Pi and Lego Mindstorms. These findings resonate with the theory of constructionism (Papert, 1993), as these tools can provide activities that promote active learning and engagement with real-life objects.

In terms of programming language, the top recommendations for beginners were JavaScript and Java. Both of them remain very popular among developers (Stack Overflow, 2019) and were the first programming languages that the majority of respondents learnt. It is worth noting that C# and Python were also highly recommended, despite only a few (C#) or none of the professionals (Python) learnt them in the past. These results do not quite reflect the current situation of the respondents – most of them nowadays use C# or Java. This could suggest that a language, despite being useful in the industry, might not be the most optimal choice for beginners. Pedagogical approaches that focus on core programming concepts and transferable skills may be more beneficial in the long run.

In their daily jobs, professionals mostly rely on the internet – both for learning and problem-solving. Their number one website is Stack Overflow, which only confirms the findings from literature review. Respondents also support and learn from each other a lot – watching tutorials and talking to their colleagues was important to them in expanding knowledge and solving problems. Many of them considered taking on a challenging project as a good way to learn as well. For problem-solving, a large number of respondents tend to re-use their existing code and half of them rely on intelligent code-completion.

This emphasis on collaborative learning and peer support aligns with Vygotsky's Social Constructivism theory. Vygotsky suggests that social interaction plays a vital role in cognitive development, and learning is most effective within a zone of proximal development (ZPD) through collaboration with more knowledgeable others (Vygotski, 1978). Learning from colleagues, working on projects together, and seeking help on websites like Stack Overflow are good examples of these social learning principles.

Answers to the open questions provided a lot of interesting and useful advice from the professionals. They see practice and solving real-life problems as a crucial part of good computing education. Many of them recommended using a variety of resources to support learning process: programmable devices, IDEs, online resources and websites. However, the most important resource is other people – respondents see human interaction as the key to effective learning and problem-solving.

4.4. #CASchat

The #CASchat Twitter discussion took place on 18th September 2018. The participants were mainly computing teachers and educators who were interested in the topic. Within this section, they are being referred to as "participants" or "Twitter users".

At least 75% of them were based in the UK and 17% abroad. The map below shows the approximate location of the British participants, and the chart next to it – all participants' locations based on their public profile data from Twitter:



Figure 4.29 - Map of #CASchat Participants' Locations

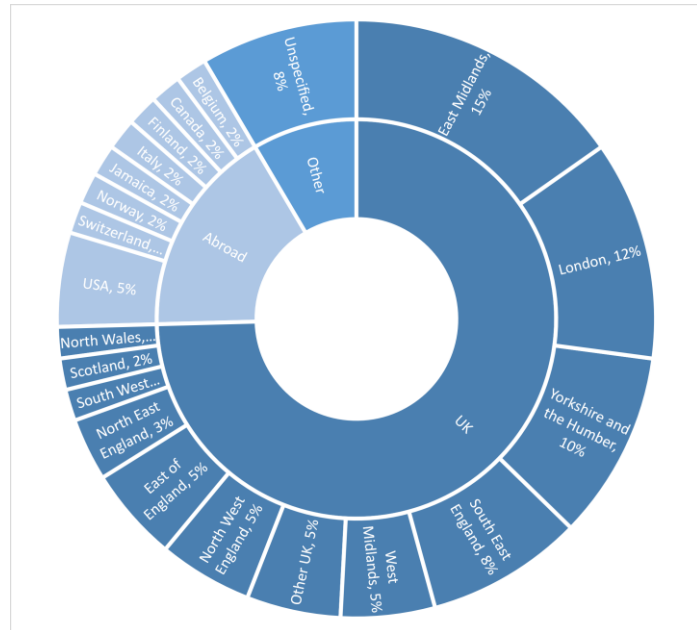


Figure 4.30 - Chart with #CASchat Participants' Locations

The participants discussed on several topics related to computing education: learning from the industry, challenges in introducing pupils to text-based programming and ways to support beginners who learn programming.

The table below presents the most common answers to the questions asked during the chat session.

Q1: How do you think the practice of industry professionals (particularly approaches to learning and problem-solving) differs from those used in the classroom?		
Students have less time and need to move forwards	Tweets: 3 Retweets: 5	Participants stated that <i>“time has a big factor to play”</i> and that <i>“time is the issue”</i> . Pupils and their teachers are in a rush and have <i>“little time to reflect and use the skills learnt”</i> . Unlike industry professionals, whose <i>“skills and knowledge are tested and refined in a circular manner”</i> , <i>“in the classroom, we need to move on”</i> .
Different goals/approaches	Tweets: 5 Retweets: 2	One of the participants stated: <i>“Very different situations with very different goals.”</i> Others mentioned that in the industry it is important to be <i>“able to fix things quickly”</i> . Professionals <i>“are more willing to take risks”</i> and <i>“don't have to worry about their approach being one that satisfies exam criteria”</i> . They are also believed to <i>“do much more group work”</i> .
Q2: As educators, what lessons do you think we can learn from industry that could be applied to the classroom?		
Teamwork	Tweets: 6 Retweets: 3	Several different people said that we should learn <i>“team work and collaboration”</i> and that there should be <i>“More emphasis”</i> on projects that require working together. Pupils should <i>“solve problems collaboratively”</i> and work in teams.
Real-life problems	Tweets: 2 Retweets: 3	Another important lesson is to <i>“use real world problems as stimulus work”</i> . One of the participants also said that lessons

		should be <i>“cross curricular”</i> and that <i>“Real world problems help this, as they are very rarely solved using just one subject”</i> .
Q3: What are the main problems/challenges faced by children when moving from blocks to text-based programming?		
Syntax	Tweets: 11 Retweets: 7	The biggest problem for pupils who move to text-based programming seems to be the syntax, as it was mentioned by a great number of participants. They said that syntax is <i>“causing frustration”</i> as it is <i>“much harder to code through experimenting”</i> . Another person said: <i>“Pupils become paralysed by spelling, spacing, syntax and forget the bigger picture/algorithm”</i> .
Poor typing skills	Tweets: 5 Retweets: 1	Several participants also reported that pupils <i>“lack typing skills”</i> or don't know keyboard shortcuts. One person, apparently a teacher, said that their students needed <i>“re-teaching typing skills”</i> .
Text-based programming is less appealing and less forgiving	Tweets: 4 Retweets: 2	Blocks have <i>“aesthetical shapes & colours”</i> so they are more user-friendly. Also, <i>“It is harder to make mistakes with blocks”</i> . The reason for that is <i>“Blocks are visual and will usually do something if not what was intended even when coded incorrectly”</i> . Programming using text is much more demanding and less appealing, which <i>“can often create a barrier”</i> .
IDE	Tweets: 2 Retweets: 2	It was also mentioned that <i>“cumbersome IDEs”</i> can be a problem and that <i>“syntax errors can probably be overcome with the correct IDE and syntax highlighter”</i> .
Q4: When is the best time to move from blocks to text-based programming? What are your tips for making the transition easier?		
When students have learnt the basics	Tweets: 2 Retweets: 1	Two participants believe that the best moment to switch is when pupils <i>“have a strong understanding of programming constructs”</i> or when they have <i>“a good foundation on the basic concepts”</i> . The first person warned that moving pupils too early <i>“will put them off”</i> , while the second one thinks that <i>“Holding on to blocks for too long might not be a good thing”</i> .
Key stage 3	Tweets: 4 Retweets: 5	Several people mentioned <i>“end of KS3”</i> , but someone also said that <i>“it depends on previous experience”</i> .
Key stage 2	Tweets: 4 Retweets: 5	Other participants think that the best moment is <i>“End of KS2”</i> . A few of them mentioned that they have tried it in their classrooms, but it looks like not all students are ready by then: <i>“I have tried in Y6 - and for some students (the ones who go to code club, etc.) they made the transition well, but for others it was certainly too soon.”</i> <i>“I have tried this with my code club and it works better with the Y6”</i> .
When students have mastered blocks	Tweets: 2 Retweets: 2	For others, the criterion is how well students can programme using blocks. The right moment is once they have <i>“mastery using block-based coding”</i> and are able to <i>“create block-based code”</i> themselves. Another participant said that, <i>“Once they know how to code, moving to text is easier than learning to code in text”</i> .

Q5: What features do you think a beginners' text-based programming language should have?		
Syntax checking	Tweets: 2 Retweets: 4	Perhaps the most wanted feature was <i>"simple syntax"</i> and to have <i>"some form of syntax checking"</i> .
Friendly error messages	Tweets: 2 Retweets: 3	Another required feature was <i>"better error catching and error messages"</i> , <i>"that make sense to a beginner"</i> .
Switching between blocks and text	Tweets: 2 Retweets: 3	Some people would like to have the <i>"ability to switch between blocks and text on the fly"</i> and to be able to <i>"find logical links between Scratch and text-based"</i> .
List of commands	Tweets: 1 Retweets: 2	Someone said that <i>"A list of commands"</i> would be useful to pupils, as the problem they face is: <i>"I don't know what's available so how can I use it?"</i>
Scaffolding	Tweets: 1 Retweets: 2	Another participant mentioned that <i>"it needs scaffolding"</i> that would introduce <i>"programming principles & procedures"</i> . There should also be a tool to create <i>"pseudo code / flow diagrams"</i> .

Table 4.3 - Twitter #CASchat: Questions and Summary of Answers

It is easy to notice that the participants of the survey share some ideas with the professionals who took part in the survey. The first thing that stands out is "real-life problems" – for #CASchat participants it was one of the things they believe schools should learn from the industry. This was also the top advice from professionals to schools. Another similarity is "teamwork and collaboration", which was the number one lesson for #CASchat participants and can also be read from the professionals' survey data. Many developers stated that they seek help from their colleagues and from the internet, especially Stack Overflow, which is a collaboration tool. Some Twitter users also recognise the need for a good IDE that would help in overcoming the difficulties of block-to-text transition.

These transition difficulties are mainly syntax (which was mentioned in the largest number of tweets during the whole chat) and poor typing skills among pupils, combined with a less entertaining look and feel of textual programming in comparison to blocks. The reason behind this could be the increasing popularity of mobile computer devices, such as smartphones and tablets. A 2018 study conducted by GlobalWebIndex shows that the number of smartphone users noted a rapid growth since 2015 and that people prefer to use mobile devices over PCs in their daily activities (chatting, shopping, email) (Trifonova, 2018). These devices are equipped with a touchscreen and have a simpler interface that does not require e.g. using a physical keyboard and learning shortcuts.

Twitter users were not in agreement on the best moment to switch from blocks to text. The most common answers were: towards the end of Key Stage 2 (primary education) or Key Stage 3 (secondary education). Some made this decision dependant on whether pupils have learnt the programming basics or have mastered programming using blocks. Students who attended Code Clubs were reported to be ready for the transition earlier than those who did not.

Among the things that could help pupils in the transition and improve their performance, participants mentioned: syntax checking, easy to understand error messages, block-to-text switch, list of commands and scaffolding. Some of these functionalities are available in some form in different educational tools, but according to the Twitter users they do not meet the needs of their pupils. This, along with the findings from the computing professionals' survey (use of IDEs in their everyday work and a direct recommendation to use IDEs for learning) suggests that there might be a need to develop a dedicated IDE for educational purposes. Earlier analysis of the most popular IDEs used in the industry

(Chapter 2) showed that they offer many functionalities that are useful in professional development, but might be overwhelming for beginners. An educational IDE could focus on the core functionalities mentioned by educators and provide a user-friendly interface specifically designed to support the learning process.

Finally, it is important to remember that, as many participants mentioned, classroom environments are not the same as work environments. Schools do not have enough time to refine pupils' skills and are forced to move on to the next topic, even if not everyone may be ready. Industry professionals have not only more time, but also more freedom when choosing the best way to solve a problem or implement a feature. Therefore, schools may require a slightly different approach than the industry.

4.4.1. Answers' Word Clouds

The 5 word-clouds below were generated from answers to each of the questions posted on #CASchat. Their shapes were chosen to be numbers for quick identification with each question.



Figure 4.31 - Word Cloud of Answers to Q1



Figure 4.32 - Word Cloud of Answers to Q2

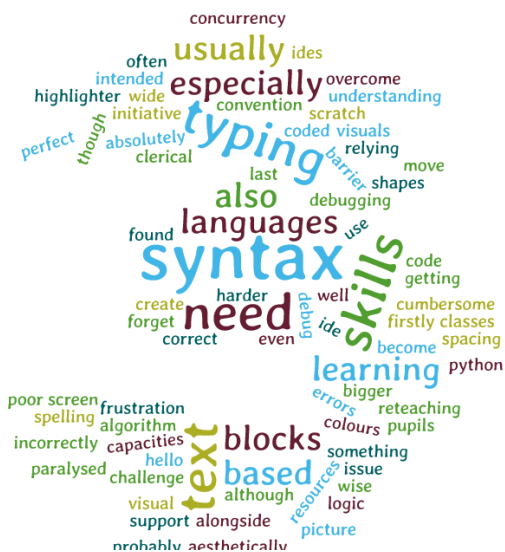


Figure 4.33 - Word Cloud of Answers to Q3

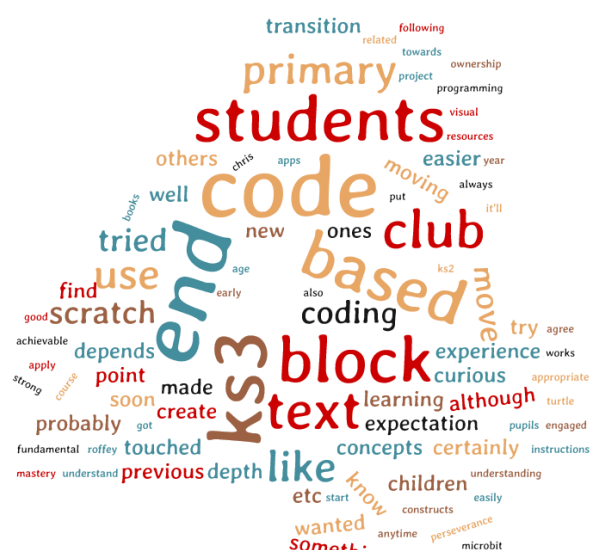


Figure 4.34 - Word Cloud of Answers to Q4



Figure 4.35 - Word Cloud of Answers to Q5

4.5. Teacher Interviews

The interviews were conducted between September 2018 and July 2019 with 7 computing teachers from secondary schools in Northamptonshire. In this section, they were referred to as “teachers” or “interviewees”. The charts below present an overview of their background:

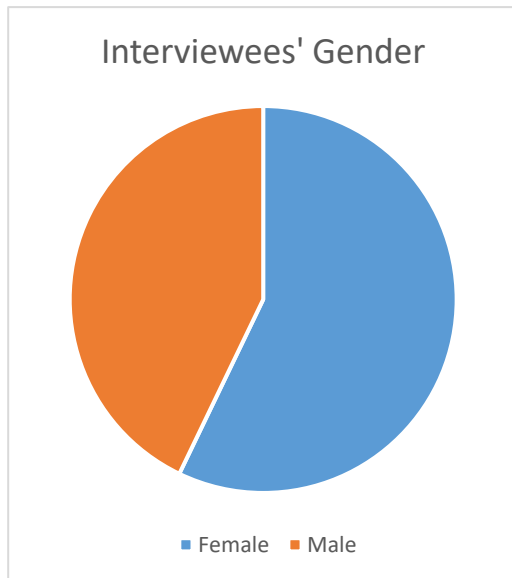


Figure 4.36 - Interviewees' Gender (Teacher Interviews)

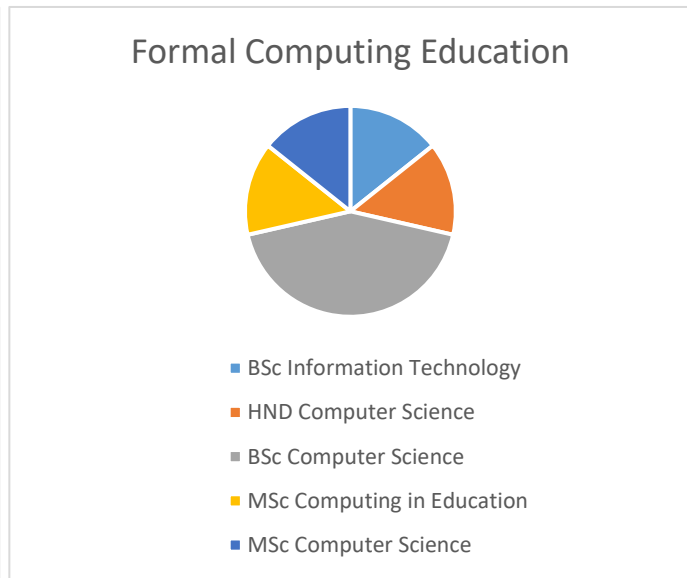


Figure 4.37 - Interviewees' Formal Education in Computing (Teacher Interviews)

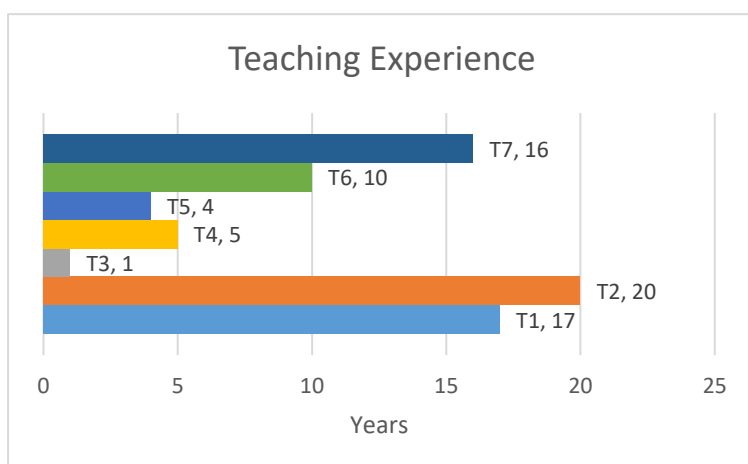


Figure 4.38 - Interviewees' Teaching Experience in Years, where T-Number Represent Each Teacher (Teacher Interviews)

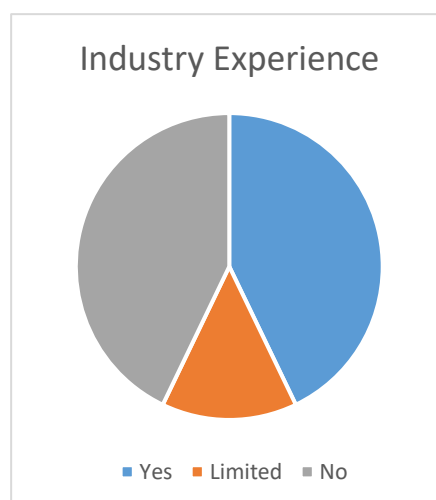


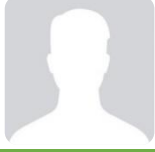
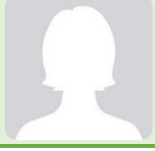
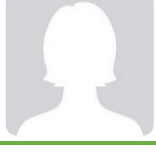


Figure 4.39 - Interviewees' Industry Experience (Teacher Interviews)

As can be seen, the interviewed teachers have different backgrounds in terms of their own education, teaching experience and industry experience, making them a well-composed sample. There is also a good gender representation.

The interviews were anonymous, hence for the purpose of analysis the participating teachers were given codenames from T1 to T7. The charts below present their short profiles for further reference:

	<p>T1</p> <p>Teaching experience: 17 years (13.5 – ICT and 3.5 Computer Science) Formal education: BSc in Information Technology Informal education: Programming courses Industry experience: no</p>
	<p>T2</p> <p>Teaching experience: 20 years Formal education: HND in Computer Science Informal education: no Industry experience: short period after university, setting up systems</p>
	<p>T3</p> <p>Teaching experience: 1 year Formal education: BSc in Computer Science Informal education: work in the industry and Microsoft qualifications Industry experience: 10 years, software developer</p>
	<p>T4</p> <p>Teaching experience: 5 years Formal education: BSc in Computer Science Informal education: CPD courses Industry experience: no</p>
	<p>T5</p> <p>Teaching experience: 4 years Formal education: BSc in Computer Science Informal education: CPD courses Industry experience: no</p>



	<p>Teaching experience: 10 years Formal education: MSc in Computing in Education T6 Informal education: Raspberry Pi Certified Educator, Microsoft Innovator Educator Expert, STEM learning Industry experience: 20 years, web consultancy</p>
	<p>Teaching experience: 16 years Formal education: MSc in Computer Science T7 Informal education: Self-education, keeping up to date with latest knowledge Industry experience: 1 year, software developer</p>

Table 4.4 - Interviewees' Profiles

4.5.1. Answer Analysis

The interviewed teachers were asked several questions related to different aspects of computing education, such as: challenges of their everyday work, the new curriculum, enhancement ideas and the relation between schools and the computing industry.

4.5.1.1. Do you think that your current knowledge levels of computing are adequate?

The responses varied. A confident 'yes' came from T3, who worked in the industry for 10 years and has moved on to teaching just recently. His opposite – T2, with 20 years of experience in teaching also believed his knowledge was adequate, but added that unfortunately this is not true for many teachers who teach computer science and find it very difficult.

T7 shared this opinion. Although, his computing knowledge was adequate, he believed that teachers without a computing background struggle with teaching this subject, as the content has become more difficult: what was taught at undergraduate level is now taught at A-Level. He also stressed that it is important for him to continuously improve his knowledge due to the recent reforms.

Interviewee T5 believed that her knowledge was accurate, but she too saw the need for constant improvement, as computing is always changing.

T1, with 13.5 years of experience in teaching ICT, admitted to having weaker and stronger areas. She was confident about teaching networking, data storage and algorithms, but lacked the skills to teach programming. She tried doing a programming course, but found it too short to learn it properly. Therefore, she decided to leave teaching computer science to her colleagues with industry background.

The only person who admitted that her knowledge was not good enough was T4, with only 5 years of teaching experience and a degree in computer science. After moving from OCR (Oxford, Cambridge and RSA Examinations) to IQA (Internal Quality Assurance of Assessment Processes and Practice) qualification she found that she needed to learn new topics.

4.5.1.2. What are the main problems that you face in your everyday job?

To this question every interviewee provided a different answer.

For T1 the biggest issue was the lack of time. She said that computing teachers are overworked (working up to 12 hours a day) and underpaid, which lowers their morale. She criticised the DFE (Department for Education) for not providing enough money to schools, making them unable to replace teachers who have left, and for delays in updating their lists of qualifications, which leaves little time for schools to prepare for any changes.

T2's answer coincides with what T7 said to the previous question. He too, believes that the new curriculum is extremely difficult, especially for years 10 and 11. The material is as advanced as it used

to be at university level. He also mentioned that computer science fails the initial expectations of students. When they begin their education, they believe that computing is about making and playing games, but then it turns out that this is something far more difficult.

T3 struggles with a lack of motivation among his students. While some of them can be passionate about computing, many are disengaged. He blames this on the fact that GCSE content is very dry in his opinion.

The biggest problem to T4 is the fact that some students struggle with applying their knowledge, for example solving a problem using an 'if' statement.

T5 struggles with finding links to the industry in order to give her students a work experience. She would like to find professionals who would talk about computer science to show the students what options are there.

According to T7, a problem lays in the attitude of school leaders, who do not understand how demanding computer science is and expect it to produce results equally good to ICT. In reality, this is not possible, because computer science is a far more difficult subject.

4.5.1.3. What is your opinion about the new computing curriculum? Is there anything you would like to change?

For T1, following the new curriculum means there is not enough time for more creative activities in the classroom that could attract the students, as teachers are tight on schedule to cover the whole material. She also criticised the new system for not giving the students enough room for choice with their GCSE subjects, which results in fewer students choosing computer science.

T2 and T3 agree that the computing curriculum is a bit too difficult. While T3, who has recently come from the industry, likes the fact that it has moved towards computer science, he believes it needs some polishing, because some material is too dry and some is covering too much. T2 has got a lot of teaching experience and thinks that the computing curriculum should be standardised with other subjects and the level of difficulty should be brought down.

Interviewees T4 and T5 were also critical about the curriculum. T4 has noticed that children these days are brought up using tablets and do not have some of the basic skills, such as using a mouse or saving files on a computer. Therefore, she sees the need for more ICT lessons. T5 complains about the fact that the practical project does not count towards the final grade.

T6 thinks it is important that businesses help to shape the syllabus to make sure it is more relevant to the 21st Century society, as the current one is a bit behind. She would like to see for example more robotics, data management and integrity, as well as an introduction to AI (artificial intelligence) and machine learning. She also believes that universities should work with secondary schools to ensure coherence with the syllabus and develop work-ready individuals.

The last teacher, T7, could not think of any areas for improvement. He only expressed his satisfaction over the fact that the government decided to bring computer science into the curriculum, as it allows to develop better skills that are transferrable to other areas.

4.5.1.4. Do you feel comfortable teaching your subject following the guidelines of the new curriculum?

T1 overall does feel comfortable, however, she upholds what she said about lack of time. There are things, such as programming and databases that she needs to learn herself, but unfortunately, she does not have enough time.

T2 feels less comfortable about the new curriculum than he used to be once. He shares the same thoughts on NEA (non-exam assessment). T5 finds it a nonsense that it does not count towards the final grade. Because of that, more effort is required to get students engaged and make them feel enthusiastic about the project.

T3 with his industry experience is familiar with the material he is required to teach, making him feel comfortable. However, he admits that some of his colleagues are a bit worried.

T5 used to teach IT and admits that the beginnings with the new curriculum were not easy, as she needed to learn a lot. However, after 3 years her confidence has improved a lot and now she helps other teachers in her area by running courses for them.

T7 does feel comfortable with the new curriculum himself, although he believes that there are not enough teachers who have no computing background. People like that tend to struggle with teaching the new material. What T7 says, matches the profile of T1 – an ICT teacher with no industry experience, who finds some of the curriculum content difficult to learn, despite her best efforts.

4.5.1.5. What teaching aids do you use (e.g. software to teach programming)?

The answers to this question varied. Every teacher enumerated a few tools or resources that they use in their classroom. The complete list contains 22 items but most of them (59%) were mentioned only by one teacher. Several were used by 2 different teachers (36%) and only one – Raspberry Pi – was mentioned by 3 teachers.

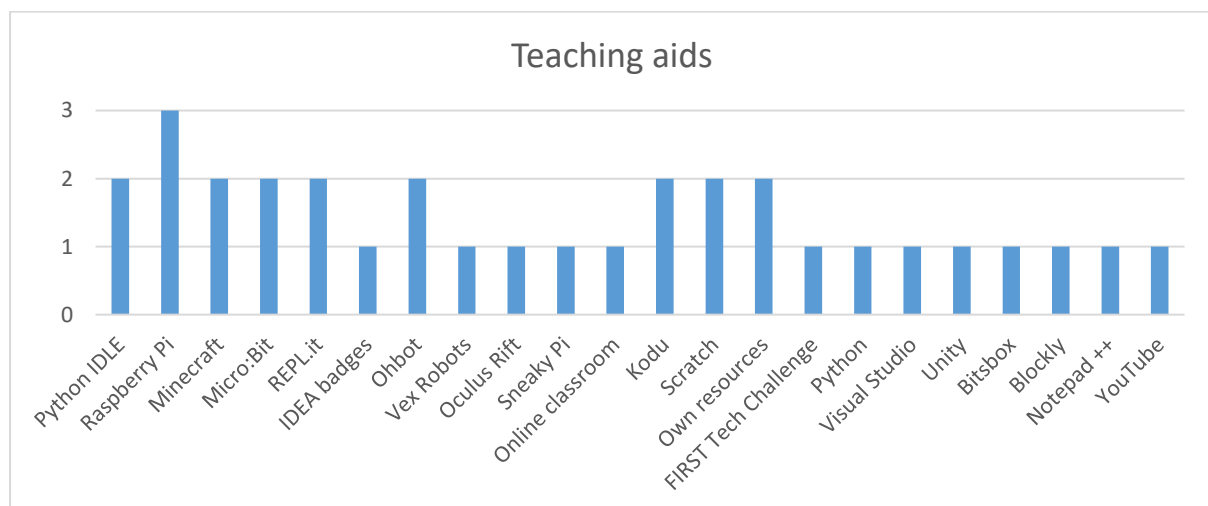


Figure 4.40 - A List of Teaching Aids Enumerated by Interviewees

The answers only partially match the results of the professionals' survey. The common element is Raspberry Pi, which was the second most recommended tool by participating software developers. None of the teachers mentioned Codecademy or Lego Mindstorms.

Interviewee T5 mentioned that robots were expensive and usually contained small pieces that can go missing, therefore not all schools can afford them. Her school was granted a bursary due to participating in FIRST Tech Challenge – a competition for pupils that involves programming robots.

Interviewee T3 used this opportunity to share his thoughts on the recent changes to NEA. As it no longer counts towards GCSE, there is a pressure to cover more theory than do practical exercises, which normally require using tools. Pupils are also less motivated to do practical tasks.

4.5.1.6. Do you use any particular methodology in your work?

Most of the interviewees do not use any particular methodology. T1 just covers the material, adjusting methods to different topics. Similarly, T4 stated that she just uses a mixture of everything: talking, practice and scenarios.

Two teachers with some industry background had more to say about their methodologies. T7 uses different methodologies depending on students he is teaching. He asks more able students to solve a problem (such as “generate a random number”). With a less able class, he uses more visual tools, such as blocks, flowcharts or asks them to write pseudo-code.

T3 is trying out a “flipped classroom” strategy: asking his students to do research at home and come back with a question that they can discuss in the classroom. The theory can then be used in practical exercises.

4.5.1.7. Would you agree that students find it difficult to switch from block-based to text-based programming? What do they find the most difficult?

Four out of seven interviewees did agree that students have problems with switching from blocks to text when learning programming. T2 tries to bridge the gap in several ways, such as doing the same programme using blocks and text, but his students find it extremely difficult.

According to T4, the problem is that programming with Python is not fun as opposed to Scratch or Kodu. T5 shares this opinion, saying that when moving to text programming, the fun element is lost and there is a lot of theory which puts students off. However, she has found a new programme that helps with the transition by using Scratch colours in Python and making it look more familiar to students.

T1 and T7 say that a lot depends on how the text-based programming is introduced. T7 believes it is important to show how blocks translate to code and rigorous syntax poses a problem to the students – a missed indentation or colon can make a difference.

T1 believes that introducing the students with simple programmes, such as “Hello World” and teacher’s positive attitude can make a difference and remove the fear of text programming. From her perspective, a more pressing problem is the lack of digital skills among the students, who are more used to tablets these days, than they are to a keyboard and a mouse. A very similar observation was shared by T4.

T3, having taught only a little over a year, has not done the transition yet. However, he believes that both blocks and text-based programming have their merits. Blocks help to remove the initial barrier which is syntax.

4.5.1.8. Do you think that an educational IDE for beginners could help to make the first steps in text-based programming?

All of the interviewees agreed that this would be a good idea. They believe that a user-friendly IDE with guides, meaningful feedback, code completion and colour-coding would be a great help to the students who very often struggle with syntax and logical errors. T4 believes that it could also make programming more entertaining.

T5 thinks that an IDE could make a teacher’s job easier, by reducing the number of students who need their help. Currently, she tries to overcome this problem with introducing pair-programming.

In T7’s opinion, IDEs are usually not very friendly, so he likes the idea of something designed for students. However, he believes that the development of such tool requires feedback from teachers.

T3, who is an experienced programmer, actually tries out different IDEs with GCSE and A-Level students. He showed some of his students Visual Studio and they were very excited about it. He also planned to install another professional IDE, PyCharm on the classroom machines. However, many teachers use more basic environments, such as Python IDLE, which are not very helpful. Simple text editors make writing code very difficult as they provide little to no feedback. Furthermore, they usually allow to develop console applications, which are not exciting to the students. Young people cannot link them with GUI (graphical user interface) applications which they are very familiar with.

4.5.1.9. How important do you consider the ability to solve problems by looking for possible solutions on the internet and exchanging ideas with others?

Six out of seven interviewed teachers agreed that using the internet should be a part of problem-solving. T3, T5 and T7 encourage their students to look for solutions on the internet, for example on GitHub. T1 and T4 believe that using the internet is important and criticise the NEA for not allowing it.

The only teacher who discourages his pupils from using the internet is T2. He believes that students should first try to solve a problem on their own, by breaking it down. His argument against the internet is that students could easily find a solution and use it without understanding.

Most of the teachers believe that collaboration in programming and problem-solving is very important, because, as said by T7, "Computing is not a solitary job". He thinks that the exam board needs to understand that and encourage working on a problem in groups. According to T6, exchanging ideas with others is the principle – whether it is done face to face or via the internet. It allows to make the students fully engaged.

T2 asks his students to discuss problems in pairs. T3 also considers this to be important as students are more likely to talk to each other than the teacher. He also believes that it makes learning more effective.

T1 uses rubber duck debugging as a way of encouraging pupils to verbalise their problems. She gives everyone a duck to talk with.

All teachers who consider using the internet and collaboration to be a key part of learning and problem-solving (T1, T3, T5 and T7), also agreed about one more thing: that this is how people from the industry work. In the real world, people talk to each other, re-use and adapt code that they found online. T6 adds that is important to address real-world problems in the classroom.

4.5.1.10. Do you know what methods computing professionals from the industry use to learn on everyday basis?

All interviewees with no industry background (T1, T2, T4 and T5) answered that they do not know. T4 and T5 added that they would like to learn more about it. T5 even puts posts on Twitter that she is looking for professionals who could come and talk to her students, but it is not easy to find someone.

T7, who had 1 year of industry experience believes he might be out of date; however, he considers this type of knowledge to be valuable. Just like T5, he thinks it would be nice to have computing professionals to come to schools and talk about how they work. Unfortunately, time is always the issue.

Prior to becoming a teacher, T6 had her own website consultancy, therefore she was able to answer this question. She stated that industry people do courses to learn about new technologies – there are a lot of them for different areas.

This question was an easy one for T3, who worked in the industry for 10 years. He mentioned several things: going to seminars and conferences, watching videos, using Twitter, subscribing to forums, talking with each other and pair programming. He believes that things like this should be done in schools, but unfortunately there is a lot of pressure to follow the curriculum and prepare for exams.

4.5.1.11. Do you think there is a value in practice of industry professionals that could be moved to schools?

Six out of seven interviewees agreed with the statement included in the question. T7 believes that it is important to show how different topics relate to the industry and get computer professionals to talk about their work. T4 would like to have links between businesses and schools, but she admits that it is hard to find the time for that. T5 would like to introduce children to different types of careers available in computing industry and learn what an average professional does. Perhaps she could apply their methods to her teaching.

T6 cautioned against putting too much pressure to try to deliver too much industry standards in schools. She thinks that while teachers should be mindful of different scenarios, they should not provide every one of them. T6, just like other interviewed teachers, sees the need for partnership between companies and schools in order to provide a pathway for students.

T3 believes that his industry experience and knowledge has helped him a lot. He would like the curriculum to be more relevant to what is needed in the industry and contain more programming.

Interviewee T2 strongly believes in the value coming from industry practice, based on his contact with a colleague who used to work in the industry. He appreciates his colleague's up to date knowledge.

Only T1 was hesitant and said she could not answer the question due to the fact that she does not know about the methods used in the industry. This is something that she would like to learn about. As for the content that is taught in schools – she believes it needs to be fairly easy and engaging for students.

4.5.2. Summary of Teacher Interviews

From interviews, emerges a picture of computing education in secondary schools, seen from the perspective of teachers. In their daily work, they struggle with many things:

- not enough time to learn and improve their own qualifications
- working long hours
- lack of motivation among the students
- making links with the computing industry
- covering the contents of the curriculum

Although, teachers see the introduction of the new computing curriculum as a step forward, most of them find it very difficult, demanding and leaving little room for creative practical projects that could engage students. They were especially critical about Ofqual's (Office of Qualifications and Examinations Regulation) decision to change the GCSE assessment process and not include NEA's mark. Because of that, pupils are less motivated to do it, which makes a teacher's job a lot harder.

There was a lack of agreement among them in terms of methodology and teaching aids that they use in the classroom, showing that there is currently no established way of teaching computer science. Teachers try out different solutions and products on their own, in an attempt to find an attractive and engaging method to introduce their students to computer science. It is worth noting, that the participants of the survey conducted by Sentence and Csizmadia (2015) were much more confident in

using different pedagogies and tools in their work. This suggests a need for further research to better understand the scale of the problems with delivering the new curriculum and develop strategies to support the teachers.

A challenging moment in computing education is when students need to move from programming in blocks to a textual language. Teachers say that text-based programming is “no fun” as opposed to blocks and that students struggle with syntax. An interesting fact was that nowadays young people, who were brought up using tablets, lack the skills to use a PC (e.g. use mouse and keyboard). This information confirms the findings from #CASchat analysis.

Teachers with no industry background admitted that they do not know how industry professionals work and learn. However, some of the interviewees said that they would be eager to host a professional in their classroom and find out more about them.

Interviewees were almost unanimous on three topics: an IDE for beginners, the role of internet and collaboration in problem solving, and the value of the industry. They agree that an IDE would be a great tool to support students in text-based programming, as it could help them to overcome problems with syntax and spelling, as well as provide meaningful feedback. Some said that it could also make programming more entertaining to students. Interestingly, these findings resonate with the insights from #CASchat analysis, where the participants agreed that features offered by IDEs would help their pupils to overcome the challenges of text-based programming. Similar thoughts were shared by Chris Roffey in his article for “Hello World” magazine (Roffey, 2019)

Teachers were very positive about collaboration on problem-solving and (with one exception – T2) searching for solutions on the internet. They believe that it is very important to encourage students to talk to each other and discuss about problems, as well as to use the internet. Four of the interviewees believe that this is how people in the computing industry work – they solve problems by collaborating and re-using code. This coincides with the earlier results of the survey conducted among industry professionals and shows that the social constructivism should be applied during computing lessons.

Finally, teachers do see a value in the practice of industry that could be transferred to schools. They would be eager to meet professionals, learn what methods they use and invite them to their classrooms. One of the teachers admitted that she actively looked for people from the industry that could introduce pupils to their work. Others agree that collaboration with businesses would bring value to computer science lessons. There were also voices that the industry should participate in shaping computing education in schools. It is worth noting, that one of the recommendations of The Royal Society’s 2017 report was that computing industry should continue to provide support for teachers. The interviews showed teachers’ agreement that more industry involvement could be beneficial for their professional development.

Chapter 5: Reflection, Conclusions and Recommendations

5.1. Conclusions

Following a comprehensive analysis of data gathered through various research activities, the time has come to review the initial aim and objectives, and reflect upon the implications of this project's findings. They will be discussed sequentially within this section.

5.1.1. Computing Education in Schools – Changes, Challenges and Teaching Methodologies

Computing began its career in British schools in the late 60s. As a subject that was very consistent with university standards. Unfortunately, with the increasing popularity of personal computers and growing demand for digital skills, this rigorous subject was gradually replaced with ICT (Information and Communication Technology).

After many years, time has come for changes. More and more people from the industry (e.g. Eric Schmidt, former CEO at Google) started to criticise computing education in British schools, saying that it did not offer the right skills. Following the publication of “Next Gen” (Hope and Livingstone, 2011) and “Shut down or Restart? The Way Forward for Computing in UK Schools.” As per the Royal Society (2012) report, the government announced the re-introduction of computer science to schools, with a more demanding curriculum, containing databases, programming (using blocks and textual languages), networking, security and computer systems (Department for Education, 2014).

This awaited change, however, brought a number of new challenges, most notably:

- many teachers were not prepared to teach computer science, especially those with no real computing background
- there was confusion over how to teach the new subject – after its long absence in schools there were no established pedagogies
- computing science was not a popular choice among secondary school students

A later report “After the Reboot: Computing Education in UK Schools” provided several recommendations aiming to improve computing education in schools (The Royal Society, 2017). One of them was to offer more support for teachers. The report suggested that the computing industry should play an active role in offering that support, recognising its importance to education. Another recommendation was to do more research in the field of computing education.

An exploration of available tools and methodologies showed that there were many of them, offering support in different areas, mainly problem-solving, block-based programming and converting from blocks to text. However, it was apparent that there were no education-specific tools to support text-based programming. This matches with an article published in “Hello World” magazine, which argued that there was a need for developing an IDE (integrated development environment) for teenage pupils (Roffey, 2019).

An insight into the world of computing industry confirmed that a great number of professional developers used advanced IDEs, such as Visual Studio (Stack Overflow, 2018). A review of the most popular ones showed that they offered many advanced features to support textual programming, yet only some of them seemed to be useful for beginners.

An interesting pattern was found by examining the stories behind the success of several famous computer programmers. They all started their journey as young people, who were *curious*. This, combined with finding real-life problems to solve, led them to learning computer science bit by bit and becoming well-known experts in this field.

A review of the existing literature demonstrates that the computing industry played an important role in the reintroduction of computer science into British schools, and its continued support for education is anticipated. These and other findings lead to a question that informed further research and helped to narrow down its scope:

- What aspects of professional practice in the computing industry could be valuable additions to secondary school computer science education?

Finding the answer to this question became the primary aim of this research. It is mainly relevant to secondary education, as during this stage pupils begin to engage with more demanding, industry-relevant material, such as text-based programming. As such, the focus of this thesis laid on computing education in secondary schools.

5.2. Further Research Findings

To address the research question, a number of methods were employed: surveys, interviews and data analysis. This section will detail the findings from each research activity, highlighting their contribution to the overall aim of identifying valuable aspects of industry practice for secondary computer science education.

5.2.1. Computing Tutors' View of Computer Science Education in Schools

The first survey was conducted among university tutors at an early stage of the research, just before the new curriculum was introduced to schools. Its original goal was to confirm whether the findings presented in "Next Gen" report were still true as well as collecting data that could be compared with the results of a similar survey in the future (Hope and Livingstone, 2011).

Analysis of the responses were consistent with the findings from Hope and Livingstone (2011) report and showed that computing tutors' view of computing education did not change. Their students, who had recently completed their school education, lacked the relevant skills in a number of computing areas. The tutors rated these to be "very limited" at best.

Their overall outlook on the quality of computing education in schools was negative, as they disagreed with all statements about different aspects of it. The statement that they especially strongly disagreed with was about preparing young people to work in the industry. This suggested that the content taught in schools before the new curriculum could not equip pupils with the knowledge and skills required by the industry.

Four years later, the survey was repeated to determine if tutors' opinion had changed following the introduction of the new curriculum. While the rating of students' computing skills was higher ("basic competence" to "very limited competence"), tutors' opinion about secondary computing education did not change a lot. They still believed that school education was insufficient and that teachers were not good enough to deliver the material. However, a positive change was observed in tutors' outlook on the schools' ability to prepare pupils for higher education or work in the industry. The level of agreement with these two statements was significantly higher than in 2014.

In both the surveys, the tutors were asked to rate a number of proposed ways to improve the quality of computing education in schools. While they were generally positive about the presented ideas,

there were some surprising differences between the 2014 and 2018 results. In the second survey, the tutors were visibly less convinced about the need for cooperation between schools and universities, as well as about hiring teachers with industry experience. These results are contrary to the expectations; the reasons behind this change of attitude are unclear. The author has made a hypothesis that with the new computing curriculum in place, the need for school-university interaction became less important. However, more research would be required to find out if this is the case.

Finally, analysis of the comments left by tutors, mainly in 2014, brought some interesting insights. A number of respondents referred to the industry and the role it should play in modern computing education. Computer science, as opposed to ICT, was seen as a suitable option for young people who would like to pursue a career in this field. Yet, it was said that information technology can be useful to a greater number of people, therefore it was important to keep the balance between the two. This statement was confirmed in the later research.

Another tutor believed that the industry should be “more directly involved with schools”. The same respondent explained that university-school partnerships are not overly a bad idea, but they could be a “drain on academic time”. This could be another explanation of the surprisingly lower rating of support ideas that involved cooperation with universities.

An interesting opinion regarding the role of industry was shared by another respondent. This tutor thought that the new curriculum was focusing too much on what was “mistakenly” seen as the “need of the industry”, for example software programmes and computer architectures. Instead, computing education should focus on delivering more timeless skills that will always be relevant to those who wish to start a career in this field.

In the last comment from the 2014 survey, a tutor said that “the biggest limitation is time and resources” – something that was also reported by many teachers and educators at the later stages of research. According to respondent’s experience, the teachers struggled with finding the time to develop new materials and schools – with buying equipment that would allow to deliver this material.

Four years later, a tutor left a comment saying that “change in content” (of the curriculum) was perhaps helpful, however the tutor could not observe any improvements as “not enough students are taking these courses”, which suggests problems with engaging pupils and making them interested in computing.

5.2.1.1. Industry professionals – their learning, problem-solving and advice to schools

Professional software developers who work in the industry were asked questions in three categories: their learning experience in the past, their current learning and problem-solving habits and learning recommendations.

Analysis of the collected quantitative data shows that in most cases professionals were driven by curiosity (they wanted to understand how their computers worked) and learnt mainly on their own. These findings are consistent with what was learnt from studying the stories of well-known computing professionals.

Despite the fact that close to $\frac{3}{4}$ of the respondents had computing lessons in schools, their most common source of computing education was university (72%), followed by self-education (51%). School was chosen only by a third of professionals and their rating of computing lessons was only 2.6 out of 5. These results are suggestive that computing education at school was not adequate to the majority of participating professionals.

Nowadays internet plays a key role in professionals' learning and problem-solving process (as stated by 90% or more respondents, depending on question), with their number one website being Stack Overflow (a question-and-answer portal). Professionals were also shown to help and learn from each other, either via tutorials (a form of sharing knowledge by more experienced programmers) or by asking their colleagues. This proves that social constructivism theory (Vygotsky, 1978) finds application in learning computer science.

Professionals' recommendations in terms of tools and programming languages suitable for beginners largely overlap with the tools/languages that they themselves used in the past. The most recommended programming languages were JavaScript and Java, tools – CodeCademy (online tutorials) and programmable devices (Raspberry Pi and Lego Mindstorms). Other popular recommendations include programming languages that are now commonly used in the industry – C# and Python (Stack Overflow, 2019).

Perhaps, the most important findings were made by analysing the comments left by computing professionals. The survey contained two open questions: advice to schools and advice to yourself from the past. Professionals believe that doing real-life projects is extremely important in computing education, as it helps to build interest in this discipline. They also recognise the significance of practical exercises and suggest doing more assignments rather than theoretical exams. Finally, they recommend using different support tools (such as Lego Mindstorms, Raspberry Pi or IDE), that can provide better learning experience, and resources – mainly the internet (e.g. using Stack Overflow, searching for technology-related information) and other people who already know how to programme.

5.2.1.2. Twitter #CASchat discussion about problems with computing education and possible lessons from the industry

The #CASchat discussion took place in September 2018 on Twitter. A group of computing educators, mainly based in the UK, debated on a number of selected topics concerning computing education in schools. The main conclusions coming from the analysis of their tweets are as follows:

One of the main differences between a professional and a classroom environment is the time. While industry professionals can refine their skills and knowledge in a circular manner, pupils have very little time to reflect on what they have learnt or use their new skills, as they constantly need to move on to a new topic. This confirms what one of the computing tutors said in their comment – that time is the issue.

There were two main lessons that schools could learn from the computing industry. First of all, educators believe that teamwork and collaboration play a very important role in problem-solving and that pupils should work on projects in teams. Secondly, educators recommend giving pupils real-life problems to work on, as this is something that can stimulate them. These findings are very compatible with the results of computing professionals' survey.

Educators agreed that pupils tend to struggle with switching from entertaining programming using blocks to more demanding text-based programming. They named several reasons behind this: syntax, less appealing look and feel, poor typing skills and cumbersome IDEs. The word 'syntax' was mentioned in a record number of tweets (11), which indicates the scale of the problem. Educators reported that strict syntax in text-based programming is causing frustration among pupils and paralyses them. This seems to be connected with young people's poor typing skills, which was a surprising discovery. It was mentioned that using a correct IDE could be a solution to these problems.

Educators' opinions were mixed as to when is the best moment to introduce pupils to text-based programming. Some had tried it with Year 6 (last year of primary school), while others recommended

Key Stage 3 (early secondary education). A number of others said that is depended on pupils' fluency in block-based programming or their knowledge of basic concepts.

Finally, there were several recommendations on how to improve text-based programming experience for beginners. Educators stated that a textual language needs some form of syntax highlighting and meaningful error messages – features that are present in IDEs. This relates to the problems that they mentioned earlier. Other suggestions included the ability to switch from blocks to text or some type of scaffolding, but the exploration of support tools showed that there was a number of them that offer similar functionality (e.g. EduBlocks or Greenfoot). One person made an interesting point: that very often pupils feel lost when moving to text-based programming, because, unlike in blocks, there is no list of available commands. Therefore, beginners could benefit from a feature that would show them a list of what they can do. Professional IDEs (e.g. Visual Studio or Eclipse) can show the user a list of available objects and methods and provide suggestions via code completion.

5.2.1.3. Secondary computing teachers' understanding of industry practice and potential value for schools

Interviews with a number of secondary computing teachers allowed to understand what problems they need to face at work. These were mainly: long working hours, not enough time for self-education, lack of motivation and engagement among pupils, demanding curriculum and making links with the computing industry.

While teachers were not overly negative about the new curriculum and some saw it as a step forward, they agreed that it was not perfect. They complained about the fact that its content is dry, too difficult and does not give them enough room to do practical activities that could engage pupils. However, what they criticised the most was Ofqual's decision to remove the NEA (non-exam assessment) from counting towards GCSE grade (Office of Qualifications and Examinations Regulation, 2018). Teachers believed that this reduced the motivation of pupils, making their work more difficult.

Earlier exploration of teaching methodologies and support tools showed a variety of options available to computing teachers and students. However, the interviews showed lack of established ways of teaching, which is consistent with the findings of the Royal Society's second report (The Royal Society, 2017). Most of the teachers admitted that they did not use any particular methodology in their classrooms. While they did name a few teaching aids that they used, their answers overlapped only to small degree.

The interviews confirmed the #CASchat findings in terms of transition from block- to text-based programming. Syntax and less entertaining look and feel were reported to be the main issues in this process. Some teachers reported that their pupils lack the skills to use a PC (keyboard and mouse), which is also consistent with #CASchat answers.

Teachers with no industry background were unable to say what learning and problem-solving methods are used by industry professionals. Some of them showed interest in learning more about it and stated that they would like a professional to visit their classrooms.

Most of the interviewees agreed about several key issues. First of all, that an IDE for beginners would be a great way of supporting pupils in text-based programming, by helping them to overcome the syntax barrier and providing meaningful feedback. Perhaps also making textual programming more attractive. Secondly, they understand the importance of teamwork and collaboration in solving problems, as well as the role of the internet in finding solutions. They believe that these should be encouraged in school environment, including exams, and their main argument was that this is how

professionals work in the real world. These findings are very consistent with previous research – they match what was learnt from computing professionals and other educators via #CASchat.

Lastly, teachers said that they could see values in the practice of computing industry that could be useful to schools. They would like both of these worlds to collaborate on improving computing education.

5.2.1.4. Experimental trials of Integrated Development Environment for school education

One of the directions that were investigated during this research was the design of an IDE tailored for young learners that could help them to overcome the difficulties of learning text-based programming. A detailed description of the “IDEal” software design, development process, and initial trials can be found in Appendix 10.

While the trials conducted with university students returned promising feedback on the overall concept and its potential to enhance the learning experience, they also revealed valuable insights into user preferences and potential challenges in designing educational software for beginners. Notably, students expressed enthusiasm for the non-standard features, suggesting a potential for innovative approaches in educational technology. However, further research and development are necessary to refine the IDE and address technical limitations before it can be fully realized for use in secondary school settings.

5.3. Implications

The results of this research clearly show that computing industry can be a source of values to school education in this field. These values were identified as:

- teamwork and collaboration
- practice and solving real-life problems
- using IDEs to write code and learning from internet resources

These methods and activities are seen as a highly important aspects of effective computing education by both tutors and industry professionals. They can increase motivation and engagement among pupils and help them to develop universal skills that are valuable regardless of career path they may choose in the future.

Collaboration with industry professionals is seen as desirable, allowing teachers and pupils to gain an insight into the real-world practice and learn valuable lessons from them that could improve the computing classroom experience.

Interviews with secondary school teachers and analysis of the #CASchat discussion among computing educators revealed support for the concept of an IDE designed specifically for beginners. Teachers acknowledged the potential of such an IDE to address the commonly identified challenge of syntax barriers in learning text-based programming. Additionally, the discussions highlighted the potential for integrating features that promote the key values identified in this research, potentially contributing to an improved learning experience for pupils.

5.4. Limitations

Although the activities that were carried out during the research brought a number of findings that allowed to answer the main question of the thesis, it should be noted that they were not free from limitations.

The limitations identified by the author include:

- Lack of a neutral option (“not sure/don’t know”) in the first survey conducted among university tutors, which was pointed out by respondents in the comments. This affected their answers, forcing some of them to give their “best guess”, but the mistake was corrected in the subsequent survey.
- Problems with finding and contacting a larger number of software developers with relevant background who could participate in the survey. The author aimed to collect responses from people who worked as software engineers/developers and preferably obtained their education in the UK. This was proven to be difficult, mainly due to the limitations of LinkedIn portal which was originally chosen for the purpose of finding and contacting the right candidates. This forced the author to deploy other methods, including a number of personal contacts.
- Problems with finding teachers who would like to participate in the interviews and experimental trials of the “IDEal” software (Appendix 10). As was found later during the interviews, teachers are overworked and struggle to find time even for self-development. This could be the reason why only a small number of them responded to the author’s email and expressed the will to share their experiences.

5.5. Recommendations

Based on the key findings that emerge from this study, the author has developed a list of recommendations for further research and development in certain areas that could contribute towards improving the quality of modern computing educations in British schools.

5.5.1. GCSE’s Practical Project

While the decision to modify the assessment process following the online task leak incident seems to be well justified and understandable, it should by all means not be permanent. This research shows how crucial practice is to promote problem-solving and improving pupil engagement in the classroom. As reported, Ofqual's decision negatively impacted pupil motivation to participate in what is now called the Practical Project.

Encouraging teamwork, collaboration, and responsible internet usage remains essential in classrooms and non-exam assessments. Identifying fair and effective methods to evaluate collaborative work and internet-based tasks undoubtedly presents a significant challenge. However, through more extensive research and experimentation it could be achievable, offering pupils a more realistic and engaging experience, as well as equipping them with useful skills.

5.5.2. Research Ways to Promote Teamwork and Collaboration in Classroom

The research identified teamwork and collaboration as the most valuable aspects of the computing industry that could be beneficial in schools. Further research is necessary to explore effective methods for promoting and fostering these values within a healthy school environment, equipping pupils with essential real-world skills. However, educators who participated in #CASchat emphasized the distinct nature of school and industry environments, highlighting the need for a tailored approach when introducing industry values into computing curricula.

5.5.3. Research How Solving Real-Live Problems Can Improve Student Engagement and Enhance Learning Process

Professional programmers emphasized the importance of incorporating real-world problem-solving into the curriculum. However more research would be required to understand how exactly it affects pupils’ engagement and motivation when learning programming at school. For example, monitoring

and comparing the performance of two groups of pupils while solving different types of problems is seen as a possible research option that could bring more knowledge about this issue.

5.5.4. Develop IDE Dedicated to Use in Classrooms

While there is a wide choice of integrated development environments (IDEs) for professional use, offering a number of advanced features, the support for text-based programming at a beginner's level seems very limited. Young people are reported to struggle with syntax and would benefit from a dedicated IDE. The preliminary feedback from computing students who tested the "IDEa" development environment was overall positive, suggesting its potential to significantly aid pupils in learning text-based programming (Appendix 10).

An IDE for beginners should definitely offer standard tools (such as syntax highlighting and error information) to support text-based programming and help to remove the syntax barrier. However, it could be extended with features that enable collaboration and learning from each other in the classroom. Development of a tool dedicated for use during computing lessons is seen as desirable.

5.5.5. Establish Closer Cooperation with the Industry

Computing teachers not only see a value in the practice of the industry, but also seek for ways to connect with computing professionals. Inviting one to their classroom would be beneficial to both pupils and teachers, giving them a better insight into the practice of industry and allowing to transfer values.

It is therefore recommended to find ways of improving the cooperation between schools and computing industry, that would allow teachers to easily establish contact with professionals and learn from them. This could be for example starting a dedicated intermediary scheme that would help to establish communication between both parties.

5.6. Concluding Summation

Conducting this research proved to be a demanding task on many levels: technical, organizational and time. Despite a number of difficulties and limitations, the author was able to find answer to the research question and provide several recommendations for further research and development that could lead to improving the quality of computing education and increasing its popularity among pupils. The following points are considered to be the most significant achievements of this project:

- increasing the awareness of methods used by computing industry professionals to learn and solve problems,
- identifying the need to develop a dedicated IDE (integrated development environment) for beginners,
- understanding the attitude that computing teachers present towards the industry and whether they see a value in the practice of professionals,
- identifying a set of values coming from the practice of the industry that could be applied in school education.

The author hopes that the findings of this research will be of use to computing teachers and educators, who work hard to make computer science a more enjoyable and interesting school subject. Expectantly, the conclusions of this thesis will also become a starting point for other researchers and consequently lead towards enhanced computing education in schools.

6. Appendices

Appendix 1 – The Professional Practitioners View of Computing Education in Secondary Schools (2014) – email to tutors

Dear [First name Last name],

I am a PhD student at the University of Northampton and I am doing a research about the quality of computing education in secondary schools in the UK.

As a part of the research I am conducting a survey among different university tutors in order to learn their opinion about computing education in secondary schools and the school leavers' level of computing knowledge.

I believe that your opinion as a computing tutor is valuable and I would be very pleased if you participated in the survey.

The survey should take approximately 5 min to complete and is available here.

If you cannot click the link above, please copy and paste the following into your internet browser to start the survey right away:

<http://www.eSurveysPro.com/Survey.aspx?id=1e981421-efc3-4f47-b47b-7d0ef74b6d80>

I do hope you will take part in this important research and look forward to hearing your views.

Kind regards

Aleksandra Dziubek

Appendix 2 - The Professional Practitioners View of Computing Education in Secondary Schools (2014) – questionnaire

1. About the survey

Thank you for agreeing to participate in this survey, which I am conducting as a part of my PhD research. Your opinion is very important to me as it will help me to determine the quality of computing education in British schools and find which aspects of it need improvement. I believe that a better secondary education in computing could encourage young people to study it further and make them better prepared for continuing their education at a Higher Education Institute, which is surely important to you as a professional educator.

This survey should only take about 5 minutes of your time. Your answers will be completely anonymous, unless you decide to leave me your name (optional).

Thank you

Aleksandra Dziubek

2. About you

Please provide the name of the institution that you represent and the computing field that you specialise in.

1. Organisation: *

2. Field of computing: *

3. Higher Education Entry Level Computing skills

In an attempt to build up a picture of the Computing skills that students entering Higher Education are currently equipped with, please indicate your view of the computing skill-set of an “average” computing student, at the point of enrolment, on any computing-based programme of study at your intuition.

3. Computing skills: *

	No Expertise	Very Limited Expertise	Basic Competence in area	Considerable Competence in area
Internet Technology (Basic Static Web-pages, HTML, CSS)	○	○	○	○
Internet Programming Skills (Dynamic Web-page construction using JavaScript, PHP, etc.)	○	○	○	○
Requirement Engineering Strategies Approaches to Software Analysis and Design	○	○	○	○
Programming Skills (in one or more programming language)	○	○	○	○

Database Design and Implementation Skills	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
Fundamental Networking Concepts (LANs, Topologies, etc.).	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
Fundamental Computer Systems Architecture Concepts (Memory, Hardware, etc.)	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>

4. Quality of Computing Education within Schools.

As a professional educator within the field of computing, an understanding of your current perceptions of the quality of computing education within schools is considered to be extremely valuable. Please indicate your level of agreement with the following statements.

4. Statement: *

	Strongly Disagree	Disagree	Agree	Strongly Agree
I believe that the current quality of computing-based education within schools is sufficient.	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
I believe the computing content taught in schools is adequate.	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
I believe that the "average" student starting a computing-based course at my intuition has a good level of existing computing knowledge.	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
I believe that the quality of computing teachers within schools is of an acceptable standard.	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
I believe the current computing curriculum within schools meets the need of the students entering higher education.	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
I believe the current computing curriculum within schools meets the need of students entering industry directly, without studying for a degree.	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
I believe the technical resources used to teach computing within schools are current adequate.	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>

5. Future Improvements to Computing Education within Schools.

As a part the research I will be looking for ways to improve the way computing is taught in schools and enrich the pupils' experience. Some provisional strategies to enhance the computing provision within schools have already been formulated. Please indicate your level of support for the following strategies:

5. Statements:

	Strongly Disagree	Disagree	Agree	Strongly Agree

Provide additional computing training for teachers currently teaching IT.	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
Seek to employ teachers with existing work experience in computing.	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
Invest in better computing facilities and resources within schools.	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
Change the existing school's curriculum to focus on computing rather than IT.	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
Introduce programming concepts to pupils in primary schools.	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
Provide financial incentives to attract competent individuals to teach computing in schools.	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
Encourage A-level students to visit University Computing departments on STEM based projects prior to applying for computing places via UCAS.	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
Develop partnerships between University Computing departments and local schools that encourage awareness of computing and associated career pathways.	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>

6. Please indicate if you have any additional suggested strategies to enhance the computing provision within schools:

6. Concluding Comments

Thank you for taking the time to complete this survey. I value your time and the responses you have provided. If you feel you have any final comments that would be helpful to this research, please use the space provided below.

7. Comments:

Appendix 3 – The Professional Practitioners View of Computing Education in Secondary Schools (2018) – email to tutors

Dear [First name Last name]

I am a PhD student at the University of Northampton and I am doing a research about the quality of computing education in secondary schools in the UK.

In 2014 I conducted a survey among university tutors in Computing departments in order to learn their opinion about computing education in secondary schools and the school leavers' level of computing knowledge.

A lot has changed in computing education during the past 4 years, therefore I decided to do a follow-up survey, as I thought that it would be interesting to find whether the tutors can see any improvements, based on their interactions with new students who have just finished school.

I believe that your opinion as a computing tutor is extremely valuable and I would be very pleased if you participated in the survey.

The survey is completely anonymous and it should take approximately 5 min to complete. Please use the following link to start the survey right away:

<https://www.eSurveysPro.com/Survey.aspx?id=54c24e91-0f59-4c1a-abc9-f15b6cc016f6>

Identifying personal data will not be included in the survey therefore by submitting a response you are agreeing to take part in the survey.

I look forward to hearing your views. May I thank you for your time in advance.

Kind regards

Aleksandra Dziubek

Appendix 4 - The Professional Practitioners View of Computing Education in Secondary Schools (2018) – questionnaire

1. About the survey

Thank you for agreeing to participate in this survey, which I am conducting as a part of my PhD research. Your opinion is very important to me as it will help me to determine whether the quality of computing education in British schools has improved during the last few years. I believe that a better secondary education in computing could encourage more young people to study it further and make them better prepared for continuing their education at a Higher Education Institute, which is surely important to you as a professional educator.

This survey should only take about 5 minutes of your time. Your answers will be completely anonymous.

Thank you

Aleksandra Dziubek

2. Higher Education Entry Level Computing skills

In an attempt to build up a picture of the Computing skills that students entering Higher Education are currently equipped with, please indicate your view of the computing skill-set of an “average” computing student, at the point of enrolment, on any computing-based programme of study at your intuition.

1. Computing skills: *

	No Expertise	Very Limited Expertise	Basic Competence in area	Considerable Competence in area	Don't know
Internet Technology (Basic Static Web-pages, HTML, CSS)	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
Internet Programming Skills (Dynamic Web-page construction using JavaScript, PHP, etc.)	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
Requirement Engineering Strategies Approaches to Software Analysis and Design	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
Programming Skills (in one or more programming language)	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
Database Design and Implementation Skills	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
Fundamental Networking Concepts (LANs, Topologies, etc.).	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
Fundamental Computer Systems Architecture Concepts (Memory, Hardware, etc.)	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
Security Fundamentals	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>

Computing Relevant Mathematical Skills	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
--	-----------------------	-----------------------	-----------------------	-----------------------	-----------------------

3. Quality of Computing Education within Schools.

As a professional educator within the field of computing, an understanding of your current perceptions of the quality of computing education within schools is considered to be extremely valuable. Please indicate your level of agreement with the following statements.

2. Statement: *

	Strongly Disagree	Disagree	Agree	Strongly Agree	Don't know
I believe that the current quality of computing-based education within schools is sufficient.	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
I believe the computing content taught in schools is adequate.	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
I believe that the "average" student starting a computing-based course at my intuition has a good level of existing computing knowledge.	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
I believe that the quality of computing teachers within schools is of an acceptable standard.	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
I believe the new computing curriculum within schools meets the need of the students entering higher education.	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
I believe the new computing curriculum within schools meets the need of students entering industry directly, without studying for a degree.	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
I believe the technical resources used to teach computing within schools are current adequate.	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>

4. Future Improvements to Computing Education within Schools.

As a part the research I am looking for ways to improve the way computing is taught in schools and enrich the pupils' experience. Some provisional strategies to enhance the computing provision within schools have already been formulated. Please indicate your level of support for the following strategies:

3. Statements:

	Strongly Disagree	Disagree	Agree	Strongly Agree	Don't know
Provide additional training for computing teachers.	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
Seek to employ teachers with existing work experience in computing.	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
Improve the current teaching methods.	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
Invest in better computing facilities and resources within schools.	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>

Provide financial incentives to attract competent individuals to teach computing in schools.	0	0	0	0	0
Encourage A-level students to visit University Computing departments on STEM based projects prior to applying for computing places via UCAS.	0	0	0	0	0
Develop partnerships between University Computing departments and local schools that encourage awareness of computing and associated career pathways.	0	0	0	0	0

4. Please indicate if you have any additional suggested strategies to enhance the computing provision within schools:

6. About you

Please provide the name of the institution that you represent and the computing field that you specialise in.

5. Field of computing: *

7. Concluding Comments

Thank you for taking the time to complete this survey. I value your time and the responses you have provided. If you feel you have any final comments that would be helpful to this research, please use the space provided below.

6. Comments:

Appendix 5 – Professionals' opinion about computing education – initial message

Hi [First name],

I am a software engineer and PhD student who is trying to find better ways of teaching computing in schools. I believe that we, the computing professionals, can make a positive contribution towards improving computing education. Therefore, I created a questionnaire to find out how people managed to learn computing in the first place, what methods they use now to keep themselves up to date and what would they recommend/advise to teachers and pupils.

I would be very grateful if you could help me by filling out this quick questionnaire. It is anonymous and should take you about 5 mins to fill in.

The questionnaire is available here: <https://www.eSurveysPro.com/Survey.aspx?id=711e14a1-6e01-4c2e-bb63-a297e3a7dd93>

Identifying personal data will not be included in the survey so by submitting a response you are agreeing to take part in the survey.

Thank you for your help.

Alex

Appendix 6 – Professionals' opinion about computing education – questionnaire

1. Introduction

Thank you for agreeing to participate in this online survey, conducted as a part of my PhD research about computing education in Great Britain. I am looking for better ways to teach young people computer science and encourage them to study this field.

You can help me to achieve that by sharing your 'story' and experience in this survey. Answering the following questions will allow me to understand how you learnt computing and how you continue to learn new things every day. You will also be able to provide some recommendations. The results will hopefully allow me to develop a new, more effective framework to teach computing.

2. Your beginnings

Please let me know how your journey with computing began.

1. How did you become interested in computing?

- I wanted to understand how my computer works
- I wanted to create a game
- I wanted to solve a problem
- Someone encouraged me
- I wanted to create my own website
- Other (Please Specify)

2. Was there someone who introduced you to computing?

- Teacher
- Parent
- Sibling
- Friend
- No one
- Other (Please Specify)

3. How did you learn computing?

- In school
- At university
- Someone else taught me
- On my own - from books
- On my own - from the internet
- Other (Please Specify)

4. What was the first programming language that you learnt?
- JavaScript
 - PHP
 - Java
 - C#
 - C++
 - C
 - Python
 - Pearl
 - Ruby
 - Other (Please Specify)
5. Did you have any computing lessons in school?
- Yes
 - No
6. If you did have computing lessons in school, please rate your experience using the scale below, where 1 is "poor" and 5 is "excellent":
- 1
 - 2
 - 3
 - 4
 - 5
7. Did you use any of the following educational tools/platforms to learn programming?
- Scratch
 - Logo (turtle)
 - Greenfoot
 - Lego Mindstorms
 - CodePilot
 - Codecademy
 - Raspberry Pi
 - Code.org
 - Alice
 - Kodu
 - Python Tutor
 - Other (Please Specify)
8. If you did use tools/platforms, how helpful they were to you? Please rate your experience using the following scale, where 1 is "not helpful at all", and 5 is "extremely helpful":
- 1
 - 2
 - 3
 - 4
 - 5

3. Learning as a professional

Please let me know how you, as a professional programmer, continue to learn.

9. How do you expand your knowledge?

- I do extra courses
- I read books
- I take on new, challenging projects
- I search on the internet
- I ask questions on forums
- I watch video tutorials
- I learn from my colleagues
- Other (Please Specify)

10. Which websites (if any) do you use to learn/solve problems?

- Stackoverflow
- Dot Net Pearls
- W3Schools
- Other (Please Specify)

11. How do you support yourself in everyday programming and problem-solving?

- I use intelligent code completion
- I search on the internet
- I use my own notes
- I re-use my existing code
- I ask colleagues
- I ask people on forums
- I look-up answers in books
- I read/watch tutorials
- Other (Please Specify)

4. Your recommendations

Please provide a few recommendations to educators and pupils, based on your personal experience.

12. Which languages would you recommend to start learning programming?

- JavaScript
- PHP
- Java
- C#

- C++
- C
- Python
- Pearl
- Ruby
- Other (Please Specify)

13. Would you recommend any of the following educational tools/platforms to learn programming?

- Scratch
- Logo (turtle)
- Greenfoot
- Lego Mindstorms
- CodePilot
- Codecademy
- Raspberry Pi
- Code.org
- Alice
- Kodu
- Python Tutor
- Micro:bit
- Snap!
- Blockly
- Other (Please Specify)

14. Imagine that you can send messages to the past. What practical tip would you send to your younger self to make the learning process easier?

15. Do you have any other recommendations/ thoughts on how to improve the computing education in schools (e.g. what to begin with or what tools to use)?

5. About you

Finally, please provide a few more details about yourself.

16. Gender:

- Male
- Female
- Prefer not to say

17. Age:

- 20-25
- 25-30
- 30-35
- over 35

18. How many years of experience in computing industry do you have?

- 1-3
- 4-7
- 7-10
- over 10

19. Which is your 'primary' programming language now?

- JavaScript
- PHP
- Java
- C#
- C++
- C
- Python
- Pearl
- Ruby
- Other (Please Specify)

20. Thank you very much for your time and your contribution. If you have any other comments please type them in the box below.

Appendix 7 – Computing Teacher Interviews – questions

1. How long have you been teaching computing?
2. Do you have any formal computing education?
3. Have you received any informal computing training/education?
4. Do you think that your current knowledge levels of computing are adequate?
5. Do you have any computing work experience in industry?
6. What are the main problems that you face in your everyday job?
7. What is your opinion about the new computing curriculum? Is there anything you would like to change?
8. Do you feel comfortable teaching your subject following the guidelines of the new curriculum?
9. What teaching aids do you use (e.g. software to teach programming)?
10. Do you use any particular methodology in your work?
11. Would you agree that students find it difficult to switch from block-based to text-based programming? What do they find the most difficult?
12. Do you think that an educational IDE for beginners could help to make the first steps in text-based programming?
13. How important do you consider the ability to solve problems by looking for possible solutions on the internet and exchanging ideas with others?
14. Do you know what methods computing professionals from the industry use to learn on everyday basis?
15. Do you think there is a value in practice of industry professionals that could be moved to schools?

Appendix 8 – Computing Teacher Interviews – initial email

Dear Ms/Mr [Last name],

I am a PhD student at the University of Northampton and my research focuses on computing education.

I am writing to you because I am looking for secondary computing teachers who could share their opinions and experience during a short interview (max. 30 mins).

The interview could be done over the phone or in person, on a date and time that suits you.

Please let me know if you would be interested in participating.

Kind regards

Alex

Appendix 9 – Computing Teacher Interviews – consent form

Teacher Interview Consent Form

I, the undersigned, agree to be interviewed by Ms. Aleksandra Dziubek as a part of her PhD research on computing education.

I have been briefed on the following research information:

- The aims and objectives of the research.
- The potential consequences of the research.
- The likely publication formats of the findings of the research.
- My right to withdraw from the trials within 2 weeks from the interview.
- All information collected during the interview process is confidential and will be anonymised.
- I will be given access to the provisional results and final research outcomes at the end of the project.
- Any publications that results from this research will be made available to me.

.....

(Teacher's signature)

.....

(Date)

Appendix 10 – Integrated Development Environment for Students

1.1. An Integrated Development Environment (IDE) for Beginners

Computer-based environments have made it rather easy in the past few years to collect learning process data. Due to this, there is an enhanced interest in the area of analytics learning which is the key for leveraging the use of learning process data as a tool to improving teaching and learning of computing education. Within the context of computing education, it is only logical to collect learning process data by use of integrated development environments (IDEs) via computing students spending a lot of time practising creating programming assignments. Although, primarily, IDEs are used to provide support in computer programming, they are also quite efficient as a mechanism to deliver learning interventions designed for improving learning of students. IDE-based learning analytics provide an opportunity to creating learning processes in computing education using IDEs and collecting learning process data (Hundhausen et al., 2017).

The following chapter provides an outline of the specification, analysis and design that were created during the development of the Integrated Software Development for beginners.

1.1.1. Rationale

The analysis of teacher interviews and #CASchat discussion from Chapter 4 show that that teachers struggle with following the guidelines of the new curriculum, as the content is difficult not only for students, but also for many teachers, especially if they used to teach ICT before and do not have any industry experience. And since the practical project no longer counts towards GCSE grades, students are less motivated to do it.

A challenging moment in computing education is when students need to move from coding in blocks to learning a text-based programming language. Teachers reported that their students find text programming less entertaining and that they struggle a lot with demanding syntax.

The results of Stack Overflow's surveys show that professional developers use IDEs in their work – the most popular being Visual Studio Code, chosen by 50% of respondents (Stack Overflow, 2019). As presented in the literature review, IDEs provide a lot of support for programmers, especially with syntax and error detection. The most popular ones also provide code completion and are suitable for programming in many languages. These features make IDE a potentially great solution for beginners, which is why this format was ultimately chosen for testing.

There are, however, more reasons. While doing the literature review the author came across an article in "Hello World" magazine, which discussed the need for an IDE dedicated to teenage pupils (Roffey, 2019). The subject of IDE also appeared during other research activities: industry professionals advised their past-selves to use IDE instead of a text editor, #CASchat participants mentioned that it could solve the issue with syntax and interviewed teachers were very enthusiastic about the idea of an IDE for beginners. One of the teachers with relevant industry experience introduced his students to Visual Studio, which they found very exciting.

The examples above justify the choice of IDE as a tool for testing and evaluating different concepts and ideas that could improve the quality of computing education. These ideas were developed based on other findings of this research. Industry professionals, educators from #CASchat and computing teachers all agreed that collaboration is very important in programming and problem solving. Professionals learn from their colleagues and other developers via the internet. It proves that teachers and educators are right in thinking that this is how real-life professional environments work. People from all researched groups believe that collaboration – face to face or online should be encouraged

when solving problems in classrooms. This is also in line with the work of Guo and Warner (2017), who developed CodePilot – a collaborative tool designed to help novice programmers.

These findings lead to the design and development of a prototype IDE for beginners that would provide different ways of support and encourage collaboration.

1.1.2. Project Aims and Objectives

The aim of this development project is to create and evaluate a prototype of a new integrated development environment which would combine several features that, based on research findings, could improve teaching and learning experience in secondary schools, when students move to textual programming.

The author has identified the key objectives that led to completing this project:

- Design GUI (graphical user interface) of the software
- Develop a formal specification of the software
- Collaborate with software engineer on the implementation by answering questions and discussing possible options
- Test and debug the software with the help of software engineer
- Find a suitable method to distribute the software
- Conduct trials with computing students
- Conduct trials with computing teachers
- Evaluate the software based on feedback collected during the trials, resulting in a list of recommendations

1.1.3. Project Assumptions and Limitations

The assumption was to collaborate with a professional software engineer to develop the prototype of an IDE for beginners. The software engineer should be able to dedicate a portion of their free time to complete the project within a given period and provide support during the trials in case any problems are reported by participants.

The IDE should be possible to develop using the free version of software development tools and run on any Windows machine.

The main limitations of this project were time and human resources. First of all, the software needed to be completed within a reasonable time to allow testing and evaluation. While the specification and design had to be done by the author, it was not possible to do the implementation as well within the given time. Therefore, it was needed to find a software developer who would be willing to take on this project in their spare time.

Having limited resources meant that the end product would be only a prototype with basic functionality and a small amount of pre-entered data available to the user, such as code error information and wiki articles. The software also cannot be guaranteed to be free from bugs.

While Python seems to be a popular programming language in schools (it was mentioned during #CASchat and teacher interviews) and Stack Overflow surveys show that it is becoming increasingly popular among developers, neither the author nor the developer hired for this project know this language. As a result, it could not be used in this project. Further details are presented in System Analysis and Design section.

1.2. Requirement Specifications

This section will cover the detailed specification of all core components of the Integrated Development Environment. The specification has been created based on the issues that were identified during the research and recommendations from industry professionals and secondary computing teachers.

Functionality	Technical Notes
Code editor	The core functionality is a simple text-based code editor. Positioned in the centre of the main window, it should take the largest amount of space and contain a white typing space with line numbers displayed on the left.
Syntax highlighting	It is a widely used feature of all professional programming tools, as it makes code more readable and easier to understand. Keywords are automatically detected and displayed using different colour, making them visually stand out.
Code completion	As the user starts typing, a dropdown with suggestions appears within the editor space. This helps the user not only to quickly complete the code, but also learn about other available options.
Compiler	Programmes written in Java require to be build and compiled in order to run. A compiler puts the code together and allows it to run if there are no errors. The output should be displayed in the horizontal panel below the editor.
Error detection	<p>Errors should be immediately detected as the code is typed. They should be clearly indicated to show the user which line of code needs to be corrected.</p> <p>The details of compilation errors should be displayed inside a horizontal panel below the editor. This should include: the total number of errors, followed by more detailed information about each error (line of code where the mistake was made, type of error and a simple explanation).</p>
Creating, saving and editing a project	The users should be able to create a project in Java. The project contents should be displayed in form of a file directory tree in the vertical Project panel on the left side of the editor.
Adding, editing and removing files	Within every project the users should be able to create many files containing parts of their code.
Code wiki	<p>A built-in wiki, containing the crucial knowledge for beginning programmers, e.g. data types with their description, application and simple examples.</p> <p>The information should be accessible in two ways. First one is a search box where the users can type a keyword, e.g. "String" and click search. The second way is by clicking on a keyword in their code – e.g. clicking on "int" would open an article about the integer data type.</p> <p>The contents of the wiki should be displayed in a vertical panel on the right side of the editor, with the search box sitting above it.</p>
Duck debugging	A simple implementation of rubber-duck debugging. It should ask the user to explain what their programme is supposed to do and how their code is constructed.
Sending emails with code	The users should be able to quickly email their code to a friend or a pre-defined teacher, allowing to ask them for help.

Google search	A simple feature made of search text box and a button. After typing their query and clicking Enter/search button the user should be taken to Google search results page in a web browser.
----------------------	---

Table 5.1. - System Functionality and Features

As can be read from the table, the software should contain not only standard IDE features that support coding (syntax highlighter, code completion and error detection), but also ‘social features’, inspired by the practice of computing industry, that encourage collaboration and searching for solution.

1.3. System Analysis and Design

1.3.1. Software Type

The software was chosen to be a desktop application and that was for several reasons. First of all, most professional development tools are desktop applications, so this was a natural choice when developing a similar software for students. Moreover, building a desktop application seems easier from the implementation point of view, as it means less time spent on designing a GUI (graphical user interface) and using just one language for everything. Web application would require good knowledge and skills in HTML, CSS, JavaScript and perhaps another programming language. Not to mention dealing with browser compatibility issues.

The downsides of a desktop application are that it requires installation and may not be available on all platforms. However, knowing that Windows remains the most common operating system on desktop computers and that the target test users of this software are computing teachers, means that these obstacles should not be a serious problem.

1.3.2. Supported Programming Languages

An important question when designing the software was: which programming language should it support? There is a huge variety of languages, with some being more popular and commonly used than others. However, there were two main candidates: Java and Python.

Python has got a fairly simple syntax and it became very popular in schools via Scratch and various “block to text” conversion tools. As could be learnt from teacher interviews, it is commonly used in British schools. Outside of education, Python has noted a visible growth in popularity in the recent years (Stack Overflow, 2018; Stack Overflow, 2019).

Java, on the other hand, is more popular within the industry. According to a blog post by Jay Patel “The 9 Most In-Demand Programming Languages of 2017” it is one of the top languages, right behind SQL (Patel, 2017). It was also high in the rankings of the Stack Overflow 2018 Developer Survey (Stack Overflow, 2018). Aside from its popularity, Java shares a lot of similarities with other object-oriented languages (such as C#), which makes the Java-programming skills more versatile.

Both languages had strong pros and cons, however the final choice was Java. The reason behind that was a higher level of familiarity with this language – both the author and developers involved in implementation had some experience with Java, but no experience with Python. This was a key factor, because the software needed to go through initial testing process, before being released to the teachers.

The languages are very similar to each other and offer similar options to the users. The choice which language to teach will be left to the teacher. However, having a choice is important as it allows to show the pupils that the basic principles of most programming languages are very similar and learning one programming language will make it easier to pick up another.

1.3.3. Interface Design

The interface design was made using Balsamiq Mockups software and it represents how each crucial part of the IDE should look like.

Welcome!

Welcome!

Let's get to know each other first!

Name:

Email address:

Figure 5.1 - IDE Student Welcome Window

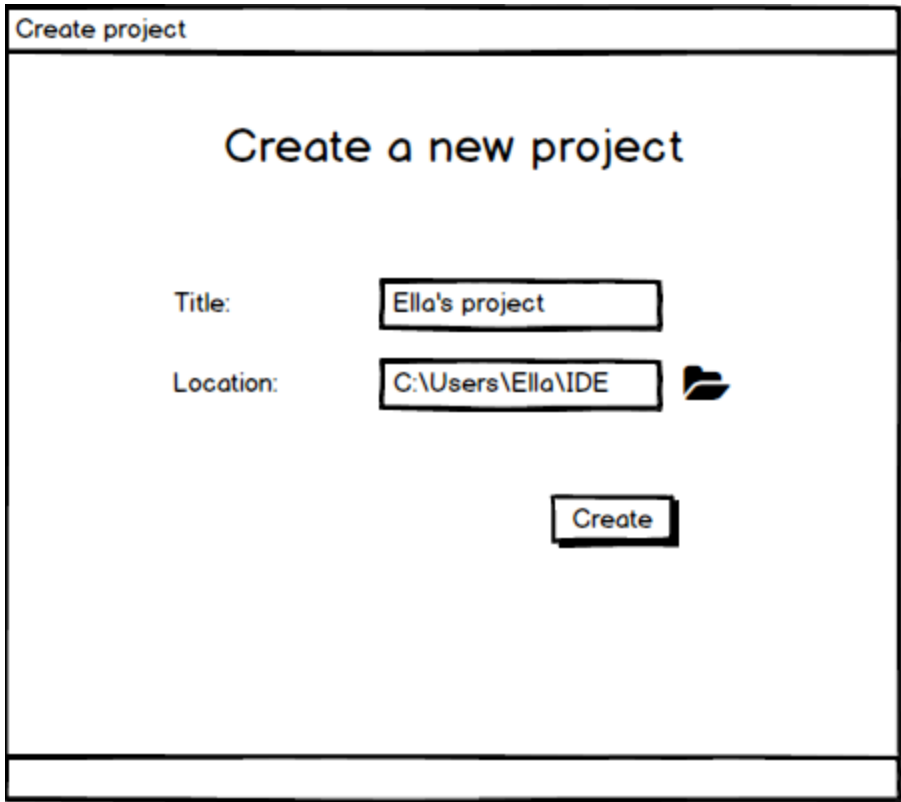


Figure 5.2 - IDE Create Project Window

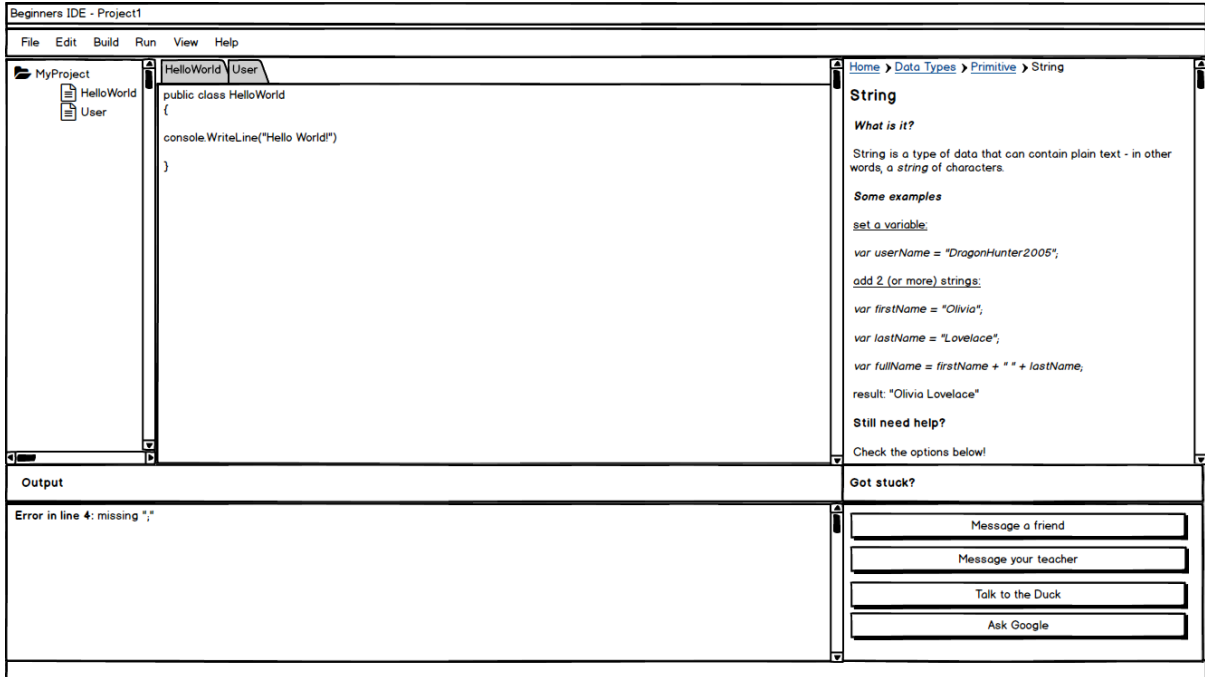


Figure 5.3 - IDE Main Editor Window

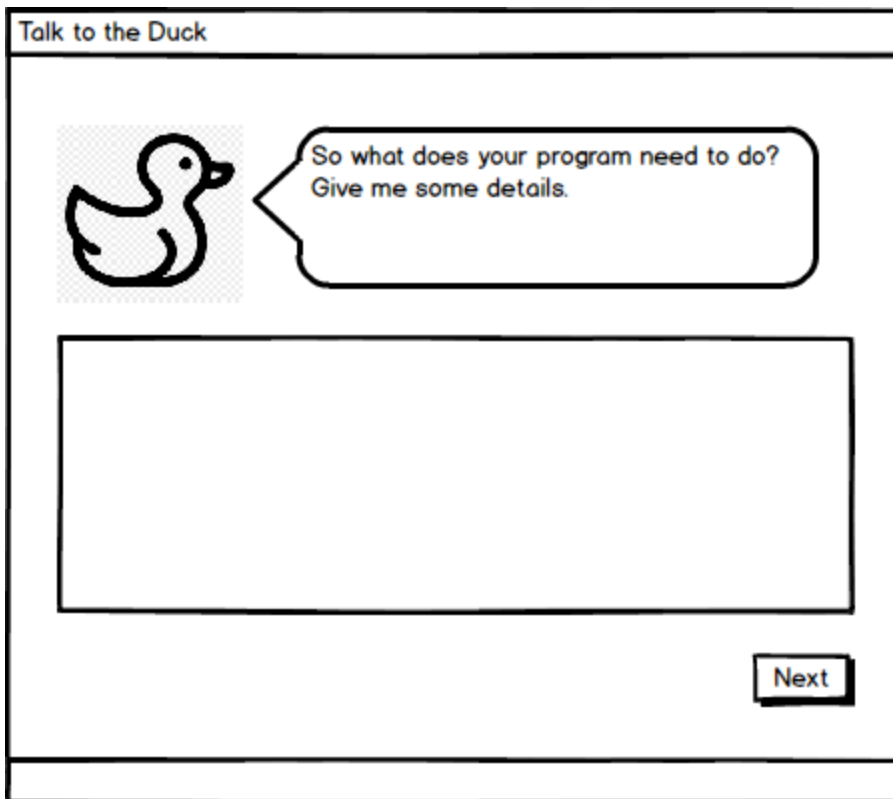


Figure 5.4 - IDE Duck Debugging Window

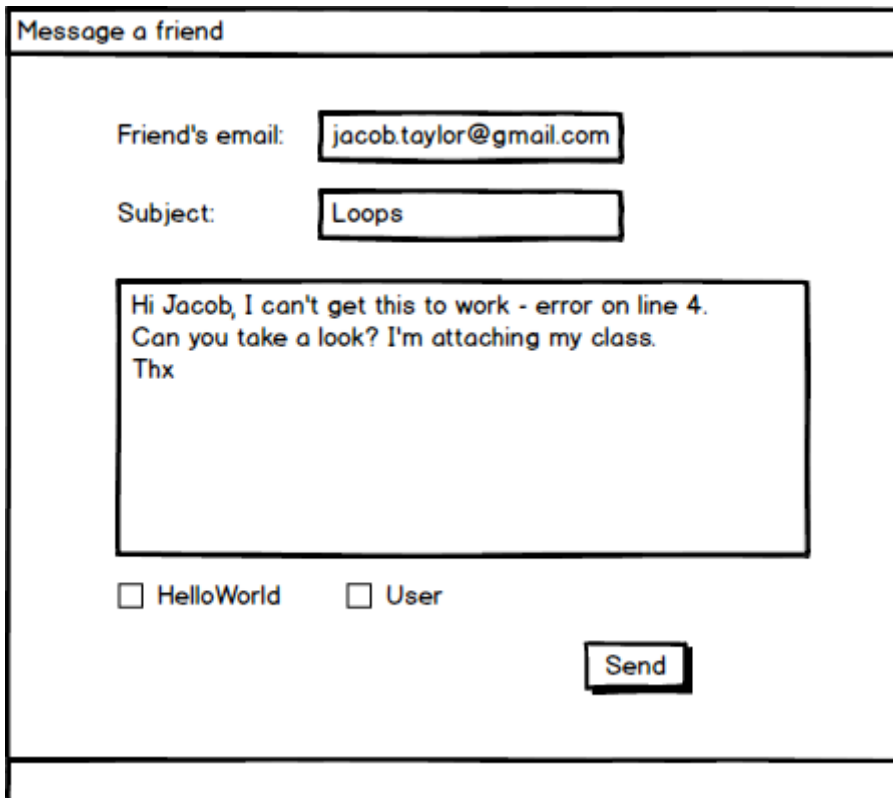


Figure 5.5 - IDE Email to a Friend Window

Message teacher

Teacher: Mr William Smith ▼

Subject: Loops

Hello Mr Smith,
 Can you take a look at my code? I've got an error on
 line 4 and I don't know how to fix it.
 I'm attaching my class.
 Thank you

HelloWorld User

Send

Figure 5.6 - IDE email to teacher window

1.3.4. Name and Logo

The name of the software was chosen to be “IDEal”, which is a combination of the word *ideal* and the acronym of Integrated Development Environment (IDE). A red and orange logo for the program was designed using online graphic editor.



Figure 5.7 – Logo of the Software

1.4. Implementation

1.4.1. The Development Team

The first prototype of the software was developed by a couple of undergraduate Software Engineering students at the University of Northampton, based on the original ideas and design provided by the author. The students, Mr Murtada Dohan and Miss Hanan Al-Mamoori (Dohan and Al-Mamoori, 2016), did this project as a part of their dissertation and closely cooperated with the author during the development and initial testing process, by sending reports and attending meetings.

Once the students have completed their dissertation, the project was taken over by a professional software engineer, Mr Tomasz Poblocki, who voluntarily continued the development work outside his day job.

The latest version of the software is available online, via GitHub (Poblocki, 2020).

1.4.2. Platform

The choice of an appropriate programming environment was discussed with the development team in order to make sure they were confident in using it and would be capable of developing the software application. As a result, the software was written in C# language and it can run on Windows operating system.

1.4.3. Graphical User Interface (GUI)

The GUI was based on the original design and it closely matches the mockups.

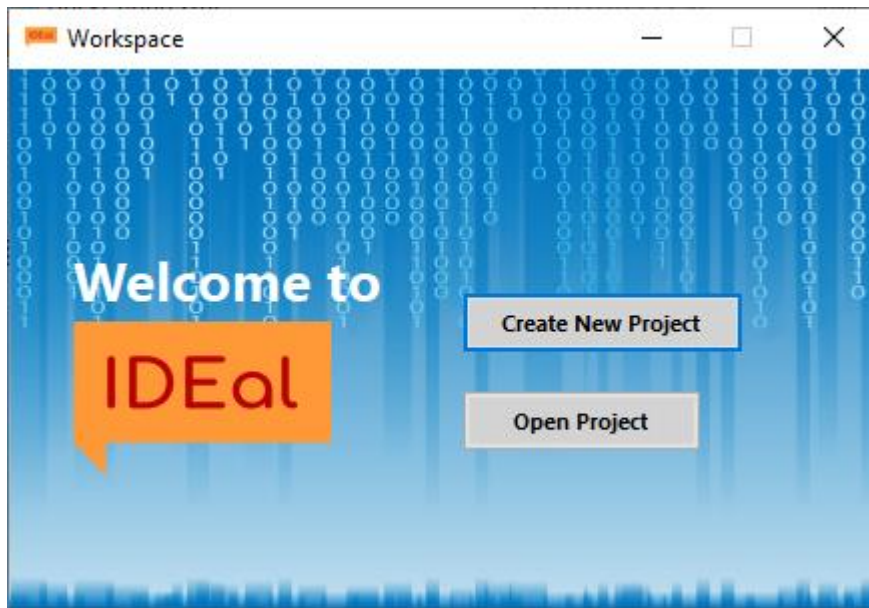


Figure 5.8 - Workspace Window

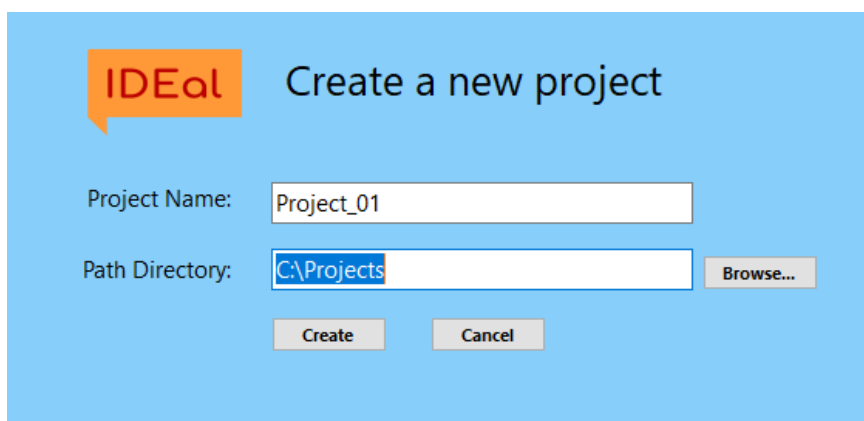


Figure 5.9 - Create New Project Window

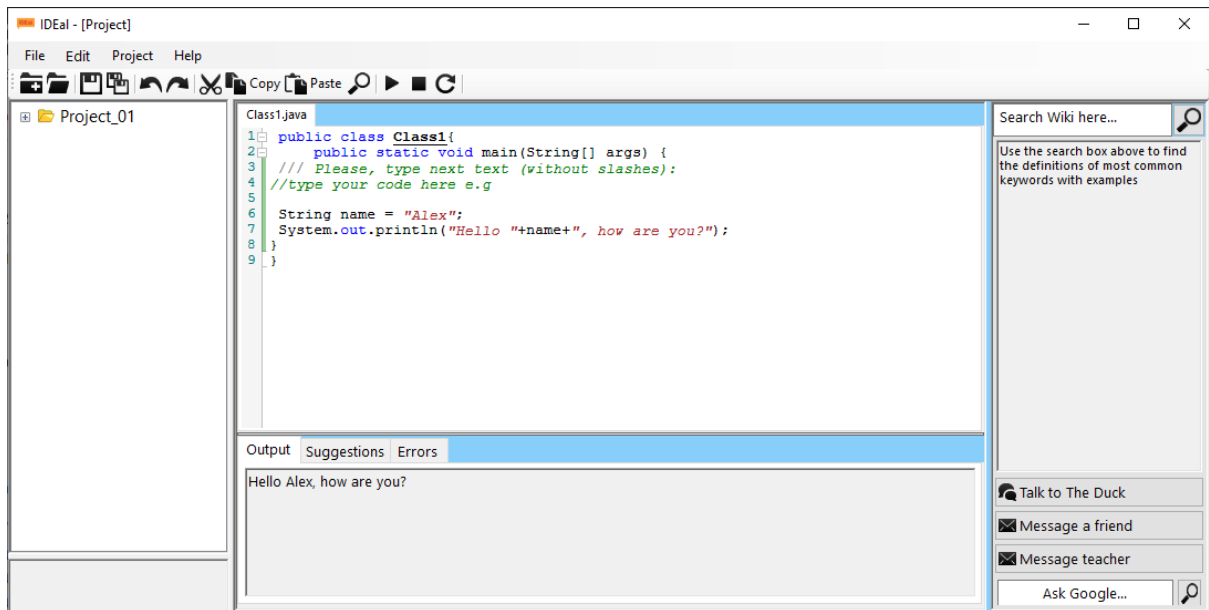


Figure 5.10 - Main Editor Window with Correct Compiled Code



Figure 5.11 - Talk to the Duck Window

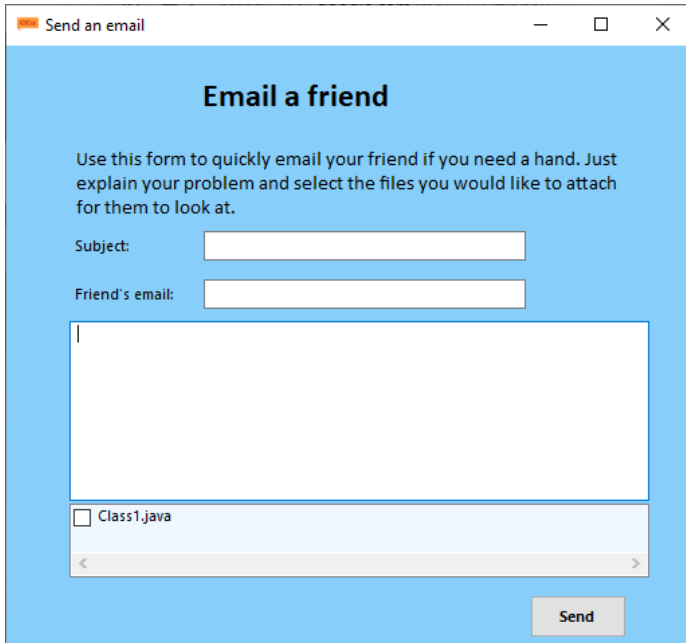


Figure 5.12 - Send Email (to Friend) Window

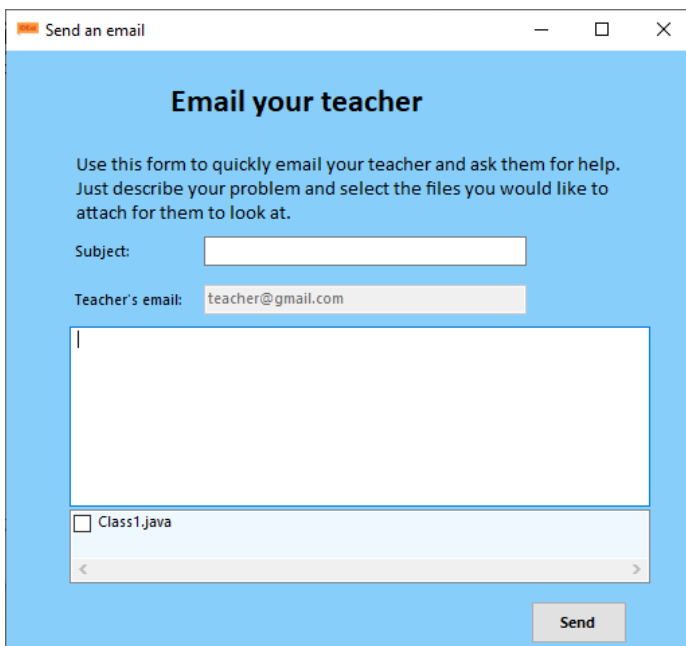


Figure 5.13 - Send Email (to Teacher) Window

1.5. Evaluation and Testing

1.5.1. Incremental Testing

The software was developed incrementally using the agile model, meaning that the software was regularly released and shared with the author via Git during the implementation process. This allowed to test new features as soon as they were implemented, provide feedback regarding functionality and report bugs.

The following features were tested:

Feature	Comments
Create a project	User can successfully create a new project with a desired name in a chosen directory. The programme also prompts them to create the first Java class file.
Open existing project	User can open a previously saved project.
Add/remove a class	User can add or remove class files from a project.
Syntax highlighting	As user type their code, the font colour is changed for different keywords, values and comments.
Code completion	When the user starts typing a dropdown with suggestions appears below the text. Clicking on a suggestion completes a keyword or other language structure.
Error detection	When the user types incorrect code this is immediately indicated by a red triangle next to the line number. Details of the errors are also displayed in the "Errors" tab, located in the horizontal panel below the editor area.
Compile – without errors	If the code does not contain any errors, the programme will compile and the output will be presented in the "Output" tab, located in the horizontal panel below the editor area.
Compile – with errors	If the code contains errors it will not compile and there will be no output. Instead, the user will be directed to the "Errors" tab where they will be able to see the details.
Save project	The user can save their project at any time. The project is also automatically saved on exit.
Wiki – find by double-click on keyword in code	When the user double-clicks on a keyword, e.g. "String", they can view the definition and examples in the Wiki section located in the right panel.
Wiki – find via search box	When the user types a keyword, e.g. "String", in the search box in the right panel, they can view the definition and examples in the Wiki section below.
Duck debugging	When the user clicks the "Talk to the Duck" a new window appears. The user can type down their answers and describe their code in 3 steps.
Email to friend	The user can send an email to a chosen person and attach a Java class file.
Email to teacher	The user can send an email to a pre-defined teacher's email address and attach a Java class file.
Google search	The user can type a query in the "Ask Google..." textbox and click the magnifier button/click Enter and they will be taken to a web browser with Google search results.

Table 5.2 - IDE's Testing Summary

1.5.2. Limitations

Due to the complexity of the software, limited time and human resources, IDEal has got a number of limitations that could not be avoided. These limitations have been identified during the development and initial testing, listed below:

- **syntax highlighting** – there was no straightforward way of implementing Java syntax highlighter in a piece of software written in C#. As a result, IDEal uses highlighting for C# language, which shows a lot of similarity to Java, especially in the most basic code. But of course, the highlighting is not 100% accurate.

- **code wiki** – due to time limitations, the wiki contains only 8 keyword definitions. There are also no links in the wiki that would allow users to navigate from one article to another.
- **code completion** – for similar reasons, code completion offers limited number of options, focusing on the most common keywords and methods.
- **object-oriented programming** – IDEal currently does not support programmes with objects.
- **user input** – it is not possible to write a programme that would read and process input from users, e.g. a console application.
- **error indication** – the red triangle that should appear next to a line that contains error is not always accurate, sometimes failing to disappear when the error has been corrected.
- **error description** – while the software does provide the key information about every error, the messages are not as user-friendly as it was originally intended.
- **duck debugging** – “Talk to the Duck” feature is a very basic implementation that lacks artificial intelligence or any form of sophisticated input processing.

While most of these limitations would not be acceptable in the ultimate version of IDEal, they are tolerable in the prototype version. The aim of the trials is to evaluate a number of possible ways to support students in programming, rather than try to develop advanced software or explore technical limitations. Therefore, the prototype of IDE is considered satisfactory for trials.

1.5.3. Trials

As described in the Methodology section in Chapter 3, the trials were conducted among two different groups: secondary school computing teachers and 3rd year computing students from the University of Northampton.

1.5.3.1. Technical Challenges

Testing software by teachers and students posed a challenge, due to the level of difficulty in setup. There are several key requirements for the software to run:

- Windows operating system,
- installed correct version of Java and Java Development Kit (JDK)
- correct software configuration

These could be troublesome to the teachers and potentially discourage them from testing – many of them said during the interviews that they were overworked, so their time is valuable. Therefore, the author needed to search for a solution that would allow the teachers to test the software without having to install it. Computing students are perhaps less busy than teachers, however, finding an easy way of delivering the software is equally important for reaching a high response rate.

The best option on the market that met all requirements was Amazon Web Services (AWS). One of Amazon’s products called AppStream 2.0 allows to securely deliver desktop applications via web browser. Administrators can install and configure their software on a cloud Windows Server, which is saved as an image. The image is then used to create a fresh instance of the server for each testing user, giving them access to the pre-configured software and guaranteeing identical experience (Amazon Web Services, 2020).

1.5.3.2. Test Scenario

Teachers and students participating in the trials had no previous knowledge and experience with the IDEal software, meaning that they would not be aware of all available functionalities straight away. To help them with the task, a test scenario was developed (Appendix 10).

The scenario lists several tasks that should be completed as a part of the test in order to make a full use of the software’s capabilities. It was designed to act as a guidance and to make sure that no functionality has been missed. It was not mandatory to follow it strictly and stick with the order of tasks – the teachers were given freedom to act on their own as well.

The scenario contains code examples which could be useful to those teachers and students who do not have any experience with Java programming language.

The table below presents a summary list of tasks and the features that they are meant to test:

Task	Feature(s) tested
Enter your name and email	Required for sending emails
Enter a student’s name and email	Required for sending emails
Create a project	Creating a new project/class
Write correct code	Syntax highlighting, code completion
Compile	Compiling and viewing output of valid code
Write incorrect code	Instant error detection
Compile	Failed compilation and error messages
Use Wiki	Code Wiki
Talk to the Duck	Rubber duck debugging
Send emails	Email friend/teacher and attach code
Search Google	Google search

Table 5.3 - A Summary List of the Tasks in Test Scenario

1.5.4. Feedback Analysis

1.5.4.1. Students’ Survey

Within 2 weeks, the IDE software was tested by 23, 3rd year computing students from the University of Northampton, who left their feedback using an online questionnaire. In this section, they will be referred to as “students” or “participants”.

The students were asked a number of questions about the software that they had tested. The purpose of these questions was to establish the validity of several issues:

- Would a software like this be suitable for school education?
- Could a software like this improve student experience?
- Are the proposed features a good or bad idea?
- Could a software like this give students industry-like experience?

The charts below present the answers that were given to each of the questions during the survey.

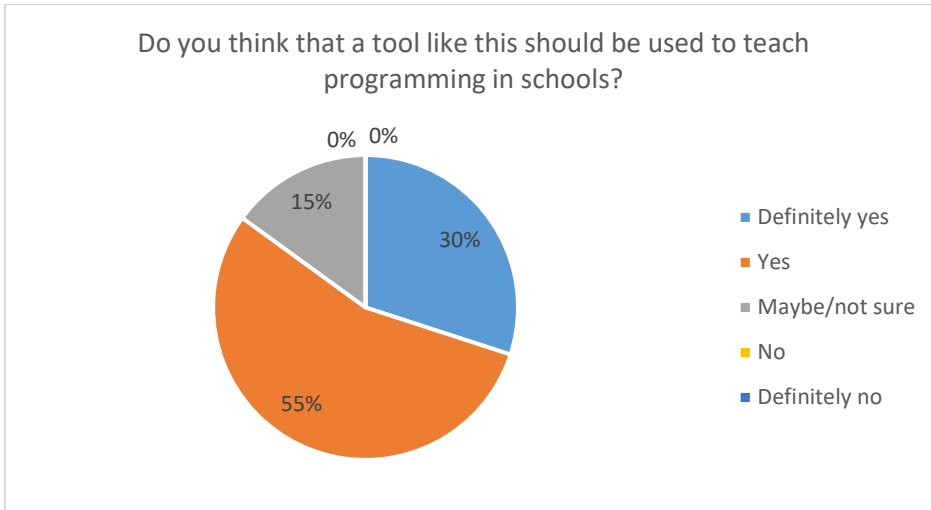


Figure 5.14 - Is Software Like this Suitable for School Education

A great number of participants, 85% in total agreed that a tool like this suitable to use in schools. Only 15% did not have a clear opinion and none of them disagreed.

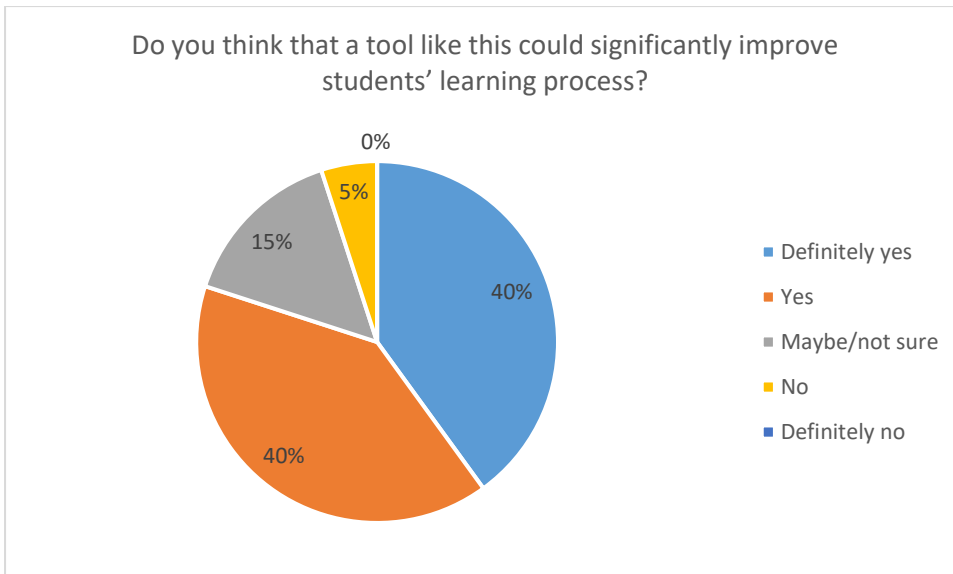


Figure 5.15 - Can Software Like this Significantly Improve Learning Processes

A similar number, 80% believe that a software like this could have a positive impact on pupils' learning process. Just like in the previous question, 15% of the participants were unsure. Only a small number 5% disagreed. The answers to this question were very consistent with the first one.

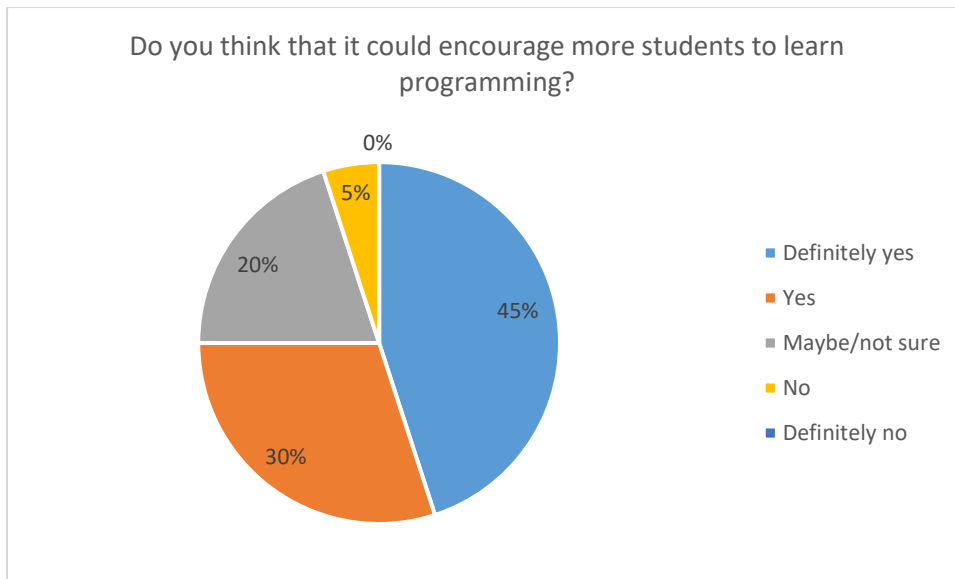


Figure 5.16 - Can Software Like this Encourage Students to Learn Programming

75% of participants believe that a tool like this could be an incentive to learn programming, with the largest number of them so far selecting the “definitely yes” answer. On the other hand, more students – 20% – were not sure about that. And again, only 5% disagreed.

In the next question, participants were asked to rate every given feature of the software on a scale from 1 (“terrible idea”) to 5 (“great idea”). Based on their answers, a ‘score’ was calculated for each feature, expressed in points that correspond to options on the scale.

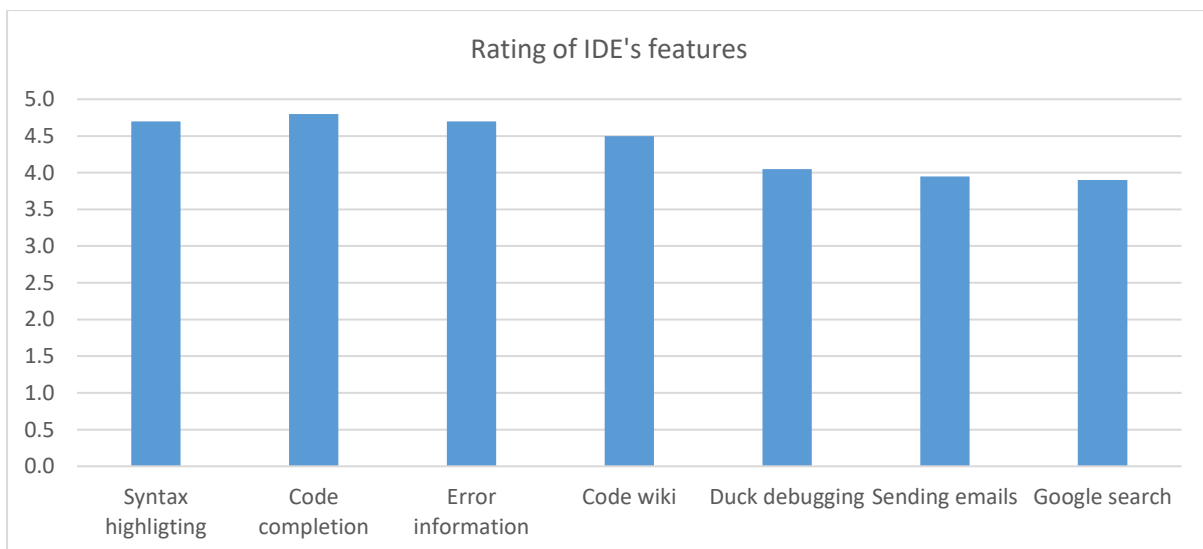


Figure 5.17 - Rating of IDE's Features

The most useful feature, with 4.8 points, was code completion, followed closely by syntax highlighting and error information which scored 4.7 points. Code wiki was also highly rated, with 4.5 points. Based on the score of these four features (≈ 5) they could be considered a “great idea” (5). The next three features had slightly lower ratings: from 4.1 to 3.9 (≈ 4), making them a “good idea” (4).

Syntax, code completion and error information were the features which computing teachers and educators expressed their interest in (via #CASchat and interviews). An interesting observation is that the ‘social features’ were rated visibly lower than the rest.

In the next question, the participants were asked to choose one feature which they personally liked the most. The results were quite surprising, as they do not match the feature rating.

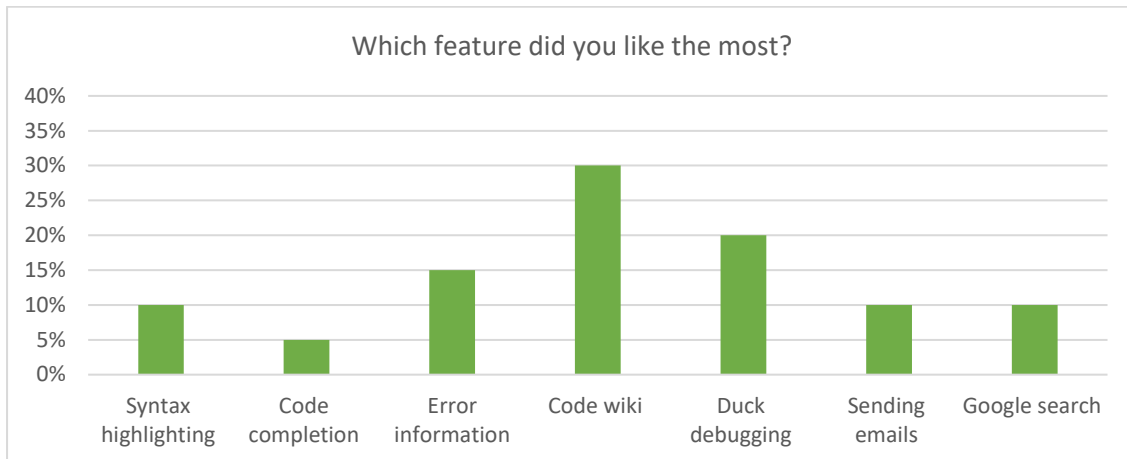


Figure 5.18 - Students' Favourite Feature

The unexpected winner was code wiki, chosen by 30% of the students. The second most popular feature was duck debugging (20%), followed by error information (15%). Sending emails, Google search and syntax highlighting were selected by 10% of participants, while the “greatest” feature (as could be read from previous chart), was selected only by 5%.

A possible explanation could be that features like code completion and syntax highlighting are common in professional IDEs and students are familiar with them. Therefore, the participants already knew the value of these features and rated them highly. Features like code wiki or duck debugging are not a standard, so perhaps they felt more innovative and exciting to students. This theory, however, would require further research.

Another factor that could affect the answers were the technical issues with sending emails and Google search, which were reported via comments. Experiencing these difficulties during the test could lead to lower satisfaction and lower rating in the survey.

The final two questions were related to the computing industry.

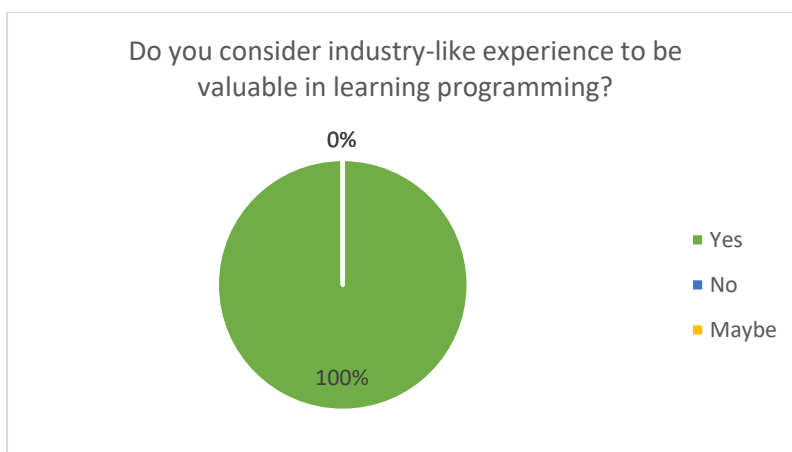


Figure 5.19 - Is Industry Experience Valuable in Learning Programming

All students, without exception, considered industry-like experience to be important when learning how to programme. It is the only time during the research when the participants were unanimous.

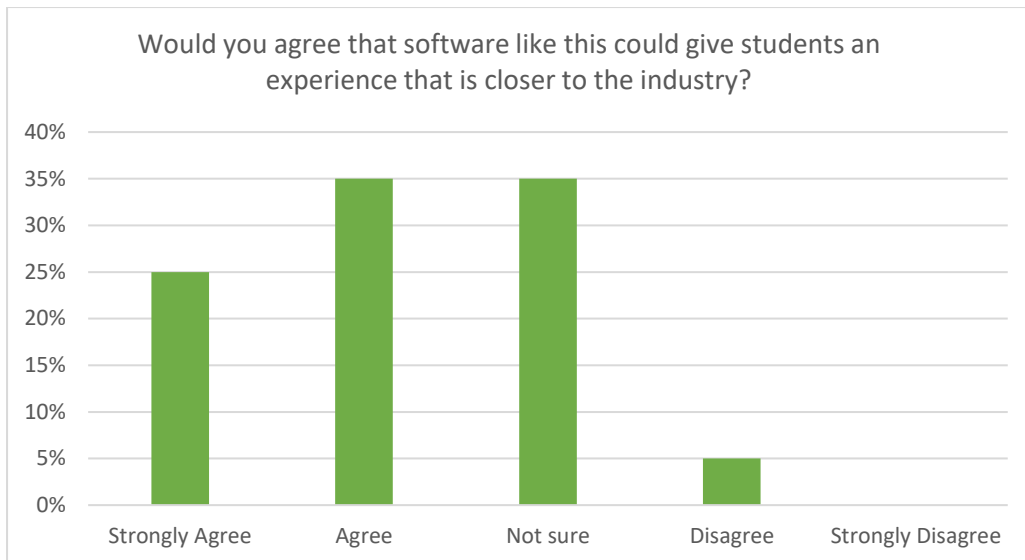


Figure 5.20 - Can Software Like this Give Students Industry-like Experience

60% of the students in total agreed that software like this could provide an experience close to the computing industry. Only a small number of participants (5%) believed that this was not true. Finally, 35% of them did not have a clear opinion, which is significantly more than in previous questions.

Further analysis of this question, based on weighted arithmetical mean, shows that the overall rating of the software in terms of industry-like experience is 3.8 points on a scale from 1 to 5. The values on the scale correspond to answers from “strongly disagree” to “strongly agree”. The 3.8 score can be interpreted as “agree” (≈ 4).

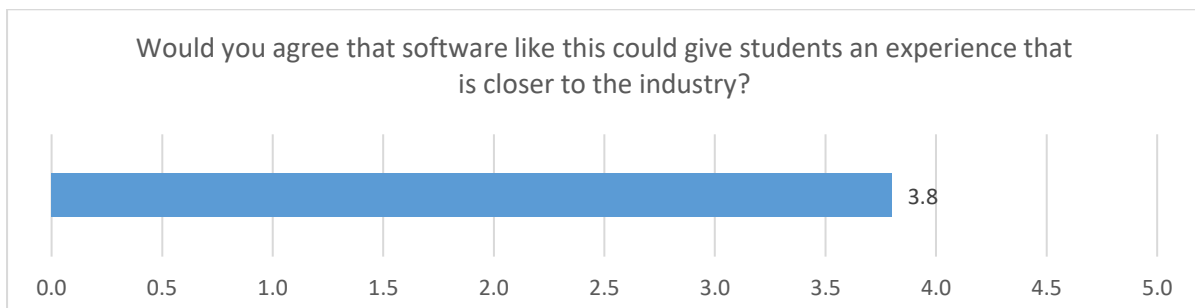


Figure 5.21 - Can Software Like this Give Students Industry-like Experience - Average Answer

Students' comments

Participating students were given the opportunity to leave a comment and 70% of them decided to do so. As could be expected from people who study computer science, many of them reported bugs and expressed their opinion about technical aspects of the software. This, however, had little relevancy to the research, so such comments were not considered in analysis. Still, all comments were read as soon as they appeared, allowing to address any problems that the participants encountered – e.g. issues with the email feature or Google search.

There was a number of positive comments (31%) from students who found IDEal a good way of supporting beginners in learning how to program:

"I feel IDEal is a great idea and could really help younger people get into programming."

"Overall, I think it is a good concept, especially the duck debugging"

"The overall program looks good to use as it is not so cluttered with options."

"Good software, great ideas for features."

"The wiki one that lets you search something is great! It works so you don't really need the google search."

Critical comments focused on several areas, as presented in the table below:

Issue	Comments
Old-fashioned GUI	31% of commenting students mentioned that they did not like the visual side of application and found it <i>"a bit old-fashioned looking"</i> . Their recommendation was to <i>"improve the aesthetics"</i> , <i>"make it prettier and more user-friendly"</i> .
Duck-debugging improvements	25% of the comments contained feedback about the duck debugging feature. Although some people said that they liked it, most of them had similar suggestions to improve this feature: <i>"it could you show me all my answers afterwards"</i> and <i>"Having a summary after the duck debugging may be useful for the students"</i> .
Inaccurate error suggestions	Some students (19% of comments) reported that <i>"the icons/colours to indicate where the errors are is not accurate"</i> . There was a suggestion that <i>"error messages could be made more specific by narrowing down where the error occurs"</i> . These issues were recognized by the author and included in the Limitations section.
Chat instead of emails	Two participants said that they would rather see a chat feature instead of sending emails: <i>"The email feature is ok, but during live classes what makes more sense is to have a sort of live chat feature instead"</i> .

Table 5.4 - Student participants' comments

Another comment that the author found interesting and worth quoting, suggests integration with third party forums:

"I would love to see 3rd party forums to be integrated into developer software as I believe that having direct access to places such as Stack Overflow, Google Search and a number of other sources could significantly improve the productivity of a person and decrease the time required to troubleshoot or implement a new feature"

Comments left by the participants show that despite a number of technical issues with the software they were overall positive about the concept that it represents and liked some of the non-standard features such as duck-debugging and code wiki. They also provided a number of constructive suggestions on how the software could be improved and made more beginner-friendly. This proves that they see room for further development of this concept.

1.5.4.2. Teachers' Survey

Emails with invitation to participate were sent to the 7 teachers who participated in the interviews in the earlier stage of research. 6 of them replied, expressing their interest in this project and were sent further details. One teacher even forwarded the email to his colleagues who assured that they would be happy to help with the research. In total, 10 emails with invitation to software trials were sent.

Unfortunately, in the end, only 4 teachers did the test and left their feedback via online questionnaire. This is a lot less than expected and cannot be considered as statistically significant. The main reason behind the low response rate could be the time. During the interviews, teachers complained that they were overworked and had little time for other activities (such as self-education or doing more activities in the classroom), so participating in a trial could be a challenge, despite their best intentions.

They were asked almost identical questions as the group of students, with only slight adjustments to the wording and one additional question about their pupils. Data collected from the 4 teachers is presented on the charts below and hopefully can provide an interesting insight.

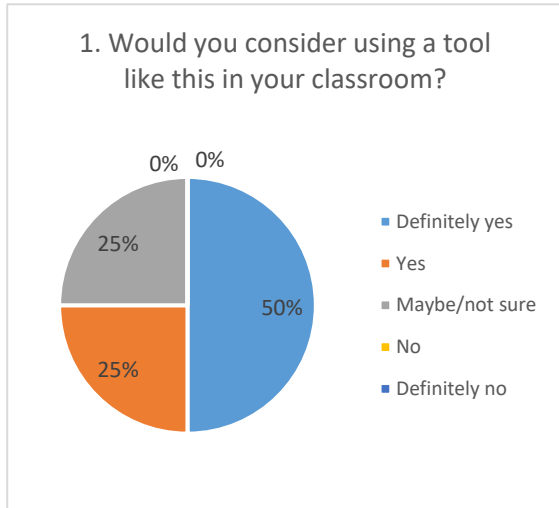


Figure 5.22 - Would Teachers Consider Using Software Like this in Their Classroom

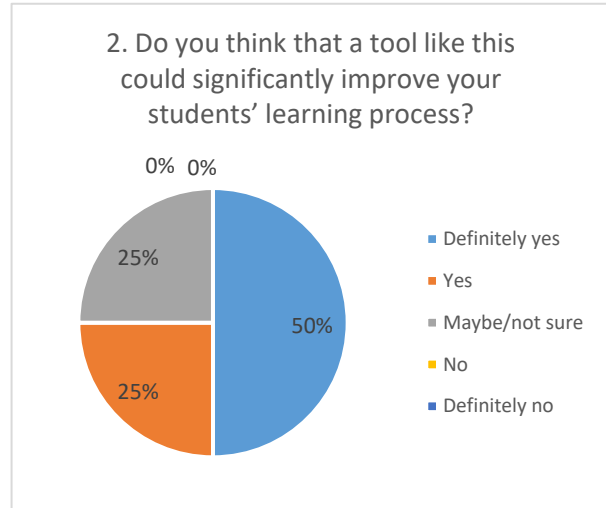


Figure 5.23 - Can Software Like this Significantly Improve Learning Process

Three teachers stated that they would consider using software like this in their classroom, two of them “definitely”. One of the teachers was not sure, but no-one disagreed. The answers were similar when asked if a tool like this could have a positive impact on their students’ learning process.

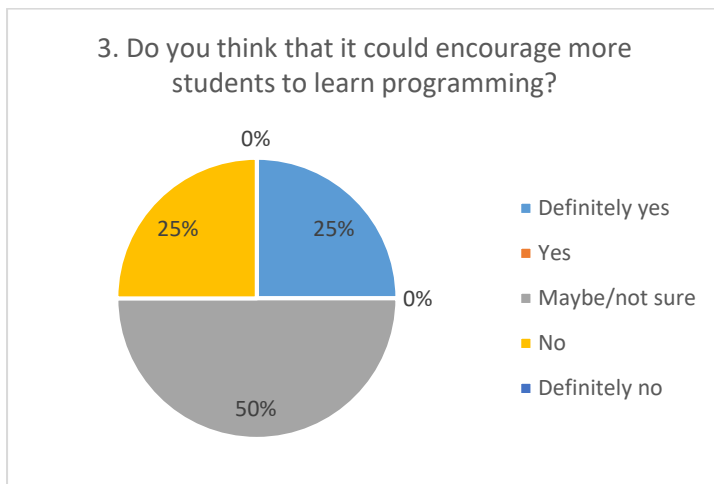


Figure 5.24 - Can Software Like this Encourage Students to Learn Programming?

The teachers were less convinced if a tool like this could encourage more students to learn how to programme – only one of them was positive about it. Two teachers were unsure and one replied “no”.

In the next question, they were asked to rate different features of the software in the same way as computing students.

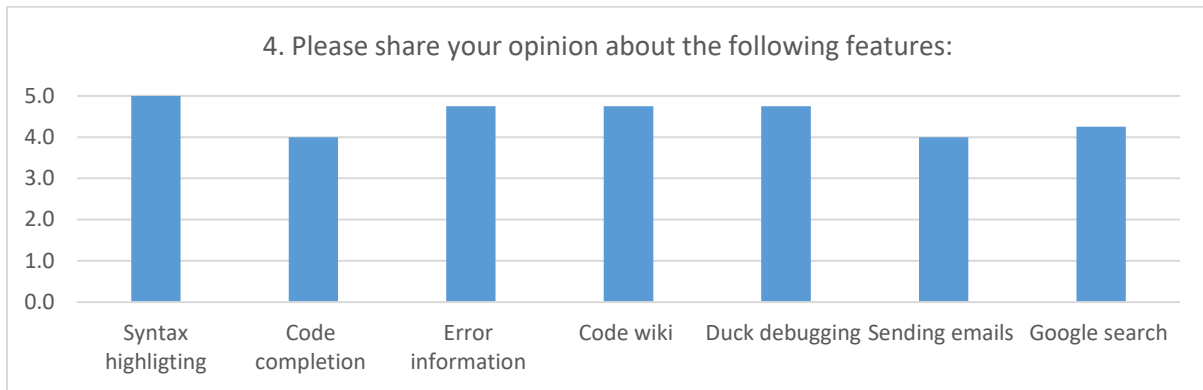


Figure 5.25 - Teachers' Rating of IDE's Features

Just like in the student survey, an average 'score' was calculated for each feature and expressed in points that correspond to the following answer options: from "terrible idea" (1) to "great idea" (5).

As can be read from the chart, all four teachers found the syntax highlighting to be a "great idea" (5 points). This does not come as a surprise: syntax is the major problem for students switching from blocks to text-based programming, as was reported by many educators during #CASchat and later by computing teachers.

Other highly rated features: error information, code wiki and duck debugging, which got 4.8 points and could also be considered a "great idea" (≈ 5). The remaining 3 features were rated from 4.3 to 4, making them a "good idea" (≈ 4).

These results share some similarity to what students thought – Google search and sending emails also were given lower rating. The most visible difference was in teachers' opinion about code completion. While students found it to be the best feature, teachers thought the opposite.

In the next two questions they were asked which feature was their favourite and which feature could be their pupils' favourite:

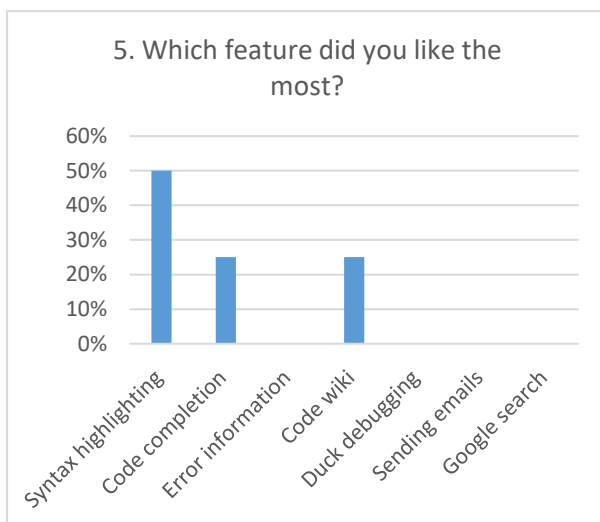


Figure 5.26 - Teachers' Favourite Feature

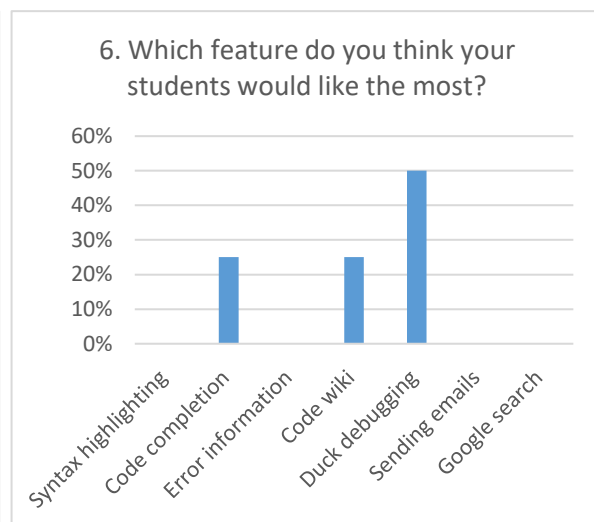


Figure 5.27 - Pupils' Favourite Feature, as Predicted by Teachers

Teachers' 'personal' favourites were: syntax highlighting (selected by 2), code completion and code wiki. Their answers were not identical in the second question: teachers believed that their pupils would appreciate duck-debugging, rather than syntax highlighting.

The last two questions were related to the potential value of the computing industry.

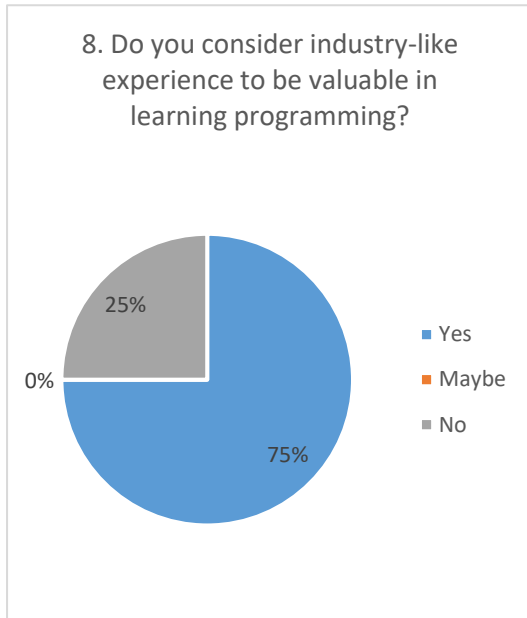


Figure 5.28 - Is Industry-like Experience Valuable in Learning Programming

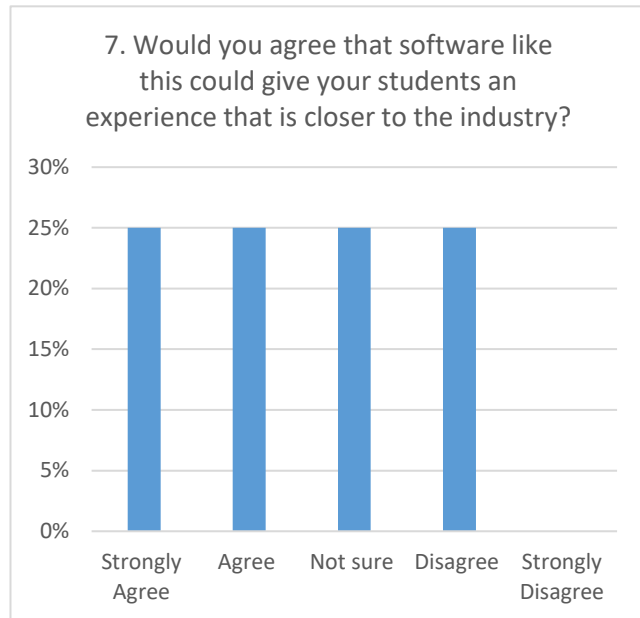


Figure 5.29 - Can Software Like this Give Students Industry-like Experience

Three teachers agreed that industry-like experience is valuable when learning how to programme. One of them, however, thought otherwise.

Opinions about the software were mixed: every teacher selected a different option. Two of them were positive ("strongly agree" and "agree"), one was unsure and one disagreed that a software like this could provide industry-like experience to students.

Teachers' Comments

Three out of four teachers decided to leave a comment. The key points from their messages are presented on the graphic below:

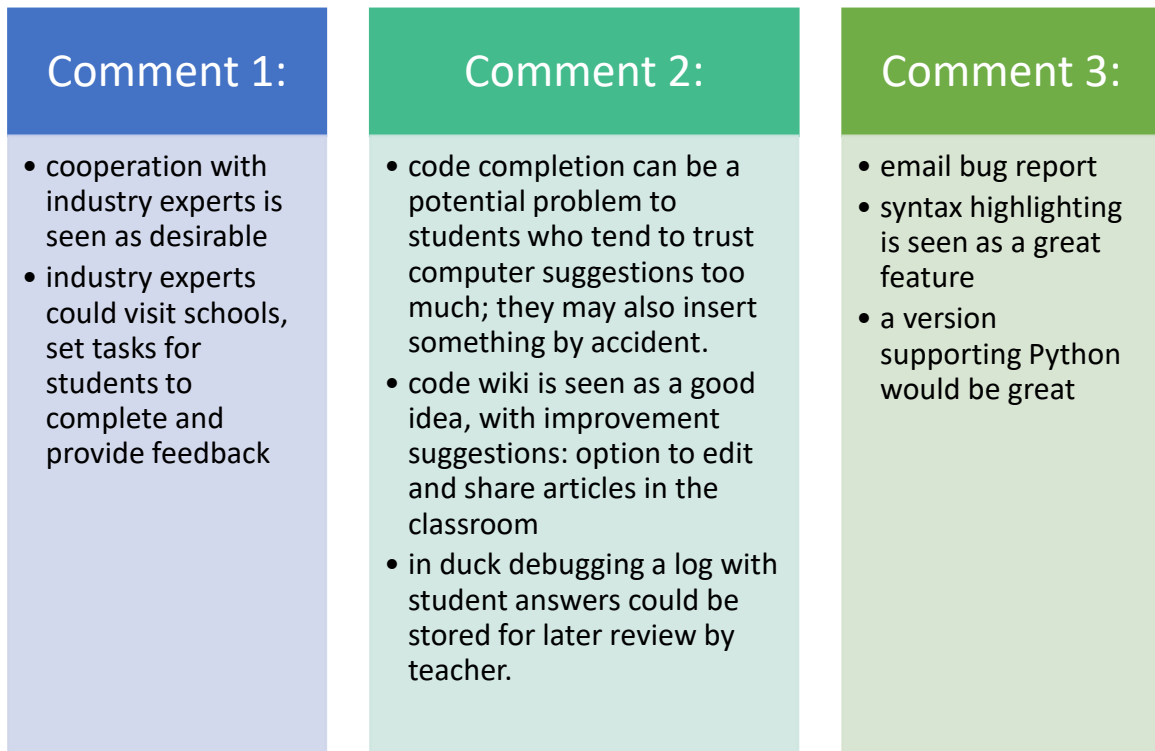


Figure 5.30 - Teacher participants' comments

1.5.5. Summary of IDE Evaluation and Testing

Despite a number of technical difficulties that were reported during the tests, students' feedback was overall very positive. 85% of them considered a software like this to be useful for schools and 80% believe that it could have a positive impact on pupils' learning experience. Only slightly less participants think that it could encourage pupils to learn programming.

Students saw the IDE's features either as "great idea" or "good idea". The highest rating was given to the standard features that are common in most of modern development environments (syntax highlighting, code completion, error detection). The more 'social' features (duck debugging, sending emails and Google search) received slightly lower rating, possibly due to technical issues.

Participants' favourite feature was clearly code wiki, chosen by 30% of them. It was followed by duck debugging (15%). Other features were not as popular and surprisingly code completion was the least favourite (5%). The reason behind it could be that this feature is already very common in professional IDEs and therefore attracts less attention than non-standard features. This theory would require further research.

Students were unanimous when asked about the value of industry-like experience in learning how to programme – all of them believed it was important. They were a little less convinced if a software like IDEal could provide such experience – only 60% of them agreed.

In the comments, many students (31%) praised the idea of the software and provided constructive feedback. This proves that they see the need for software like this and room for further improvement and development.

The four teachers who participated in the trials were more cautious in their assessment. Most of them (3) would like to see software like this in their classroom and believe it could improve their pupils'

performance, however they were less convinced if it could attract more pupils to programming (2 of them were unsure, 1 disagreed and 1 agreed).

Teachers rating of the IDEal's features was just as high as students, ranging from "great idea" to "good idea". The top feature was syntax highlighting, which was rated as "great" by all four teachers. The lowest rated feature was code completion, which is opposite to what the students thought.

Their most favourite features were: syntax highlighting (chosen by 2 teachers), code completion and code wiki. However, they believe that their pupils' favourite would be duck debugging (chosen by 2 teachers).

Three teachers agreed that industry-like experience is valuable for learning programming, but the opinions were divided whether a software like IDEal could provide such experience: answers ranged from "strongly agree" to "disagree".

Three teachers decided to write comments with feedback. Just like students, they also believed there was a potential in the software and shared some improvement suggestions.

Overall, the software was well-received and at this stage, based on available data it can be considered a success. Further research is required to collect statistically significant data from computing teachers that would allow to come up with more accurate answers to the key questions: is the software suitable for schools, what are the most useful features and can it provide industry-like experience to students.

Appendix 11 – Software Trials – list of tasks

Task	Feature(s) tested	Instructions
Enter your name and email	Required for sending emails	Enter a name and your real email address . This will be used to test emails.
Enter a student's name and email	Required for sending emails	Enter a name and a real email address . This will be used to test emails.
Create a project	Creating a new project/class	Create a project with a desired name (preferably within the default location) and add a class as prompted.
Write correct code	Syntax highlighting, code completion	Type simple valid code. E.g.: <pre>String name = "Alex"; System.out.println("Hello "+name+", how are you?");</pre> OR <pre>String birthYear = "1985"; String currentYear = "2020"; int age = Integer.parseInt(currentYear) - Integer.parseInt(birthYear); System.out.println(age);</pre>
Compile	Compiling and viewing output of valid code	Click the compile button [▶] and wait for the results (<i>this may take a little longer than expected due to the testing environment</i>)
Write incorrect code	Instant error detection	Type invalid code and take a look at the Errors tab below the editor. E.g.: <pre>int favouriteNo = "seven";</pre> (<i>declaring integer, but giving it a string value</i>) OR <pre>String month = "February"</pre> (<i>missing semicolon</i>)
Compile	Failed compilation and error messages	Click the compile button [▶] and wait for the results. Check the Errors tab.
Use Wiki	Code Wiki	<ol style="list-style-type: none"> 1) Hover over a keyword in your code, e.g. <code>int</code> or <code>String</code>. 2) Double click on a keyword in your code, e.g. <code>int</code> or <code>String</code>. 3) Use the search box in the top right corner to search for a keyword, e.g. <code>comment</code> or <code>array</code>
Talk to the Duck	Rubber duck debugging	Click on the [Talk to the Duck] button below the Wiki and follow the instructions.
Send emails	Email friend/teacher	Click [Email friend] / [Email teacher] buttons. Send emails with attachments.
Search Google	Google search	Write any query e.g. <i>how to trim string in java</i> OR <i>how to convert string to int in java</i> ...and click [Enter] or search button [🔍].

Appendix 12 – Software Trials – initial email to teachers

Dear [First name],

This is Alex – PhD student from The University of Northampton. Some time ago you were kind enough to help me out with my research by participating in an interview.

Today I would like to ask for your help one last time. I have developed a prototype of a development environment for school students and I would be delighted if you could try it out and let me know your impressions. I am aware that teachers are very busy these days, but I can promise that this would not take more than 30 mins of your time.

The software is a desktop application; however, I found an easy way to make it available via browser, using Amazon Web Services. Thanks to this the application does not need to be configured and it's ready for testing at any time.

Please let me know if this is something you could do and if yes, I will send you further details. I would greatly appreciate your help on this final stage of my PhD research.

Many thanks,

Alex

Appendix 13 – Software Trials – email to teachers with further information

Hi [First name],

Thank you so much! Here are the instructions:

Please check your inbox for an invitation from Amazon, with the following subject: Start accessing your apps using Amazon AppStream 2.0 (if it's missing please check the spam folder too).

The email contains instructions on how to access the application. Once you have logged in, please click on the application logo (orange with red "IDEa" text) and wait a moment for the server to build.

I have prepared a simple testing scenario to guide you through the software's features. Please use it as a reference, but feel free to try things on your own as well.

Important information: any data that you enter (such as email address) will NOT be stored permanently or processed in any way. As soon as you finish testing and logout all data will be immediately erased.

Once you have finished testing, please spend a few minutes to answer this short anonymous survey:

<https://www.eSurveysPro.com/Survey.aspx?id=e65506fc-0edd-42d1-aa2f-1ca800582da6>

Once again, thank you so much for your time and help!

Kind regards,

Alex

Appendix 14 – Software Trials – teachers’ questionnaire

1. Before you begin...

Thank you for testing the IDEal software. This short survey contains a few questions regarding your experience and opinion about the software that will help me with evaluation.

It is completely anonymous and it should only take a couple of minutes of your time.

2. Your opinion about IDEal

In this section please provide your opinion about IDEal.

1. Would you consider using a tool like this in your classroom? *
 - Definitely yes
 - Yes
 - Maybe/not sure
 - No
 - Definitely no

2. Do you think that a tool like this could significantly improve your students’ learning process? *
 - Definitely yes
 - Yes
 - Maybe/not sure
 - No
 - Definitely no

3. Do you think that it could encourage more students to learn programming? *
 - Definitely yes
 - Yes
 - Maybe/not sure
 - No
 - Definitely no

4. Please share your opinion about the following features: *

	Great idea	Good idea	Not sure	Bad idea	Terrible idea
Syntax highlighting	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
Code completion	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
Error information	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
Code wiki	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
Duck debugging	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
Sending emails	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
Google search	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>

5. Which feature did you like the most? *
- Syntax highlighting
 - Code completion
 - Error information
 - Code wiki
 - Duck debugging
 - Sending emails
 - Google search
6. Which feature do you think your students would like the most? *
- Syntax highlighting
 - Code completion
 - Error information
 - Code wiki
 - Duck debugging
 - Sending emails
 - Google search
7. Would you agree that software like this could give your students an experience that is closer to the industry?
- Strongly Agree
 - Agree
 - Not sure
 - Disagree
 - Strongly Disagree
8. Do you consider industry-like experience to be valuable in learning programming? *
- Yes
 - No
 - Maybe
9. Do you have any comments or suggestions?

3. About you

Finally, please provide a few details about you.

10. How long have you been teaching computing?
- less than 2 years
 - 2-5 years
 - 5-10 years
 - over 10 years

11. Do you have some experience from the industry?

- Yes, a lot
- Yes, but only a little
- No experience at all

12. How would you rate your own programming skills?

- Excellent
- Good
- Average
- Below Average
- Poor

Appendix 15 – Software Trials – students’ questionnaire

1. Before you begin...

Thank you for testing the IDEal software. This short survey contains a few questions regarding your experience and opinion about the software that will help me with evaluation.

It is completely anonymous and it should only take a couple of minutes of your time.

2. Your opinion about IDEal

In this section, please provide your opinion about IDEal.

1. Do you think that a tool like this should be used to teach programming in schools? *
 - Definitely yes
 - Yes
 - Maybe/not sure
 - No
 - Definitely no

2. Do you think that a tool like this could significantly improve students’ learning process? *
 - Definitely yes
 - Yes
 - Maybe/not sure
 - No
 - Definitely no

3. Do you think that it could encourage more students to learn programming? *
 - Definitely yes
 - Yes
 - Maybe/not sure
 - No
 - Definitely no

4. Please share your opinion about the following features: *

	Great idea	Good idea	Not sure	Bad idea	Terrible idea
Syntax highlighting	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
Code completion	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
Error information	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
Code wiki	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
Duck debugging	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
Sending emails	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
Google search	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>

5. Which feature did you like the most? *
 - Syntax highlighting
 - Code completion

- Error information
 - Code wiki
 - Duck debugging
 - Sending emails
 - Google search
6. Would you agree that software like this could give students an experience that is closer to the industry?
- Strongly Agree
 - Agree
 - Not sure
 - Disagree
 - Strongly Disagree
7. Do you consider industry-like experience to be valuable in learning programming? *
- Yes
 - No
 - Maybe
8. Do you have any comments or suggestions?

3. About you

Finally, please provide a few details about you.

9. Did you have computer science classes in your secondary school?
- Yes
 - No
10. How would you rate your own programming skills?
- Excellent
 - Good
 - Average
 - Below Average
 - Poor

7. References

- AbilityNet Tech4Good Awards (2019) Joshua Lowe – EduBlocks. Finalist category: BT Young Pioneer [online]. Available from: <https://www.tech4goodawards.com/finalist/joshua-low/> [Accessed 16 January 2020]
- Alexa (2019) Alexa - Top sites [online]. Available from: <https://www.alexa.com/topsites> [Accessed 15 October 2019]
- Alice (2014) An Educational Software that teaches students computing programming in a 3D environment [online]. Available from: <http://www.alice.org/index.php> [Accessed 9 May 2014]
- Amazon Web Services (2020) Amazon AppStream 2.0 | fully managed application streaming on AWS [online]. Available from: https://aws.amazon.com/appstream2/?did=ft_card&trk=ft_card [Accessed 7 January 2020]
- Bateman, K. (2014) The UK's new computing curriculum is here: Are teachers ready? 14 August 2014. *Computer Weekly*. [online] Available from: <https://www.computerweekly.com/feature/The-UKs-new-computing-curriculum-is-here-Are-teachers-ready> [Accessed 24 July 2015]
- Bau, D. (2015) Droplet, a blocks-based editor for text code. *Journal of Computing Sciences in Colleges*. **30** (6), 138-144. Available from: <https://dl.acm.org/citation.cfm?id=2753052> [Accessed 25 January 2018]
- BCS, The Tech Partnership (2016) *The Women in IT Scorecard. A definitive up to-date evidence base of data and commentary on women in IT employment and education* [online]. Available from: https://www.thetechpartnership.com/globalassets/pdfs/research-2016/womeninit_scorecard_2016.pdf [Accessed 15 January 2017]
- BCS, The Tech Partnership (2017) *BCS deeply concerned over stagnation of number of computer science GCSE applicants* [online]. Available from: <https://www.bcs.org/content/conWebDoc/57904> [Accessed 17 January 2018]
- Bell, T., Alexander, J., Freeman, I., Grimley, M. (2009). Computer science unplugged: School students doing real computing without computers. *New Zealand Journal of Applied Computing and Information Technology*, 13(1)
- Ben-Ari, M. (1998). Constructivism in computer science education. Proceedings of the twenty-ninth SIGCSE technical symposium on Computer science education. Atlanta, Georgia, United States: ACM.
- Biography.com Editors. (2017) Bill Gates Biography.com [online]. A&E Television Networks. Available from: <https://www.biography.com/people/bill-gates-9307520> [Accessed 10th October 2017].
- Biography.com Editors. (2017) Jack Dorsey Biography.com [online]. A&E Television Networks. Available from: <https://www.biography.com/people/jack-dorsey-578280> [Accessed 13th October 2017].

Biography.com Editors. (2017) Mark Zuckerberg Biography.com [online]. A&E Television Networks. Available from: <https://www.biography.com/people/mark-zuckerberg-507402> [Accessed 11th October 2017].

BJSS (2017) 67% of teachers across Britain feel under-equipped to teach coding. [online] Available from: <https://www.bjss.com/67-of-teachers-across-britain-feel-under-equipped-to-teach-coding/> [Accessed 30 May 2018]

Boyle, R. D., Clark M. A. C. (2006) Computer science in English High Schools: We Lost the S, Now the C Is Going. In: Mittermier, R. T. (ed.) *Informatics Education – The Bridge between Using and Understanding Computers*. pp. 83-93

Brandt, A.M., Colton, M.B. (2008) Toys in the Classroom: LEGO MindStorms as an Educational Haptics Platform. *Symposium on Haptic Interfaces for Virtual Environment and Teleoperator Systems, March 2008*, pp.389-395

Brennan, K., Resnick, M. (2012). New frameworks for studying and assessing the development of computational thinking. in. Proceedings of the 2012 Annual Meeting of the American Educational Research Association, Vancouver, Canada., Vancouver, Canada.

Brown, N. C., Sentance, S., Crick, T., & Humphreys, S. (2014). Restart: The resurgence of computer science in UK schools. *ACM Transactions on Computing Education (TOCE)*, 14(2), 1-22.

Bruner, J. (1996). *Towards a theory of instruction*. Cambridge, MA: Harvard University Press.

Burns, J. (2012) *School ICT to be replaced by computer science programme* [online] BBC News, Education & Family. Available from: <http://www.bbc.co.uk/news/education-16493929> [Accessed 31 August 2012]

Butler, B. (2012) Talent pool can't meet skyrocketing demand for cloud skills. *Computerworld*. 14 March [online]. Available from: <https://www.computerworld.com/article/2502648/talent-pool-can-t-meet-skyrocketing-demand-for-cloud-skills.html> [Accessed 24 June 2019]

Carbonnelle, P. (2019) *Top IDE index* [online] Available from: <https://pypl.github.io/IDE.html> [Accessed 5 February 2019]

Child Development Media (2017) Play: The Work of Lev Vygotsky [online]. Available from: <http://www.childdevelopmentmedia.com/articles/play-the-work-of-lev-vygotsky/> [Accessed 12 December 2017]

Code.org. (2013). What Most Schools Don't Teach [online]. Available from: <https://www.youtube.com/watch?v=nKlu9yen5nc> [Accessed 10 October 2017]

Codecademy (2017) Codecademy [online]. Available from: <https://www.codecademy.com/> [Accessed 12 May 2017]

- Codecademy (2019) What Is an IDE? [online]. Available from <https://www.codecademy.com/articles/what-is-an-ide> [Accessed 10 June 2019]
- Combs, V. (2019) Decade in review: Software developers and cloud architects in demand and well paid. *TechRepublic*. 6 December [online]. Available from: <https://www.techrepublic.com/article/decade-in-review-software-developers-and-cloud-architects-in-demand-and-well-paid/> [Accessed: 21 January 2020]
- Computing at School (2019) NCCE News 2019 [online]. Available from: https://www.computingatschool.org.uk/custom_pages/393-ncce-news-2019 [Accessed 30 January 2020]
- Crick, T. (2017). Computing education: An overview of research in the field. *London: Royal Society* [online]. <https://royalsociety.org/-/media/policy/projects/computing-education/literature-review-overview-research-field.pdf> [Accessed 25 October 2023]
- Curzon, P., McOwen, P., Cutts, Q., Bell, T. (2009). Enthusing and inspiring with reusable kinaesthetic activities. Proceedings of the ITICSE 2009.
- Dehouck, R. (2015) The maturity of visual programming. 29 September 2015. *craft ai*. [online] Available from: <https://www.craft.ai/blog/the-maturity-of-visual-programming/> [Accessed 23 October 2016]
- Department for Education (2013a) *Consultation Report: Changing ICT to Computing in the National Curriculum* [online]. Available from: https://assets.publishing.service.gov.uk/government/uploads/system/uploads/attachment_data/file/193838/CONSULTATION_REPORT_CHANGING_ICT_TO_COMPUTING_IN_THE_NATIONAL_CURRICULUM.pdf [Accessed 20 March 2020]
- Department for Education (2013b) *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 19 October 2016]
- Department for Education (2013c) *The national curriculum in England. Framework document* [online]. Available from: https://www.gov.uk/government/uploads/system/uploads/attachment_data/file/210969/NC_framework_document_-_FINAL.pdf [Accessed 13 October 2013]
- Department for Education (2014) *National curriculum and assessment from September 2014: information for schools* [online]. Available from: https://www.gov.uk/government/uploads/system/uploads/attachment_data/file/358070/NC_assessment_qualifications_factsheet_September_update.pdf [Accessed 10 November 2016]
- Department for Education and The Rt Hon Michael Gove MP (2012) *'Harmful' ICT curriculum set to be dropped to make way for rigorous computer science* [online]. Available from: <https://www.gov.uk/government/news/harmful-ict-curriculum-set-to-be-dropped-to-make-way-for-rigorous-computer-science> [Accessed 11 October 2016]

Dewey, J. (1938). *Experiential education*. New York: Collier Books.

Diethelm, I., Hubwieser, P., Klaus, R. (2012). *Students, teachers and phenomena: Educational reconstruction for computer science education*. Koli, Finland: ACM.

Dohan, M., Al-Mamoori, H. (2016) *Educational Support tool framework for student to learn coding*. BSc dissertation. The University of Northampton.

Eckert, J. (2018) Duckie. [online] Available from: <https://www.jeckert.com/duckie/> [Accessed 25 June 2018]

Edinburgh International Television Festival (2011) EdinburghTVFest - Eric Schmidt delivers the James MacTaggart lecture. *YouTube* [online]. Available from: <https://www.youtube.com/watch?v=hSzEFsf9Ao> [Accessed 6 November 2016]

Forbes (2019) *The World's Billionaire List* [online] Available from: <https://www.forbes.com/billionaires/list/> [Accessed 10 October 2019]

G2 Crowd (2019) The Top 20 Integrated Development Environment (IDE) Software [online] Available from: https://www.g2.com/categories/integrated-development-environment-ide#highest_rated [Accessed 25 February 2019]

Google (2017) Blockly I Google Developers [online]. Available from: <https://developers.google.com/blockly/> [Accessed 12 May 2017]

Greenfoot (2014) Teach & Learn Java Programming [online] Available from: <http://www.greenfoot.org> [Accessed 9 May 2014]

Grover, S., Pea, R. (2013). Computational thinking in K–12: A review of the state of the field. *Educational Researcher*, 42(1), 38–43. <https://doi.org/10.3102/0013189X12463051>

Guo, P. (2017) *Building Tools to Help Students Learn to Program*. *Communications of the ACM*, Vol. 60 No. 12, Pages 8-9. [online] Available from: <https://cacm.acm.org/magazines/2017/12/223041-building-tools-to-help-students-learn-to-program/fulltext> [Accessed 17 December 2017]

Guo, P., Warner, J. (2017) *CodePilot: Scaffolding End-to-End Collaborative Software Development for Novice Programmers* [online] Available from: https://bid.berkeley.edu/files/papers/CodePilot-collaborative-programming-IDE_CHI-2017.pdf [Accessed 17 December 2017]

Hankin, S. (2017) How Students 'Rubber Duck Debug' with Screencastify. 11 December 2017. *Screencastify*. [online] Available from: <https://www.screencastify.com/blog/rubber-duck-debugging-screencastify/> [Accessed 25 June 2018]

Hawksey, M. (2012) Twitter Archiving Google Spreadsheet TAGS v3. 11 January 2012. *MASHe*. [online]. Available from: <https://mashe.hawksey.info/2012/01/twitter-archive-tagsv3/> [Accessed 26 September 2018]

- Hawksey, M. (2018) TAGSExplorer. *MASHe*. [online]. Available from: <https://mashe.hawksey.info/category/tagsexplorer/> [Accessed 26 September 2018]
- Hayes, D. B. (2019) Rubber Duck Debugging: The Psychology of How it Works. 9 January 2019. *Thoughtful Code*. [online] Available from: <https://www.thoughtfulcode.com/rubber-duck-debugging-psychology/> [Accessed 23 January 2019]
- Henson, M. (2019) The National Centre for Computing Education website beta is live! 22 January 2019. *Teach Computing*. [online] Available from: <https://blog.teachcomputing.org/the-national-centre-for-computing-education-website-beta-is-live/> [Accessed 23 February 2020]
- HESA (2018) Destinations of Leavers from Higher Education 2016/17 [online]. Available from: <https://www.hesa.ac.uk/news/19-07-2018/DLHE-publication-201617> [Accessed: 29 March 2019]
- History.com Staff (2011) *Invention of the PC* [online]. A+E Networks. <http://www.history.com/topics/inventions/invention-of-the-pc> [Accessed 30 October 2016]
- Homer, M. and Noble, J. (2014) Combining Tiled and Textual Views of Code. In: *2014 Second IEEE Working Conference on Software Visualization (VISSOFT)*, Victoria, BC, Canada, 2014, pp. 1-10.
- Hope, A., Livingstone I. (2011) *Next Gen. Transforming the UK into the world's leading talent hub for the video games and visual effects industries* [online]. Available from: https://media.nesta.org.uk/documents/next_gen_wv.pdf [Accessed 23 March 2014]
- House of Commons Science and Technology Committee (2016) *Digital skills crisis. Second report of Session 2016-17* [online]. House of Commons. <https://www.publications.parliament.uk/pa/cm201617/cmselect/cmsctech/270/270.pdf> [Accessed 22 May 2017]
- House of Commons Science and Technology Committee (2017) *Digital skills crisis: Government Response to the Committee's Second Report of Session 2016–17* [online]. House of Commons. <https://publications.parliament.uk/pa/cm201617/cmselect/cmsctech/936/936.pdf> [Accessed 26 May 2017]
- Humphreys, S. (2016) #CASchat. 23 November 2016. *Computing at School* [online]. Available from: https://www.computingatschool.org.uk/news_items/247 [Accessed 28 December 2018]
- Hundhausen, C. D., Olivares, D. M., Carter, A. S. (2017). IDE-based learning analytics for computing education: a process model, critical review, and research agenda. *ACM Transactions on Computing Education (TOCE)*, 17(3), 1-26.
- Hunt, A., Thomas, D. (1999) *The Pragmatic Programmer. From Journeyman to Master*. USA: Addison–Wesley.
- Kafai, Y. B., Burke, Q. (2014). *Connected code why children need to learn programming*. Cambridge MA: MIT Press.

Kemp, P. (2014) *Computing in the national curriculum. A guide for secondary teachers*. [online] Computing at School. Available from: https://www.computingatschool.org.uk/data/uploads/cas_secondary.pdf [Accessed 5 May 2016]

Kodu. Game Lab Community (2014) [online] Available from: <http://www.kodugamelab.com/> [Accessed 9 May 2014]

Kölling, M., Brown, N. C. C., Altadmri, A. (2017) 'Frame-Based Editing' *Journal of Visual Languages and Sentient Systems*, vol. 3. Available from: https://kclpure.kcl.ac.uk/portal/files/71018111/Frame_based_editing.pdf [Accessed 21 January 2018]

Kölling, M., Brown, N. C.C. & Altadmri, A. (2015) Frame-Based Editing: Easing the Transition from Blocks to Text-Based Programming. In: *WIPSCCE '15 Proceedings of the Workshop in Primary and Secondary Computing Education*, 9-11 November 2015. London: ACM

Lee, I., Martin, F., Denner, J., Coulter, B., Allan, W., Erickson, J., Malyn-Smith, J., Werner, L. (2011) Computational thinking for youth in practice. *ACM Inroads*. 2 (1), 32-37

Lego (2018) Mindstorms [online]. Available from: <https://www.lego.com/en-gb/mindstorms> [Accessed 13 September 2018]

Lister, R. (2011) Concrete and other neo-Piagetian forms of reasoning in the novice programmer. In: *ACE '11 Proceedings of the Thirteenth Australasian Computing Education Conference, 17 January 2011*. Perth: Australian Computer Society, Inc. Darlinghurst

Lister, R., Adams, E. S., Fitzgerald, S., Fone, W., Hamer, J., Lindholm, M., Thomas, L. (2004). A multi-national study of reading and tracing skills in novice programmers. Working Group Reports from ITICSE on Innovation and Technology in Computer Science Education, Leeds, United Kingdom. 119-150.

Logo Foundation (2015) *What is Logo?* [online]. Available from: http://el.media.mit.edu/logo-foundation/what_is_logo/index.html [Accessed 25 May 2015]

Lopez, M., Whalley, J., Robbins, P., & Lister, R. (2008). Relationships between reading, tracing and writing skills in introductory programming. Proceedings of the Fourth International Workshop on Computing Education Research, Sydney, Australia. 101-112. doi:10.1145/1404520.1404531

Lowe, J. (2016) EduBlocks [online]. Available from: <https://edublocks.org/> [Accessed 16 January 2020]

Massachusetts Institute of Technology (2013) Dropbox Founders Drew Houston '05 and Arash Ferdowsi '08. MIT EECS Connector. 57-59 [online]. Available from: <http://www.eecs.mit.edu/docs/newsletter/connector2013.pdf> [Accessed 12 October 2017]

Micro:bit Educational Foundation (2023) Micro:bit Educational Foundation | micro:bit [online]. Available from: <https://microbit.org/> [Accessed 15 November 2023]

National Careers Service (2017) *Primary school teacher* [online]. Available from: <https://nationalcareersservice.direct.gov.uk/job-profiles/primary-school-teacher> [Accessed 18 April 2017]

National Centre for Computing Education (2019) Teach Computing [online]. Available from: <https://teachcomputing.org/> [Accessed: 11 December 2019]

Neoreason, Inc. (2019) *Repl.it* [online]. Available from: <https://repl.it/> [Accessed 26 June 2019]

Nishida, T., Kanemune, S., Idosaka, Y., Namiki, M., Bell, T., Kuno, Y. (2009). A CS unplugged design pattern. Proceedings of the 40th ACM Technical Symposium on Computer Science Education (SIGCSE '09), 231. doi:10.1145/1508865.1508951

Nolen, J.L. (2017) Paul Allen [online]. Encyclopædia Britannica, inc. Available from: <https://www.britannica.com/biography/Paul-Allen> [Accessed 11 October 2017].

Norbury, C. (2016) *BLOG: Policies of the Past: Computer Literacy Project* [online]. Creative England. Available from: <http://www.creativeengland.co.uk/story/blog-policies-of-the-past-computer-literacy-project> [Accessed 10 November 2016]

Office of Qualifications and Examinations Regulation (2018) *Decisions on GCSE computer science assessments regulated by Ofqual* [online]. Available from: https://assets.publishing.service.gov.uk/government/uploads/system/uploads/attachment_data/file/673216/Decisions_on_GCSE_computer_science_assessments_regulated_by_Ofqual.pdf [Accessed 21 January 2019]

Papert, S. (1993). *The children's machine: Rethinking school in the age of the computer*. Basic Books.

Patel, J. (2017) *The 9 Most In-Demand Programming Languages of 2017* [online] Coding Dojo Blog. Available from: <http://www.codingdojo.com/blog/9-most-in-demand-programming-languages-of-2017/> [Accessed 10 April 2017]

Pavlenko, V. (2019) *Teach Python 3 and web design with 200+ exercises* [online] Available from: <https://snakify.org/en/> [Accessed 15 June 2019]

Piaget, J. (1950). *The psychology of intelligence*. Cambridge, MA: Harvard University Press.

Poblocki, T. (2019) *IDEaI* [Software]. Windows. Available from: <https://github.com/PobTom/IDEaI> [Accessed 7 February 2020]

Powers, K., Ecott, S. & Hirshfield, L.M. (2007) Through the looking glass: Teaching CS0 with Alice. in *SIGCSE 2007: 38th SIGCSE Technical Symposium on computer science Education*. pp. 213-217, SIGCSE 2007: 38th SIGCSE Technical Symposium on computer science Education, Covington, KY, United States, 3/7/07 [online]. Available from: <https://dl.acm.org/citation.cfm?doid=1227310.1227386> [Accessed 23 May 2017]

Price, T.W., Brown, N. C. C., Dragan, L., Barnes, T. & Kölling, M. (2016) Evaluation of a Frame-based Programming Editor In: *ICER '16 Proceedings of the 2016 ACM Conference on International Computing Education Research*. Melbourne: ACM. Available from: <http://twistedsquare.com/Frame-Evaluation.pdf> [Accessed 21 January 2018]

Python Software Foundation. (2019) *IDLE — Python 3.8.0 documentation* [online] Available from: <https://docs.python.org/3/library/idle.html> [Accessed 24 June 2019]

Robertson, A. (2018) Exams ‘useless’ for computer science, say experts. 13 November 2018. *Schools Week* [online]. Available from: <https://schoolsweek.co.uk/exams-useless-for-computer-science-say-experts/> [Accessed 30 January 2019]

Rodger, S. H. (2014) *Adventures in Alice Programming* [online] Available from: <http://www.cs.duke.edu/csed/alice/aliceInSchools/> [Accessed: 29 April 2014]

Rodriguez, S. (2018) *Former Facebook engineer quit to build the programming tool he always wanted* [online] Available from: <https://www.cnbc.com/2018/10/22/andreessen-horowitz-leads-4point5-million-seed-round-in-replit.html> [Accessed 29 June 2019]

Roffey, C. (2019) Why we need a new IDE for secondary schools. *Hello World*. No. 8, p. 72-73.

Rossum, G. van, 16th Nov 1998. IDLE 0.1 -- a Python IDE. *Linux Weekly News* [online] Available from: <https://lwn.net/1998/1119/idle.html> [Accessed 24 June 2019]

Rubber Duck Debugging (2017) *Rubber Duck Debugging – Debugging software with a rubber ducky*. [online] Available from: <https://rubberduckdebugging.com/> [Accessed 11 September 2018]

ScratchED (2014) [online] Available from: <http://scratched.media.mit.edu/> [Accessed 9 May 2014]

Sentance, S., Csizmadia, A. (2015, June). Teachers’ perspectives on successful strategies for teaching Computing in school. In *IFIP TCS*. <https://www.computingatschool.org.uk/media/ugwpungo/teachersperspectivesifipwithauthordetai1sv2.pdf> [Accessed 15 September 2023]

Sentance, S., Dorling, M., McNicol, A. (2013). Computer science in secondary schools in the UK: Ways to empower teachers. In *Informatics in Schools. Sustainable Informatics Education for Pupils of all Ages: 6th International Conference on Informatics in Schools: Situation, Evolution, and Perspectives, ISSEP 2013, Oldenburg, Germany, February 26–March 2, 2013. Proceedings 6* (pp. 15-30). Springer Berlin Heidelberg.

Sentance, S., Waite, J. (2017) PRIMM: Exploring pedagogical approaches for teaching text-based programming in school. In: *WiPSCe '17 Proceedings of the 12th Workshop on Primary and Secondary Computing Education, 2017*. Nijmegen, Netherlands: ACM.

Sentance, S., Waite, J., Kallia, M. (2019) Teaching computer programming with PRIMM: a sociocultural perspective. *Computer Science Education* [online], vol. **29** (2-3), p. 136-176. Available from: <https://www.tandfonline.com/doi/abs/10.1080/08993408.2019.1608781> [Accessed 5 March 2020]

Snap! (2015) [online] Available from: <http://snap.berkeley.edu/about.html> [Accessed 20 June 2017]

Stack Overflow (2018) *Stack Overflow Developer Survey Results 2018* [online] Available from: <https://insights.stackoverflow.com/survey/2018> [Accessed 10 February 2019]

Stack Overflow (2019) *Stack Overflow Developer Survey Results 2019* [online] Available from: <https://insights.stackoverflow.com/survey/2019> [Accessed 10 February 2019]

The Headmasters' & Headmistresses' Conference (2016) *The British Education System* [online] Available from: <http://www.hmc.org.uk/about-hmc/projects/the-british-education-system/> [Accessed 3 November 2016]

The Royal Society (2012) *Shut down or restart? The way forward for computing in UK schools* [online]. Available from: https://royalsociety.org/~media/Royal_Society_Content/education/policy/computing-in-schools/2012-01-12-Computing-in-Schools.pdf [Accessed 15 March 2014]

The Royal Society (2017) *After the reboot: computing education in UK schools* [online]. Available from: <https://royalsociety.org/~media/policy/projects/computing-education/computing-education-report.pdf> [Accessed 18 November 2017]

The Royal Society. (2018). *After the reboot: computing education in UK schools* [online]. <https://royalsociety.org/~media/events/2018/11/computing-education-1-year-on/after-the-reboot-report.pdf> [Accessed 30 August 2023]

Toyon, M. A. S. (2021). Explanatory sequential design of mixed methods research: Phases and challenges. *International Journal of Research in Business and Social Science* (2147-4478), 10(5), 253-260.

Trifonova, V. (2018) How Device Usage Changed in 2018 and What it Means for 2019. 20 November 2018. *GlobalWebIndex*. [online] Available from: <https://blog.globalwebindex.com/trends/device-usage-2019/> [Accessed 11 February 2020]

UK Government (2012) *Education System in the UK* [online] Available from: https://www.gov.uk/government/uploads/system/uploads/attachment_data/file/219167/v01-2012ukes.pdf [Accessed 21 October 2016]

UK Government (2016) *The national curriculum* [online] Available from: <https://www.gov.uk/national-curriculum> [Accessed 25 October 2016]

Van Gorp, M. J., & Grissom, S. (2001). *An empirical evaluation of using constructive classroom activities to teach introductory programming*. *Computer Science Education*, 11(3), 247-260.

Vargas, J.A. (2010) The Face of Facebook. Mark Zuckerberg Opens up. [online]. *The New Yorker*. Available from: <https://www.newyorker.com/magazine/2010/09/20/the-face-of-facebook> [Accessed 11 October 2017].

Vygotsky, L. S. (1978) *Mind in society: The development of higher psychological processes*. Cambridge, Mass.: Harvard University Press.

Walker, A. (2018) *What is an IDE: Integrated Development Environments for Beginners in 2019*. [online] Available from: <https://learn.g2crowd.com/ide> [Accessed 11 January 2019]

Weintrop, D., Wilensky, U. (2017) Comparing Block-Based and Text-Based Programming in High School computer science Classrooms. *ACM Transactions on Computing Education*, **18** (1).

Whittaker, F. (2018) Ofqual: 'Not possible' to use non-exam assessment in computer science. 5 November 2018. *Schools Week* [online]. Available from: <https://schoolsweek.co.uk/ofqual-not-possible-to-use-non-exam-assessment-in-computer-science> [Accessed 30 January 2019]

Wing, J. M. (2006) Computational thinking. *Communications of the ACM*, **49** (3), 33-35.

Wohl, B. (2017) Coding the curriculum: new computer science GCSE fails to make the grade. 21 June 2017. *The Conversation*. [online]. Available from: <http://theconversation.com/coding-the-curriculum-new-computer-science-gcse-fails-to-make-the-grade-79780> [Accessed 26 February 2018].

Woollard, J. (2018) Non Examined Assessment - a positive future? 10 September 2018. *Computing at School* [online]. Available from: https://www.computingatschool.org.uk/news_items/726 [Accessed 21 January 2019]