A FLOWCHART-BASED INTELLIGENT TUTORING SYSTEM MODEL TO IMPROVE STUDENTS' PROBLEM-SOLVING SKILLS

DANIAL HOOSHYAR

FACULTY OF COMPUTER SCIENCE AND INFORMATION TECHNOLOGY
UNIVERSITY OF MALAYA
KUALA LUMPUR

2016

A FLOWCHART-BASED INTELLIGENT TUTORING SYSTEM MODEL TO IMPROVE STUDENTS' PROBLEM-SOLVING SKILLS

DANIAL HOOSHYAR

THESIS SUBMITTED IN FULFILMENT OF THE REQUIREMENTS FOR THE DEGREE OF DOCTOR OF PHILOSOPHY

FACULTY OF COMPUTER SCIENCE AND INFORMATION TECHNOLOGY
UNIVERSITY OF MALAYA
KUALA LUMPUR

2016

ABSTRACT

Many students fail to succeed in programming courses or face difficulties. Lack of problem-solving skills is one of the most important factors contributing to this challenge. Several researchers believe that forming accurate mental models may yield improvement in novice programmers' problem-solving skills and should thus be a key goal of any introductory programming course. The flowchart has always been deemed ideal in forming accurate mental models of imperative programming concepts. Another concern is the lack of assistance when students encounter problems, which may lead to demotivation. In order to address this concern, one-to-one tutoring provided by an Intelligent Tutoring System (ITS) is known to be effective.

Although numerous ITSs have been developed for the programming field, none are designed to enhance problem-solving skills of novice programmers by focusing less on language and syntax and more on solution designing activities in the shape of flowchart development. Hence, the goal is to address the aforementioned gaps in this thesis by developing and evaluating a novel Flowchart-based Intelligent Tutoring System model (FITS) to produce improvement in students' problem-solving abilities and help them learn basic and imperative computer programing concepts.

The decision-making process in FITS is managed by a Bayesian network to handle uncertainty based on the probability theory. Additionally, an online formative assessment game called Tic-tac-toe Quiz for Single Players (TRIS-Q-SP) is incorporated into FITS to promote student motivation in case timely guidance and interaction are deficient. Unlike other existing ITSs related to computer programming, FITS not only promotes the idea of navigating online learning materials and updating the Bayesian network by applying an online game-based formative assessment, it also offers an adaptive and personalized flowchart development environment. The aim of

FITS is to improve problem-solving ability besides suggest learning goals along with appropriate reading sequences to students. Therefore, FITS can offer students an accurate mental model of execution, as it visualizes the solution development for a programming problem by converting the given problem statement into a relevant flowchart while actively engaging users in the process. Since a flowchart-based multi-agent system and an online formative assessment game are incorporated into the domain model and student model of the proposed ITS, FITS contributes to two different components of intelligent tutoring systems. FITS also expands and improves on many existing ITSs aimed at teaching programming.

At the end of the study, the prototype of FITS was evaluated by university students. According to the results, students who used FITS showed higher scores for the post-test than the pre-test with a learning gain of 60% compared to 36%. A two-tailed paired t-test with a 95% confidence interval was performed against the null hypothesis. The p-value of two-tailed paired t-test of 0.000 was obtained, showing strong evidence against the null hypothesis. Therefore, from the result of this t-test, it can be concluded that the scores in the post-test are significantly higher from the scores in the pre-test and the use of FITS in practice is supported. The students' opinions about FITS were collected via questionnaires and the results signified that the students really liked FITS, the online game and the personalized flowchart development environment as a learning approach.

ABSTRAK

Ramai pelajar yang gagal untuk cemerlang dalam kursus pengaturcaraan atau menghadapi banyak kepayahan. Kelemahan dalam kemahiran penyelesaian masalah adalah satu faktor paling penting yang menyumbang terhadap cabaran ini. Sebilangan penyelidik percaya yang pembinaan model mental yang tepat boleh membantu dalam peningkatan kemahiran penyelesaian masalah dan sepatutnya menjadi matlamat utama dalam kursus awal pengaturcaraan. Carta alir telah membantu dalam pembentukan model mental yang tepat bagi konsep pengaturcaraan imperatif. Satu lagi kerisauan ialah kekurangan bantuan apabila pelajar menghadapi masalah yang boleh menghambat motivasi. Untuk menangani kerisauan ini, bimbingan satu untuk-satu yang disediakan oleh Sistem Bimbingan Cerdas (SBC) diketahui sangat berkesan.

Walaupun banyak SBC telah dbangunkan dalam bidang pengaturcaraan, tiada yang direkabentuk khusus yang kurang menekankan sintak dan bahasa dan lebih tertumpu kepada merekabentuk penyelesaian dalam proses pembangunan carta-alir dengan bermatlamatkan meningkatkan kemahiran penyelesaian masalah. Oleh itu, matlamat kajian ialah menangani kekurangan yang ada dengan membangunkan satu model Sistem Bimbingan Cerdas berasaskan Carta-alir (FITS) untuk meningkatkan kebolehan penyelesaian masalah dan membantu dalam mempelajari konsep pengaturcaraan imperatif.

Proses membuat keputusan dalam FITS diuruskan oleh rangkaian Bayesian untuk mengendalikan ketidaktentuan berdasarkan teori kebarangkalian. Tambahan pula, satu permainan pengujian formatif dalam talian dipanggil Kuiz Tic-tac-toe untuk pemain tunggal dimasukkan dalam FITS untuk meningkatkan motivasi pelajar dalam kes sekiranya terdapat kekurangan bimbingan dan interaksi. Berlainan dengan SBC yang berkaitan dengan pengaturcaraan komputer sedia ada, FITS bukan saja menggalakkan

navigasi bahan pengajaran dan sentiasa mengemaskini rangkaian Bayesian dengan pengujian formatif berasaskan permainan, ia juga menawarkan persekitaran pembangunan carta-alir yang anjal dan persendirian. Tujuan FITS ialah meningkatkan kebolehan penyelesaian masalah di samping mencadangkan matlamat pembelajaran beserta dengan bahan bacaan yang sesuai kepada pelajar. Oleh itu, FITS menawarkan pelajar model mental yang tepat, dan ia mengambarkan pembangunan penyelesaian bagi sesuatu masalah pengaturcaraan dengan penukaran pernyataan masalah kepada carta-alir yang berpadanan serta berinteraksi secara aktif dengan pelajar. Oleh kerana sistem multi-agen berdasarkan carta-alir dan permainan pengujian formatif dalam talian dimasukkan dalam model domain dan pelajar , FITS menyumbang kepada dua komponen SBC. FITS juga mengembangkan dan meningkatkan SBC sedia ada yang tertumpu kepada pengajaran pengaturcaraan.

Di akhir kajian, satu prototaip FITS telah dinilai oleh pelajar-pelajar universiti. Hasil penilaian pelajar yang menggunakan FITS menunjukkan yang keputusan pelajar selepas penggunaan FITS adalah lebih tinggi dibandingkan dengan sebelum penggunaan FITS dengan dapatan pembelajaran 60% dibandingkan dengan hanya 36% bagi pelajar-pelajar yang tidak menggunakan FITS. Ujian t- satu-hujung berpasang dengan 95% *selang kenyakinan* dijalankan untuk menyanggah hipotesis null. P-value dengan nilai 0.0 diperolehi, menunjukkan bukti kukuh menyanggah hipotesis null. Oleh itu, keputusan t-test dengan keputusan post-test lebih tinggi dari pre-test di sokong dengan baik. Maklumbalas pelajar di kumpulkan melalui borang soal-selidik dan keputusannya menunjukkan yang pelajar sangat menggemari FITS, permainan dalam talian serta persekitaran pembangunan carta-alir persendirian sebagai satu kaedah pembelajaran.

ACKNOWLEDGEMENTS

First and foremost, I would like to express my sincere gratitude to my principal supervisor Assoc. Prof. Dr. Rodina Ahmad for all the support and encouragement given during the course of my PhD studies. You really helped me to move forward in my career and my life at a time when I felt that all my previous efforts were fruitless. I would also like to thank my co-supervisor Dr. Ram Gopal Raj for his assistance.

I would also like to thank all the staff at the Faculty of Computer Science and Information Technology and the University of Malaya for providing me with the infrastructure to complete this project as well as helping me through my studies.

Last but not least, I extend my heartfelt appreciation to my parents who stood by me through the triumphs and defeats of the past three years. Your love and support contributed in no small amount to the completion of this thesis.

TABLE OF CONTENTS

Abstract iii

Abstrak v

Acknowledgements vii

Table of Contents viii

List of Figures xiv

List of Tables xvii

List of Appendices xix

CHAPTER 1: INTRODUCTION 1

1.1 Background 1

1.1.1 Context 1

1.1.2 Problem Statement 3

1.2 Motivation 6

1.3 Aims and Objectives 7

1.4 Significance, Methodology and Scope 9

1.5 Contribution 11

1.6 Thesis Outline 13

CHAPTER 2: LITERATURE REVIEW 14

2.1 Intelligent Tutoring System and Its Key Components 14

2.1.1 The Expert Knowledge Module 15

2.1.2 The Student Module 16

2.1.2.1 Student modeling using Bayesian Networks 17

2.1.3 The Teaching Module 18

2.1.4 The Communication Module 18

2.2 Intelligent Tutoring Systems in the Programming Domain 18
2.2.1 PROUST 19
2.2.2 The LISP Tutor 20
2.2.3 J-LATTE 22
2.2.4 OOPS 24
2.2.5 JITS: Java Intelligent Tutoring System 25
2.2.6 BITS 27
2.2.7 iList 27
2.2.8 JavaGuide 29
2.2.9 The PHP ITS 30
2.2.10 Discussion 31
2.3 Intelligent Tutoring Systems and Gamification 34
2.4 Problem-solving Skill in Computer Programming 36
2.5 Flowcharts -- Effective Visualization for Novice Programmers 44
2.6 Dynamic Flowcharts 47
2.7 Flowchart-based Learning Approach for Improving Students' Problem-Solving Skills 48
2.8 Discussion 49
2.9 Summary 51

CHAPTER 3: RESEARCH DESIGN AND SYSTEM ARCHITECTURE 52

3.1 Research Design 52
3.1.1 Methodology 52
3.1.1.1 Problem identification and motivation 53
3.1.1.2 Definition of the objectives for a solution 53
3.1.1.3 Design and development 54

3.1.1.4 Demonstration and evaluation 54
3.1.1.5 Communication 55
3.1.2 Research Design and Instruments 55
3.1.3 Procedure and Timeline 57
3.1.4 Participants 58
3.1.5 Ethics and Limitations 58
3.2 System Requirements 58
3.2.1 Functional Requirement 59
3.2.2 Non-Functional Requirement 59
3.3 System Architecture 60
3.3.1 Domain Module 60
3.3.2 Student Module 60
3.3.3 Teaching Module 61
3.3.4 Communication Module 61
3.4 Summary 62

CHAPTER 4: THE FLOWCHART-BASED LEARNING APPROACH 63

4.1 FMAS's Architecture 63
4.1.1 The First System Scenario 64
4.1.1.1 NLP agent 64
4.1.1.2 Key finder agent 65
4.1.1.3 Dictionary agent 65
4.1.1.4 Flowchart agent 66
4.1.1.5 System chat agent 70
4.1.1.6 Error detection agent 70
4.1.1.7 Crawler agent 70

4.1.2 The Second System Scenario 71
4.1.2.1 NLP agent 71
4.1.2.2 Key finder agent 72
4.1.2.3 Process orientation agent 72
4.1.2.4 Admin agent 73
4.2 Summary 73

CHAPTER 5: DESIGN AND IMPLEMENTATION 75
5.1 The Given Tasks 75
5.1.1 The Entrance Test 75
5.1.2 Navigational Support Task 76
5.1.3 Test Flowchart Development Task 76
5.2 The Architecture of FITS 77
5.2.1 FITS's Student Module 80
5.2.1.1 Determining the student current knowledge 80
5.2.1.2 Personalized learning 86
5.2.2 FITS's Knowledge Module 88
5.2.3 FITS's Communication Module 88
5.2.3.1 Login window 89
5.2.3.2 Pre-test and post-test window 90
5.2.3.3 FITS's main window 91
5.2.4 FITS's Teaching Module 93
5.2.4.1 Navigational support and tic-tac-toe game 93
5.2.4.2 Pre-requisite recommendations 96
5.2.4.3 Adaptive flowchart development 97
5.3 Continuing to Work at a Later Time 98

5.4 Implementation Details 100
5.5 Finding and Updating the Student Current Knowledge 104
5.6 Summary 105

CHAPTER 6: EVALUATION 106

6.1 The Evaluation Process 107
6.1.1 The Evaluation of the First Release of FITS 108
6.1.2 The Evaluation of the Second Release of FITS 110
6.2 The Effectiveness of FITS 111
6.2.1 The Degree of Correlation between the Length of Time with the Increase in Student Scores 116
6.2.2 The Degree of Correlation between the Number of Help Requests with the Increase in Student Scores 117
6.2.3 The Degree of Correlation between the Number of Times the Game Played with the Increase in Student Scores 120
6.2.4 The Degree of Correlation between the Number of Tasks Completed with the Increase in Student Scores 121
6.3 The Student Group that Gets the Highest Benefit from FITS 123
6.4 The Attractiveness of FITS when Learning Programming 128
6.4.1 Is FITS Easy-to-use? 129
6.4.2 Does FITS help the student to learn programming? 132
6.4.3 Is the learning approach in FITS attractive? 139
6.5 Does FITS focus on problem-solving and assist the improvement of problem-solving skill? 145
6.6 The Suggestion of using FITS for other Units 148
6.7 Summary 151

CHAPTER 7: CONCLUSIONS.....153

7.1 Research Summary.....153

7.2 Research Contribution.....154

7.3 Future Work.....157

References.....160

List of Publications and Papers Presented.....168

Appendix A: FITS Questionnaire.....170

Appendix B: Pre- and Post-Test Questions.....175

LIST OF FIGURES

Figure 1.1: Design science research methodology process model 11

Figure 2.1: Interactions among ITS modules 15

Figure 2.2: The simplicity of flowchart notation 45

Figure 3.1: Structural relationship among modules in FITS 62

Figure 4.1: FMAS architecture 64

Figure 4.2: Workspace provided by the toolbar sub-agent 67

Figure 4.3: Toolbar sub-agent workspace with brief feedback 68

Figure 4.4: Workspace provided by the guidance sub-agent 69

Figure 4.5: Workspace provided by the guidance sub-agent along with full flowchart from the Internet 69

Figure 4.6: Workspace of the guidance sub-agent with extra information added by the crawler agent 71

Figure 4.7: Workspace of the process orientation agent 73

Figure 5.1: FITS's architecture 78

Figure 5.2: The Entry page of FITS 79

Figure 5.3: All programming concepts in FITS 81

Figure 5.4: Learning materials and the game button 88

Figure 5.5: Login window 90

Figure 5.6: Window for pre- and post-test 91

Figure 5.7: FITS's entry page with a navigational menu, offering study goals 92

Figure 5.8: The TRIS-Q-SP 95

Figure 5.9: The random multiple-choice test question chosen from the database 95

Figure 5.10: The player gives the right answer 95

Figure 5.11: The player gives the wrong answer 96

Figure 5.12: A pre-request recommendation generated by FITS 97

Figure 5.13: Workspace offered to user A based on the taken profile 98

Figure 5.14: Workspace offered to user B for the same programming problem 98

Figure 5.15: Database diagram 100

Figure 5.16: Software architecture used in FITS 102

Figure 6.1: The evaluation process of FITS 107

Figure 6.2: Boxplot chart of pre- and post-test scores 116

Figure 6.3: Scatter graph of increase in score vs. number of help request 118

Figure 6.4: Scatter chart of number of task completed to the increase in score 123

Figure 6.5: The number of participants grouped by the programming background 124

Figure 6.6: The effect of the student programming background to the student score 128

Figure 6.7: The user interface is well-designed 130

Figure 6.8: It is easy to use FITS 131

Figure 6.9: Assistance from FITS is helpful 133

Figure 6.10: FITS is helpful and can increase the student score 134

Figure 6.11: Response time in FITS 135

Figure 6.12: The number of tasks in FITS 136

Figure 6.13: Overall, FITS helps the student learn to program 138

Figure 6.14: The flowchart-based approach is a good idea 140

Figure 6.15: Giving goal to be pursued is a good idea 141

Figure 6.16: Learning programming by involving in flowchart development is a good idea 142

Figure 6.17: Give information when the student is in trouble is a good idea 143

Figure 6.18: Incorporating the mini-game could raise students' motivation 144

Figure 6.19: The helpfulness of flowchart in visualizing solutions in students mind 146

Figure 6.20: The flowchart development of FITS is a good problem-solving aid.....147

Figure 6.21: The usefulness of flowchart development when designing problem solutions 148

Figure 6.22: FITS should be used to help students in Programming I.....149

Figure 6.23: FITS should be used for other programming units.....150

LIST OF TABLES

Table 2.1: Comparative analysis of different reviewed ITSs 31

Table 2.2: Comparative analysis of flowchart-based programming environments 40

Table 5.1: Conditional probability for node "“If_statement” 82

Table 5.2: Probability value for node " Variable" 84

Table 5.3: Conditional probability distribution value for node " Floating point numbers" 84

Table 5.4: Conditional probability distribution of node " Other operators " 86

Table 6.1: Participants’ comments of the first release of FITS 109

Table 6.2: The average student score in the pre-test, post-test, and increase in scores 112

Table 6.3: Correlation between week when each student did the pre-test and their pre-test score 113

Table 6.4: Correlation between week when each student did the post-test and their pre-test score 114

Table 6.5: Paired t-test of post-test and pre-test 115

Table 6.6: Correlation between length of time spent by students and their increase in score 116

Table 6.7: Correlation between number of help requests and the increase in score 117

Table 6.8: Correlation between the number of help requests and increase (without outliers) 119

Table 6.9: Correlation without outliers between the pre-test score and the number of help 119

Table 6.10: Correlation between numbers of time game played and increase in score 121

Table 6.11: Correlation between the number of tasks completed and the increase in score 122

Table 6.12: The students’ programming backgrounds and their pre-test scores 125

Table 6.13: The students’ programming background and the post-test score 126

Table 6.14: The students' programming background and the increase in score 127

Table 6.15: Students' comments about easy-to-use of FITS 131

Table 6.16: Correlation of opinions about the number of tasks to the actual number of tasks completed 137

Table 6.17: Students' comments about help provided by FITS 138

Table 6.18: Students' comments about the mini-game incorporated in FITS 144

Table 6.19: Students' comments about FITS suggestion to other programming units . 150

Table 6.20: Different hypothesis used to evaluate different aspects 151

LIST OF APPENDICES

Appendix A: FITS Questionnaire …..…….……..…170

Appendix B: Pre- and Post-Test Questions …….175

CHAPTER 1: INTRODUCTION

This chapter presents the background, motivation, aims and objectives, significance, methodology, and contributions of this research. The key concepts of Artificial Intelligence (AI) in education and the flowchart-based approach are briefly discussed in section 1.1, providing an overview of their roles in this research. An outline of the thesis structure concludes this chapter.

1.1 Background

1.1.1 Context

Recent advancements in multimedia, high-speed Internet connections, and computer-mediated communication and communities all have potential to generate revolutionary improvements in education. Numerous countries are striving to support this revolution and are learning how best to adopt these new technologies (Aleven & Ashley, 1997). An even larger revolution is approaching, which is based on the extensive use of Artificial Intelligence in educational technology (Koedinger, 2001). The two fundamental research bodies that support this paradigm shift originate from cognitive science and Artificial Intelligence (Aleven & Ashley, 1997). One reason for the increasing use of Artificial Intelligence in education is the fact that powerful computers are becoming very affordable. Another reason for this revolution is the scientific progress from two aspects. First, progress in Artificial Intelligence has led to a deeper understanding of how to represent knowledge, how to reason, and how to describe procedural knowledge (i.e., "how to" knowledge). Second, cognitive science research has contributed to better understanding of how people think, learn, and solve problems. There is a powerful synergy here. Cognitive scientists often use Artificial Intelligence techniques to build simulation models of cognitive processes, dependencies, and behaviors (Conati & Van Lehn, 1999). Artificial Intelligence scientists apply results from cognitive science to

guide their explorations and design software with more human-like characteristics. When applied to education, this synergy leads to combinations of software and activities that can help more students learn better. Intelligent Tutoring Systems (ITSs) are employed in numerous areas of education including mathematics, physics, cognitive skill development, and workplace simulations (Cabada, Barrón Estrada, González Hernández, & Oramas Bustillos, 2014; Koedinger, 2001; Li, Zhao, & Xu, 2015)**.** For instance, an ITS called the Pump Algebra Tutor (PAT), developed by Kenneth Koedinger of Carnegie Mellon University, demonstrated extremely positive results (Anderson, Corbett, Koedinger, & Pelletier, 1995). Many middle schools, high schools, and colleges around Europe and the United States use PAT, which is designed to teach students how to model real-life problem situations using algebraic representations. As modern mathematics focus more on creating models that can answer multiple questions and less on computing single answers, PAT is mainly designed to help students acquire and make sense of formal mathematical strategies and representations. As part of the Algebra I curriculum, both students and teachers have been enthusiastic to use PAT. Experimental study results showed drastic student achievement gains in the control groups, indicating 15-25% and 50-100% improvement on standardized basic skill tests and problem-solving assessment, and representation use respectively.

In summary, the purpose of this study is to design and construct a novel flowchart-based ITS model that will help students learn imperative, rudimentary concepts of computer programming effectively and to enhance the problem-solving ability of university students undertaking their first introductory courses. This proposed pioneering research is expected to impact the fields of cognitive science, Artificial Intelligence and education.

1.1.2 Problem Statement

Introductory programming learning causes difficulties for many students worldwide. Since there are several programming courses in fields like engineering and computer science, students in these fields should be able to do some programming (M. McCracken et al., 2001). High dropout and failure rates in initial programming courses are reported in literature (Soloway & Spohrer, 1988; Teague & Roe, 2009). For instance, as Carter and Jenkins indicated, in final year projects students mostly avoid programming because they are not able to program or do not believe they can(Carter & Jenkins, 1999). The main reason for such difficulties is the lack of problem-solving skills (Carlisle, Wilson, Humphries, & Hadfield, 2004; D. Hooshyar, Ahmad, Nasir, Shamshirband, & Horng, 2015; Jonassen, 2000; Ma, Ferguson, Roper, & Wood, 2011; Moser, 1997; Pillay, 2003; Pillay & Jugoo, 2005). Nonetheless, class size, motivation, programming language syntax, and students' background in science and mathematics are regarded as other reasons for difficulty. As learning to solve problems algorithmically leads to learning how to program, many scientists indicate that more focus should be directed toward problem-solving skills rather than programming language (Conati & Van Lehn, 1999; Scott, 2010).

Mental models are considered crucial to building understanding in programming. It has been shown that students who do not develop mental models or develop unviable ones, are significantly disadvantaged. The importance of such models to developing understanding was also emphasized by Winslow (1996). Students usually attempt to code solutions without a mental model of the solution to a problem because they are not provided with a cognitive or visual model in the learning context. Instructors mostly emphasize pseudocodes in conveying programming concepts. However, the program execution flow between the flowchart and program components cannot be addressed by pseudocodes. With the lack of a mental model of execution, students encounter serious

difficulties in understanding the relationships among individual programming components. Therefore, the formation of accurate mental models should be a key goal of an introductory programming course, as it could enhance the problem-solving skills of novice programmers and overcome the aforementioned issue in learning computer programming.

The flowchart has always been advocated as an effective visual aid to learning programming and the best in forming accurate mental models of imperative programming concepts (Carlisle et al., 2004; Scott, 2010; Swain, Moses, Anderson, & Davis, 2013). Flowcharts have a small learning curve and can be easily understood with little or no prior training. Focus is on the basic imperatives of sequence, selection and iteration, while emphasizing program composition and the flow of execution. Using visualization-based tools and environments can even extend the effectiveness of flowcharts to aid novices in problem-solving and program development (Bassat Levy, Ben-Ari, & Pekka, 2001). These visualization-based environments can also afford novice programmers a concrete model of execution that facilitates understanding algorithms and programming concepts. This will place greater emphasis on the underlying abstractions of programming, problem-solving and program composition, i.e. how the pieces fit together to form a solution to a problem or specification.

Besides, preventing novice programmers from involvement in statements of programming problem may cause serious difficulties upon encountering new programming exercises (Areias & Mendes, 2007). Novice programmers need to observe the relationship between the statement of programming problem and relevant flowcharts. This should enable novices to relate the statement of a programming problem to its corresponding flowchart more effectively. As such, novices could focus on overcoming conceptual difficulties and developing problem-solving skills whilst

minimizing the impact of a complex programming language. Therefore, an automatic text-to-flowchart conversion approach can be an effective visual representation that enables novice programmers to directly observe how a textual programming problem maps onto a flowchart. Such approach could significantly minimize syntax overhead and allow attention to problem-solving.

Another reason why students may not practice enough is the lack of mentors to help when they encounter problems. With generally a 30:1 student-teacher ratio, it is very difficult to be able to help students every time they encounter trouble in efforts to develop a solution for a programming problem. Some students will try to find help from the tutor or teacher during the next class, but others may simply stop trying.

Woolf (2008) described some principles of human learning that can make learning effective. To learn effectively (Rutherford, 1991), students must:

- be involved, engaged, and active in the learning process (Schank & Cleary, 1995).
- learn at their own learning pace (Vygotsky, 1978).
- learn material that is in accord with the state of their current knowledge (B. P. Woolf & Hall, 1995).

These requirements are very challenging to realize in a classroom environment with a 30 to 1 student-teacher ratio. One possible solution to applying the principles of human learning is to use an Intelligent Tutoring System (ITS). ITS research is a multidisciplinary field (Artificial Intelligent, Cognitive Science, and Education) that investigates means of devising educational systems that provide customized instruction tailored to students' needs (Conati, 2009). Developing an ITS for the programming

domain will enable students to learn material at their own learning pace. Material to be learned can also be tailored to match the condition of the students' current knowledge.

A number of ITSs have been developed in the area of computer programming to help students, for example PROUST (Johnson, 1990), LISP Tutor (Anderson et al., 1995), JITS (Sykes, 2007), BITS (Butz, Hua, & Maguire, 2008), J-LATTE (Holland, Mitrovic, & Martin, 2009), iList (Fossati, Di Eugenio, Brown, & Ohlsson, 2008), OOPS (Gálvez, Guzmán, & Conejo, 2009), JavaGuide (Hsiao, Sosnovsky, & Brusilovsky, 2010) and PHP ITS (Weragama & Reye, 2013). However, none of these ITSs incorporate an adaptive flowchart-based approach to engage novice programmers in a personalized flowchart development environment to minimize the overhead of syntax and focus on problem-solving. Therefore, the potency of exploiting the problem-solving ability of ITSs in the programming domain has not been fully explored. In addition, the aforementioned ITSs may cause students to lose motivation during the learning process when timely guidance is lacking. The best way to facilitate active learning and presentation in a context that is fun and engaging is game-based learning. Thus, incorporating an online game into the ITS with the aim of boosting interaction and student motivation during learning, especially when timely guidance is lacking and students may lose enthusiasm, could be an effective way to address this issue.

1.2 Motivation

The motivation of this research is summarized as follows:

- To develop a generic model to boost problem-solving skills for novice programmers and that can be adapted to different programming languages. This model will comprise new design approaches to create a flowchart-based intelligent tutoring system.

- To overcome the limitations of current ITSs in the computer programming area, which is more focus on programming language feature and syntax and less on problem-solving.
- To facilitate problem-solving skill improvement in programming for large numbers of learner populations around the world who are required to learn programming.

1.3 Aims and Objectives

The main goal of this research is to investigate whether a flowchart-based ITS model can help improve students' problem-solving skills and to learn basic computer programming concepts effectively. In order to achieve this, the system should be able to:

- Check the correctness of each student's solution, and provide feedback and help that fits the problems faced by students as they develop flowcharts. Moreover, the system should not only tell whether a student's program is correct or incorrect overall, but it should also be able to locate any errors in the student's program and provide help accordingly.
- Engage students in the process of flowchart development with the aim of improving their problem-solving skills. Therefore, the system should use approaches that will get students involved in the process of flowchart development (solution designing activities).
- Provide a system that requires minimal effort to use. Students should be able to use the ITS seamlessly within the development environment they are using to develop each solution. With this, students are expected not to exert significant additional effort when required to use the ITS.

- Provide pre-request recommendations based on the students' level of knowledge. Additionally, they will be informed of the percentages of their specific and total knowledge in each programming concept. Accordingly, the system should know whether the student needs to repeat a particular topic or continue to the next topic.
- Provide encouragement and enjoyment to the students to learn the material. Therefore, the system should employ an approach that is able to attract and encourage students to the learning materials.

The research questions addressed in this research are:

1) What is the best method of knowledge representation that can be used to model the subject matter necessary to effectively teach basic programming concepts while achieving the following?
   a. Involve students in flowchart development
   b. Provide feedback based on specific errors made by students
2) How to provide adaptive guidance that fits the students' current knowledge?
3) How to attract students to the learning material and raise motivation when they are demotivated?

In order to answer these questions and achieve the research goal, the present research objectives are listed below:

1) To review current techniques employed in existing ITSs for programming.
2) To design a flowchart-based model for an ITS with the aim of improving students' problem-solving skills in programming using Artificial Intelligence methods.

3) To construct a framework for developing a flowchart-based ITS with the following features:

- Design and implement a knowledge base that can be used for checking the correctness of the student's flowchart and offer feedback and help accordingly.
- Design and implement a student module that can be used to customize the tasks that must be performed by the student.
- Design and implement an ITS that integrates with the flowchart development environment used by the targeted students.
- Design and implement an online game into the ITS with the aim of improving interaction and raising student motivation during the learning process, especially when timely guidance is lacking and students' enthusiasm may diminish.

4) To evaluate the developed prototype in an educational environment.

1.4 Significance, Methodology and Scope

Research on Intelligent Tutoring Systems has been gaining momentum over the past few decades. Still, the ITS is not an extensively familiar concept to educators. One of the main reasons is that although many ITSs have been built, only few are used in practical teaching situations. This indicates there is significant room for improvement in the field of ITS. This research attempts to improve on existing ITSs, at least to a certain degree. Existing ITSs are employed for teaching in many different domains, from primary school reading to higher education programming and electronic circuit design. ITSs for teaching programming languages such as Pascal, Prolog, C and Java, focus mainly on developing console and Windows applications (Corbett, 2000; Jacqui Chetty & Barlow-Jones, 2015; Song, Hahn, Tak, & Kim, 1997; Sykes, 2007). However, none are designed to focus less on language and syntax and more on problem-solving skills by engaging novice programmers in the process of flowchart development. On the other hand, many computerized teaching systems that aim improving students' problem-

solving ability in computer programming are available (D. Hooshyar et al., 2015). However, they offer a flowchart development environment in the same way for each student, i.e. instruction is not individualized during flowchart development. The literature does not reveal any instances of the integration of these two ideas: i.e. ITSs that are designed to offer both an adaptive and personalized flowchart development environment. Therefore, this study addresses a domain that is completely new to ITS research. If the ITS is found effective and attractive, these findings will enable educators to consider this method in their efforts to teach programming effectively.

Currently, C++ is one of the most popular programming languages in the world (Stroustrup, 2000). Many of today's most widely used systems (e.g. Amazon, Amadeus, Facebook, Google, and Bloomberg) have critical parts written in C++. Hence, this study is in tune with the current trend and demands of the science and technology and education sectors of society. However, with the aim of simplifying the task of developing the proposed flowchart-based ITS, only elementary topics are covered, such as variables, assignments, and control structures, while more sophisticated topics like pointers and inheritance are excluded. In case of more complicated concepts, the instructional content and learning material can easily be modified without having to reformulate the entire system.

In terms of methodology, as this thesis involves a rigorous process of designing an artifact to solve observed problems, making research contributions and evaluating the design, as well as communicating the results to appropriate audiences, the science design research methodology (SDRM) is selected as the present research methodology. SDRM usually serves to create and evaluate IT artifacts aimed at solving identified organizational problems (Peffers, Tuunanen, Rothenberger, & Chatterjee, 2007). These artifacts may consist of algorithms, human/computer interfaces, design methodologies

(including process models) and languages. Its application can be found in several disciplines and fields, the most notable being engineering and computer science. SDRM comprises six steps: 1) problem identification and motivation, 2) definition of the objectives, 3) design and development, 4) demonstration, 5) evaluation, and 6) communication, as illustrated in Figure 1.1.

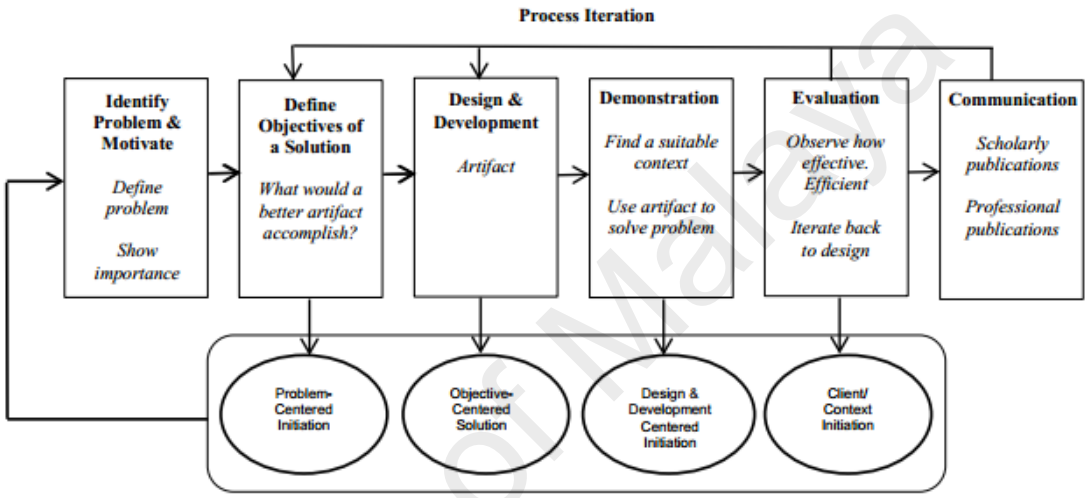

Figure 1.1: Design science research methodology process model

1.5 Contribution

This research contributes to the areas of 1) teaching and learning computer programming, as well as 2) intelligent tutoring system architecture.

Regarding the former, unlike other ITSs developed for computer programming, the proposed ITS offers an adaptive and personalized flowchart development environment with the aim of improving problem-solving ability. Because the proposed ITS helps visualize the solution development for a programming problem by converting the given problem statement into a relevant flowchart while engaging users in the process, it facilitates the creation of a mental model of execution for students. Regarding the latter, a flowchart-based multi-agent system capable of converting the statement of programming problem to a relevant algorithm and flowchart is improvised in the

domain module of ITS so as to involve students individually in the process of flowchart development with the goal of improving their problem-solving skills. The multi-agent system benefits from a novel text-to-flowchart conversion approach, which enables novices to observe the relationship between the textual forms of a programming problem and the relevant flowcharts. This approach is an effective visual representation that enables novice programmers to directly observe how a textual programming problem maps onto a flowchart. This could significantly minimize syntax overhead and assist with focusing on problem-solving.

Moreover, an online game is incorporated in the student model of ITS to raise student motivation and enthusiasm when timely guidance is lacking and students may become demotivated. This game can not only improve students' motivation, but it also uses their knowledge on the subject matter and get the Bayesian Network, used for the process of decision making in the proposed system, updated frequently without the students even realizing it. Thus, the current ITS contributes to two different components of intelligent tutoring systems, namely the domain module and the student module.

In sum, the contributions of this research are as follows:

- The first dynamic and adaptive environment is presented for flowchart development.
- This is a novel, first flowchart-based multi-agent system that benefits from an automatic text-to-flowchart conversion approach (improvised in the domain module of the proposed ITS). This multi-agent system can be utilized in many academic applications, for instance problem-solving (in different fields, e.g. mathematics and engineering), drawing diagrams, and more.

- This novel flowchart-based Intelligent Tutoring System model deals with problem-solving skill improvement in learning computer programming.
- Game-based learning is applied in the proposed ITS as a way to raise user motivation and concurrently obtain the user's profile to update the Bayesian network for the process of decision making.

1.6 Thesis Outline

Following this introduction (Chapter 1), the thesis continues with a literature review in Chapter 2. In Chapter 3, the research methodology and system architecture are described. The general ITS model and flowchart-based approach are illustrated in Chapter 4, while Chapter 5 explains the design and implementation of the proposed flowchart-based ITS in more detail. Chapter 6 discusses the evaluation of the proposed flowchart-based ITS and the results obtained from the evaluation. Finally, Chapter 7 summarizes the outcomes of this research. The appendices provide more detailed information on some aspects, such as the questions used for pre- and post-testing.

CHAPTER 2: LITERATURE REVIEW

This chapter starts with the overview of Intelligent Tutoring System (ITS) and continues by reviewing existing ITSs in the programming domain – the domain of this research; ITSs and Gamification are then briefly explained. The concept of problem-solving skill in programming is then explained along with some studies that have been done on improving problem-solving skill using flowchart. This chapter ends with a discussion of the shortcomings of the existing ITSs in the programming domain and the research work in this thesis that aim at addressing those shortcomings.

2.1 Intelligent Tutoring System and Its Key Components

Studies of student learning have long shown that learning can be more effective if it is performed through private tutoring rather than teaching in a classroom (Anderson & Reiser, 1985). In classroom teaching, all the students have to listen to the same lectures regardless of the knowledge that they have. It is not possible for the teacher to deliver a lecture that is customized to each student's existing knowledge. And when it comes to the homework, the students will mainly do the homework by themselves. The teacher may not be able to provide as much help as needed by the students. The problems of classroom teaching may not occur in private tutoring. In private tutoring, the tutor can provide material that fits with the students' current knowledge and provide as much help as needed by the students. However, providing as many tutors as the number of students is not feasible from an economic point of view. An Intelligent Tutoring System (ITS) is a solution that can be used to address this problem. ITSs are computer programs that are designed to work like tutors (Anderson & Reiser, 1985). The ITSs can teach the materials that relevant to the students' current knowledge and provide help that is specific to the students' problem (Woolf, 2008). Therefore ITSs enable private tutoring. There are four components that construct an ITS, shown in Figure 2.1 (adopted from

(Nwana, 1990)). The arrows in the figure represent the possible flow of information from one module to another. These components are: (i) the expert knowledge module or domain module, (ii) the student knowledge module, (iii) the teaching module, and (iv) the communication module (Nwana, 1990).

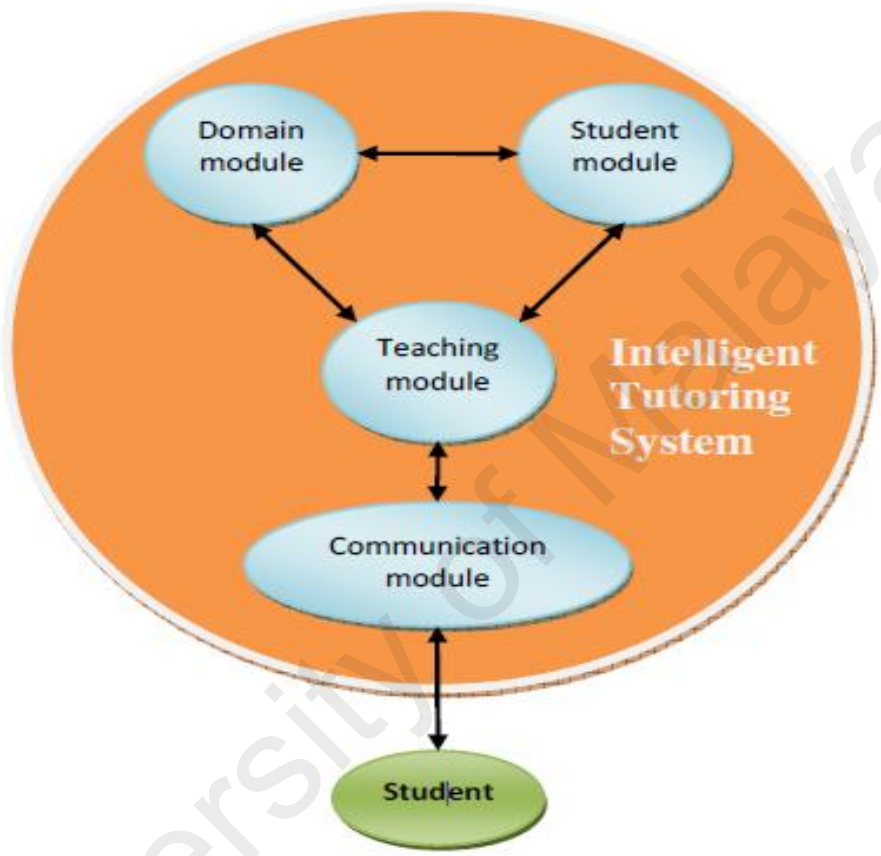

Figure 2.1: Interactions among ITS modules

2.1.1 The Expert Knowledge Module

The rules and fact of a particular domain that should be conveyed to the student are stored in the expert knowledge module. It must be specific and detailed and therefore a lot of effort has to be put in exploring the domain and codifying the knowledge. Besides of simply storing the domain knowledge, the expert module has the task to present the information, to solve exercises in the same context as the student does and compare

these solutions to the answer given by the student. The main tasks of the expert module are summarized in below. It should:

- select the content that is displayed by the communication module,
- select a tutoring strategy depending on the learning process,
- control and adjust the speed of tutoring actions,
- select and generate questions to check the learning progress,
- select and generate constructive feedback,
- provide assistances and additional information to deal with gaps in student's knowledge and take actions to guarantee student's motivation during instruction.

These tasks should be seen as a range of responsibilities the expert module can have. Different tutoring systems while implement some of the tasks better than others by omitting other tasks at the same time.

2.1.2 The Student Module

The student knowledge module is used to represents the student capabilities in the domain and other information about the student (e.g., the time spent on completing the task, help requested, etc.). It can be seen that the student module is an important component in such systems in order to individualize the interactions based on the characteristics of the students. Students are human beings who have many different traits such as knowledge levels, learning styles, motivation, likes and many more. All these traits contribute to their preferred methods of learning and should therefore theoretically be modeled in order to individualize the interaction. Without an explicit student model, the teaching strategies component is unable to make decisions to adapt instructional content and guidance and is forced to treat all students similarly. Student modeling presents well-known difficulties stemming from the fact that modeling the

student within an intelligent tutoring system involves a good deal of inherent uncertainty (Hafidi & Bensebaa, 2014). In the next sub-section the reason of using Bayesian Network in the student model of our proposed system has been explained.

2.1.2.1 Student modeling using Bayesian Networks

As one of the biggest challenges in designing ITSs is the effective assessment and representation of the student's knowledge state and specific needs in the problem domain based on uncertainty information, the task of dealing with the uncertainty management for the student model is thus challenging (Gertner, Conati, & VanLehn, 2006). Until the late 1980s, researchers interested in student modeling had only limited techniques for uncertainty management available, and they mostly had to rely either on poorly understood ad hoc techniques or on general techniques (Jameson, 1995). Fortunately, over the past decade the question of how to manage uncertainty has been a rapidly expanding and increasingly mainstream research topic in Artificial Intelligence. Various approaches in Artificial Intelligence have been proposed for uncertainty reasoning (Nilson, 1998), including rule-based systems (Buchanan & Shortliffe, 1984), fuzzy logic (Klir & Yuan, 1995), DempsterShafer theory of evidence (Zadeh, 1986), and neural networks. Bayesian networks (Pearl, 1988) are a powerful approach for uncertainty management in Artificial Intelligence (Wong & Butz, 2001; Wong, Butz, & Wu, 2000). Several ITSs used variations of the Bayesian modeling technique in their student modeling (Beck, Chang, Mostow, & Corbett, 2008; Butz et al., 2008; Mitrovic, 2003; Weragama & Reye, 2013). Overall, the reason for using this method so extensively is that it can accurately handle uncertainty. The theoretical basis for Bayesian Belief Networks (BBNs) is also highly developed and therefore it is expected to provide a relatively accurate student model, when sufficient observations are available.

2.1.3 The Teaching Module

The next module is the teaching module. This module uses the knowledge about the student to decide what pedagogic activities should be presented. Some pedagogic activities are hints, explanations, providing different tasks, etc. the presentation of a new topic, which topic to present, and how to guide and recommend them are some example of this module. As this component needs to take appropriate actions to manage one-on-one tutoring, namely the use of a variety of teaching approaches at the appropriate times and switch teaching strategies, the input of this component is the assessment result of the student model. Therefore, the system's pedagogical decisions reflect differing needs of students.

2.1.4 The Communication Module

The last module, the communication module is used to control the interaction between the student and the system. The ease of use and pleasant appearance of the system can become part of the components in attracting the student to use the system. The process in an ITS may start by searching for a particular problem in the domain module based on the student's current knowledge that is stored in the student module. Information from the student module is used by the teaching module to present the problem with an appropriate teaching strategy. The student sees and responds to this information through the communication module. The student solution is then analyzed to see if it solves the given problem. The outcome of this comparison can be used to update the student's knowledge in the student module.

2.2 Intelligent Tutoring Systems in the Programming Domain

Several ITSs have been developed in the programming domain. The following subsections describe: PROUST (Johnson, 1990), LISP Tutor (Anderson et al., 1995), JITS (Sykes, 2007), BITS (Butz et al., 2008), J-LATTE (Holland et al., 2009), iList

(Fossati et al., 2008), OOPS (Gálvez et al., 2009), JavaGuide (Hsiao et al., 2010) and PHP ITS (Weragama & Reye, 2013). PROUST and the LISP Tutor can be considered as two early ITSs in the programming domain. Each of them has different techniques to recognize the correctness of the student code to solve the given task. Another technique to recognize the student code can also be seen in a newer ITSs in programming domain such as in J-LATTE and the PHP ITS. This literature review also discusses JITS, although its technique can only be used to check the correctness of the syntax of the student code. The next two sections, on BITS and JavaGuide, discuss how these systems give the student the right material that they need. Finally the last two sections, on OOPS and iList, discuss adaptive learning and feedback.

2.2.1 PROUST

PROUST (Johnson, 1990) is well-known as one early application that tried to help students learn to program. PROUST is a system that was developed to help novice Pascal programmers. Students had to write Pascal programs (in any development environment) to solve two medium-level programming tasks defined in PROUST. These tasks are called the Rainfall Problem and the Bank Problem. After writing his/her program, the student can invoke PROUST to check if their program solves the given task. PROUST will only give the feedback if the student program does not have any syntax errors. To analyze the student program, PROUST makes a mapping between the program requirements and the student program. This mapping is expected to show the design and the implementation used by the programmer to solve the task. A knowledge base of programming plans and strategies, together with common bugs is used by this process. There are two types of programming plans that are used in PROUST, the variable plans and control plans. Variable plans are plans that generate a result which is usually stored in a variable, and control plans are plans that are used to regulate the generation of the result in variable plans.

The effectiveness of PROUST was tested by comparing the midterm exam results of two groups of students. Before the exam, the students were asked to write the program to solve the Rainfall Problem. This homework was due one week before the exam. When doing this homework, the students in the first group could access PROUST to get help and feedback, but the students in second group could not. The midterm examination itself required the students to identify and repair bugs in three programs. One buggy program was structurally identical to the Rainfall Problem. Johnson (1990) showed that the mean midterm exam score of the group that could access PROUST was higher than the group that could not access PROUST. He concluded that the students who succeeded in fixing bugs on their programming assignments were significantly better at repairing similar bugs in the midterm examination. Although PROUST was developed with the aim of helping students learn to program, it was never fully developed into an ITS. Consequently, PROUST can be considered more as a debugging aid to solve the Rainfall Problem and the Bank Problem (Sack, Soloway, & Weingrad, 1992).

2.2.2 The LISP Tutor

Another early application that also tried to help students learn to program is the LISP Tutor. It is renowned as the first application that uses Model Tracing as its approach. The LISP Tutor was created as an attempt to develop a computer-based tutor that is as effective as a human tutor in teaching LISP (Anderson et al., 1995). The main characteristic of Model Tracing tutoring approach is the existence of the production rules that represent the path or a series of path that lead to the solution state. Buggy rules are also used to show diversions from these paths. As the students write their code in the LISP Tutor, the system monitors the code entered and provides immediate feedback when the code is incorrect or ambiguous. In this way, the LISP Tutor directs

the code that must be written by the students in each step. Therefore, unlike a normal programmer, the students cannot write their program freely in the LISP Tutor.

Because the LISP tutor provides immediate feedback before the student finishes all of the code then sometimes the LISP Tutor need to confirm the intention of the code that the student has just written. For example, let's consider that the student has just finished writing a function parameter. With this very limited information, the immediate feedback feature of the LISP Tutor will not know the purpose of that parameter. In this case, the LISP Tutor will display some options that ask the student of his/her intention for the parameter written. When the student selects an option given, the LISP Tutor will display information regarding the chosen selection. If the student choose an incorrect answer then an explanation will be given and the student can choose another option until finally the correct option is chosen. The LISP Tutor also provides help in regards to code writing, such as right parenthesis balancing, expanding templates for function calls, and advancing the cursor over the remaining symbols that must be expanded. Unlike PROUST which contains only two exercises (Johnson, 1990), there were ten lessons included in the LISP Tutor. Each lesson involved a small instructional booklet and several tasks. Students needed one to four hour to complete each lesson. Because the instructional booklet is brief, most of the time in any lesson is expected to be used to solve the tasks given. The inclusion of these comprehensive materials enabled the LISP Tutor to teach a student step-by-step from having very basic knowledge to an advanced level. However the requirement for the student to follow each step defined in the LISP Tutor to reach the final solution makes the process of programming in the LISP Tutor less flexible than with PROUST.

2.2.3 J-LATTE

A renowned approach in tutoring system is Constraint Based Modeling (CBM). As a way to overcome problems with student modeling, the CBM approach was introduced by Ohlsson (1992). This approach was firstly used in SQL-Tutor, an ITS that is used to teach student about SQL (Mitrovic, 2003). In CBM approach, unlike the Model Tracing approach which stores the paths to the correct answers, the domain knowledge is represented as a set of state constraints. A constraint specifies certain conditions that must be satisfied by all correct solutions. A CBM tutoring system evaluates the student's program and produces a list of relevant, satisfied, and (possibly) violated constraints in order to check the correctness of the student's solution. The student's program is considered correct if there are no constraints violated by the student's code. Kodaganallur, Weitz, and Rosenthal (2005) called this a product-centric approach, compared to Model Tracing tutors which use a process-centric approach.

J-LATTE is one example of an ITS that uses Constraint Based Modeling (CBM) (Holland et al., 2009). J-LATTE was used to help students learn to program in Java. Each constraint in J-LATTE (and in any CBM tutoring system) is expressed as an ordered pair (Cr, Cs), where Cr identifies the relevance condition and Cs identifies the satisfaction condition. If (part of) the student solution matches the relevance condition then the student solution should also match the satisfaction condition, otherwise the student solution is incorrect. Feedback associated with the constraint can be given to the student. There are three constraint types in J-LATTE, that is syntax, semantic, and style constraints. Syntax constraints are used to check if the student's solution contains valid Java code, semantic constraints are used to check if the student's solution is the correct answer for a given task, and style constraints are used to encourage student to follow good practices in writing programs. The following are examples for each constraint:

- Syntax constraint: “Each assignment must contain a valid expression on the right hand side”
- Semantic constraint: “If the task requires a function to be applied to a range of values, the solution must contain a loop”
- Style constraint: “The return statement should be at the end of a method”

As a CBM tutor, J-LATTE is expected to be more flexible than an ITS that uses a Model Tracing approach. Because J-LATTE does not store the solution paths then the students can write their program freely. Unfortunately, because there is no planning capability, J-LATTE (and other CBM Tutors) typically does not include a component that is able to solve the given task by itself (Kodaganallur et al., 2005; Mitrovic, 2003). Therefore the feedback in J-LATTE is usually general and not specific to the problem that is faced by the students. This can be considered a drawback because novice programmers sometimes need detailed feedback that can point out the specific mistake made.

J-LATTE was evaluated with 26 students from an introductory programming course at the University of Canterbury in 2008. The students were separated into two groups, the control and the experiment group. In the sixth week of the course, the students from both groups did a pre-test, used J-LATTE for 90 minutes, did a posttest, and filled-in a questionnaire. The students in the experiment group used the full version of J-LATTE, while the students in control group used J-LATTE without the feedback feature. The pre-test showed that there were no significant differences with the students’ performance between the two groups. Although it was expected that JLATTE would be beneficial to the students learning Java, the result from the posttest showed that there is no significant improvement achieved.

2.2.4 OOPS

Another approach to help students learn to program was adopted by Gálvez et al. (2009). In their research, they tried to help students learn the concepts of Object Oriented Programming (OOP) by using a strategy named blended learning. In this strategy, different modes of teaching and learning styles are used with the students. In addition to the traditional lecture, the students can use a learning tool called OOPS (Object Oriented Programming System) to do some OOP exercises and can access a testing system called SIETTE. In their previous research (Conejo et al., 2004), a system named SIETTE (System of Intelligent Evaluation Using Tests for Tele-education) was developed. This system can be used by the students to do self-assessment. SIETTE contains a collection of questions that are created by teachers and (as described further below) can provide the best questions to the students based on the students' current knowledge (Guzman & Conejo, 2005). Therefore the students' knowledge can be improved with the least number of questions. Unfortunately, with a complex domain such as OOP, the use of SIETTE will require a large number of questions and the students may have to answers too many questions in the assessment.

OOPS was developed with the intention of presenting the students with a few problems that are enough to be used to increase their performance, instead of using a large number of questions from the self-assessment test. However in its current version, OOPS only covers the main concepts of the object oriented data abstraction. No selection or iteration statements are covered in OOPS. The workspace that is used by the students to write the code is also limited to drag and drop (i.e. the students cannot write code directly). The approach that is used in OOPS is CBM. So the student model in OOPS is created from a list that contains every constraint violated by a student. This information is then used by its pedagogical module to select the appropriate assistance that should be given to the student, as well as to present the next question.

The students' performance was evaluated by firstly asking the students to attend a lecture on Object Oriented Programming. Then the experimental group of students was asked to do a pre-test through SIETTE. After the test, these students solved problems using OOPS and finally they did a post-test in SIETTE. On the other hand, a control group of students were only asked to attend the lecture and to do a post-test. The result from the evaluation showed that the experimental group has an increased performance after using OOPS.

2.2.5 JITS: Java Intelligent Tutoring System

The Java Intelligent Tutoring System (JITS) was designed and developed to help the students in their first programming course in Java at a college or university (Sykes & Franek, 2003). Although it is named as an "Intelligent Tutoring System", JITS does not contains some modules that usually exist in an ITS, such as the student module and the teaching module. There are two types of functionality in JITS, named "A" and "B" type. The "A" type functionality is used to handle very straight-forward programming tasks. For such tasks, JITS requires the task statement, the specification, and the solution. The Intent Recognition (IR) module inside JITS uses pattern matching between the solution and the student code. If the student types the code incorrectly, JITS will ask the student to confirm the code that he/she wants to write. For example, if the correct code is "int t =1" and the student writes "intt = 1" then JITS will ask if the student intention in the keyword "intt" is actually "int". Because the IR module uses pattern matching then it will oblige the student to write their code exactly the same as the solution code. A simple difference in the style of code writing, such as: "int a = 0;" versus "int a; a=0;" will not be allowed in the "A" type functionality of JITS. For a task with many solutions, JITS uses the "B" type functionality. In this, JITS employ a minimum distance error-correcting scanner-parser algorithm that is used to fix the student's code. For example, if the student writes the following code:

"public status flot TAX = 5;", the minimum distance error-correcting in JITS will correct this syntax to become: "public static float TAX = 5;".

For a more complex program (with more syntax errors in it), such as:

```
For (intt i = 1; i <= 10 i++ {
smu += i
}
```

JITS will try to find each syntax error in the program and confirm its findings with the student as feedback. For example, for the code above JITS will ask the student if the keyword "For" (with the capital "F") is a misspelling and the student's intention actually is the keyword "for" (with a lower case "f"). If the student answers "yes", then JITS will change the keyword to the correct one and continue to search another error. However, with these two types of functionality JITS can only help the student to write a syntactically correct program. No functionality exists to help the student with any logical errors. Therefore in "B" type functionality, JITS cannot check whether the student program solves the given task or not. The "B" type functionality in JITS is very similar to the syntax error checker that is included in many modern programming development environments.

JITS was used in Sheridan Institute of Technology and Advanced Learning in June to August 2004. Two classes participated in this study. One class was used as the experimental group and the other class as a control group. 14 students in the experimental group used JITS on regular basis. From the experiment, Sykes (2007) showed that there is a significant statistical difference in performance scores between the control group and the experimental group.

2.2.6 BITS

A different approach to help students learn to program was used in BITS (Bayesian Intelligent Tutoring System). It was created to help students learning to program in the C++ programming language (Butz et al., 2008). In their research background, Butz et al. (2008) described the problem of students of not understanding a particular programming topic because the students do not understand the prerequisite topics. Therefore, by providing an ITS that can understand the students' current knowledge of C++ programming and can suggest topics that they need to learn, the students will be able to learn C++ programming more effectively. Based on this approach, BITS does not ask the students to write programs inside of BITS. BITS concentrates more on how to provide the most suitable material to be learnt based on the students' current knowledge. As the name implies, BITS uses a Bayesian Network to keep track of the student's knowledge. For this purpose, BITS uses a node to represent each concept that is taught in the first programming course at the University of Regina, and a directed edge to represent a prerequisite from one concept to another concept. The nodes and their directed edges are a Directed Acyclic Graph (DAG). A conditional probability distribution is then specified for each node, conditional on its parents.

BITS was used in the initial computer programming course at the University of Regina. Even though Butz et al. (2008) claimed that the result of BITS was very positive, the lack of information about how BITS was evaluated, makes the claim unsupported.

2.2.7 iList

iList was created to help students learn linked list (Fossati et al., 2008). In their research, Fossati et al. (2008) also investigated the effectiveness of different types of feedback. For this purpose, two versions of iList were created. The first one uses simple

feedback and the second one uses more sophisticated feedback. Apart from the feedback differences, both versions of iList contain the same processes and modules. A simulated environment is created to allow the student to see the linked list as a visual entity. Two types of tasks are provided in iList. The first type of task is one that can be solved by entering the code step-by-step. For this kind of task, iList will display the effect of the code on the linked list, as the student submits the code. The second type of task is one that requires the student to write a snippet of code (not the whole program) to solve the task. A loop construct can be used in the code snippet. Five tasks of the first type and two tasks of the second type are provided in iList. As in OOPS, the CBM approach is used in iList. Some CBM constraints are defined in iList to check the correctness of the students' code. Although these constraints can be used to provide feedback when the students violate a relevant constraint, they cannot tell if a path that will never lead to a correct solution is being followed by the students (Fossati et al., 2008). The feedback in iList is grouped into three categories: syntax feedback, execution feedback, and final feedback. The syntax feedback is a given guidance and feedback to student when any unknown command is entered to iList. The execution feedback is for commands that can be understood by iList but cannot be executed due to the current condition of the linked list. The final feedback is given when the student asks about the correctness of their code. The difference between the first and the second versions of iList lies in the way the syntax and execution feedback is presented. The first version of iList was created with simple feedback, while the second version of iList provides a more sophisticated feedback.

In the evaluation, Fossati et al. (2008) firstly asked the students to take an introductory data structure class. Then in the students' scheduled lab sessions, the students did a pre-test, completed some linked list tasks from iList (except for the control group), and did a post-test. The students were grouped into four groups. The

first group – used as a control group – did not do the linked list tasks (they could do any irrelevant activities) before the post-test. However all the other groups did the linked list tasks. The second group did the tasks using iList version 1, the third group used iList version 2, and the fourth group did the tasks with human tutors. Although the result of the post-test showed that the students with human tutors achieved the best results, the performance of students with iList version 2 is less distinguishable from human tutor. Based on the results from the log analysis, Fossati et al. (2008) also concluded that the type of feedback in iList version 2 enabled the students to achieve a good performance level.

2.2.8 JavaGuide

This system, called JavaGuide, provide students with guidance to select suitable questions in Java programming (Hsiao et al., 2010). The basic idea in JavaGuide is similar to the idea in BITS. However while BITS provides guidance for the students to find the most appropriate topics to be learned, JavaGuide provides guidance to find the appropriate Java programming questions. For this purpose, the questions in JavaGuide were created using the same design as the one used in a system named QuizJET (Hsiao, Brusilovsky, & Sosnovsky, 2008). Several parameterized program fragments were created as the Java programming questions for the students. The actual value in each parameter variable is generated randomly and the users answer questions about the output of the program, or the value inside a particular variable. The basic difference between QuizJET and JavaGuide is that JavaGuide provides guidance for the students to find the suitable questions according to the students' current knowledge.

In their evaluation, Hsiao et al. (2010) showed that the adaptive guidance in JavaGuide encourages the students to do more work in the easy and moderate questions, while preventing them from venturing too early into the hard questions. As the result,

the students with JavaGuide were 2.61 times more likely to answer the questions correctly than the ones with QuizJET.

2.2.9 The PHP ITS

The PHP ITS was created to teach students the PHP scripting language for developing web pages (Weragama & Reye, 2013). The PHP ITS starts its analysis of a student's PHP program by converting the student code into an Abstract Syntax Tree (AST). This AST is then used to create facts or activate actions. The facts that exist after all nodes in the AST have been processed are called the final state. To check the correctness of the student program, the PHP ITS analyses whether all predicates in the goal for a particular task are present in the final state. Through this method, the student can write the solution of the given task in their own way, as long as the final state of their program satisfies the specified goal. The PHP ITS provides several exercises to be solved by the students. The students write their PHP programs within the PHP ITS user interface. The system recommends exercises to the students based on the student model (although the students can choose a different exercise if they wish). The student model is initialized from the students' pre-test answers. Multi-level hints are given to the students when they ask for help.

The PHP ITS was evaluated with postgraduate students enrolled in a unit to study PHP at the Queensland University of Technology in 2012. There were 34 students who took part of the evaluation. The control group was not used in the evaluation due to the ethical problem of limiting use to only some students. At first, the students were required to do a pre-test. After that the students used the PHP ITS to solve exercises in their own time. No lectures or tutorials given to these students. At the end, the students were asked to do a post-test and fill-in a questionnaire. From these tests, it can be shown that the results in post-test were significantly higher than the results in pre-test.

2.2.10 Discussion

Table 2.1 shows comparative analysis of different reviewed ITSs in this chapter. The developers of ITSs (in the programming domain), that were reviewed above, claim that they can help students learning to program in some way. Some ITSs help students by simply providing the most suitable material to learn (Butz, Hua, & Maguire, 2006) or by providing adaptive programming questions to answer (Hsiao et al., 2010) – see sections 2.2.6 and 2.2.8. However, the rest of the mentioned ITSs focus on practicing the writing of programs. Some ITSs ask the student to write programs to learn to program and they use two different approaches to check the correctness of the student program.

Table 2.1: Comparative analysis of different reviewed ITSs

Name of ITS	Year developed	Domain	focus on practicing the writing of programs	Focus on providing the most suitable learning material	Model Tracing (MT)	Constraint Based Modeling (CBM)	(MT) & (CBM)	Feedback and Hints	Associated Pedagogy (Gamification)
PROUST	1990	Pascal	*				*	*	
The LISP Tutor	1995	Lisp	*		*			*	
J-LATTE	1998	SQL	*			*		*	
OOPS	2005	OOP	*			*		*	
JITS	2005	Java	*		*			*	

BITS	2008	C#		*				*	
iList	2009	linke d list	*			*		*	
JavaGu ide	2010	Java		*				*	
The PHP ITS	2013	PHP	*				*	*	

One attempt to check the correctness of a student program is to use pattern matching and scanner-parser algorithms (Sykes & Franek, 2003), as described in section 2.2.5. Unfortunately, these methods can only help the student to write a syntactically correct program and not a logically correct program. The latter is actually more useful to the student because the former is already provided by many modern compilers. Another method to check the correctness of the student program, as described in section 2.2.1, was proposed by Johnson (1990). In his method, the correctness of the student program can be determined using variable and control plans. In variable plans, one or several variables are used to check if the student program can produce an expected value in those variables. On the other hand, control plans are used to regulate the generation of results in variable plans. Unfortunately, Johnson (1990) only produced two exercises that can be checked using this method. Two approaches that are commonly used in Intelligent Tutoring Systems are Model Tracing (MT) and Constraint Based Modeling (CBM). One example of an Intelligent Tutoring System (in the programming domain) that uses the MT approach is by Anderson et al. (1995), as described in section 2.2.2. On the other hand, tutoring systems using the CBM approach can be found in Holland

et al. (2009), Gálvez et al. (2009), and Fossati et al. (2008) – see sections 2.2.3, 2.2.4, 2.2.7.

An ITS in the programming domain that is built with the MT approach obliges the students to follow the same steps as the ones defined in the solution. This characteristic limits the students' freedom to write the program using their own steps. On the other hand, ITSs in the programming domain that are built with the CBM approach are concerned only with the final state of the student's solution. Such ITSs lack knowledge about how to solve the problem itself and therefore are limited in giving useful feedback to the students when they make a mistake in their attempt to achieve the goal (Gálvez et al., 2009; Kodaganallur et al., 2005). Revisiting the work of Johnson (1990) shows that it is actually similar to the combination of the MT and CBM approaches. His variable plans can be used to represent the final state of the solution (as used in CBM), and the control plans can be used to represent the process required to get the final state (as used in MT). A drawback of his approach is the complexity of the format of the variable and control plans used, so that he could produce only two exercises to be checked with this approach. If this problem is reduced or eliminated then more exercises or tasks can be created to support the students learn to program. One possible format that can be used can be seen in Weragama and Reye (2013), as described in section 2.2.9. In the system built, the student learns material that fits to their current knowledge. Learning the material that fits to the student current knowledge is proved to be effective (Butz et al., 2006; Hsiao et al., 2010). In order to know the student's current knowledge in programming, the student must write a program for a task given and the system will evaluate whether the student code can solve the given task or not. Some similar systems that ask students to write program in order to learn how to program can be found in Johnson (1990), Anderson et al. (1995), Sykes and Franek (2003), Holland et al. (2009), Gálvez et al. (2009), Fossati et al. (2008), and Weragama and Reye (2013). Unlike

Sykes and Franek (2003) which checks only the syntax of the student code, the system built checks the logic that is used to solve the task. This feature is achieved through a similar method that is proposed by Johnson (1990) and Weragama and Reye (2013).

Although it is valuable to provide guidance by identifying and correcting errors and misconceptions by having students simulate their own programs, students who are weaker do not benefit from this. The reason is their incapability to develop initial solution propositions. It is worth mentioning that rather than focusing on problem-solving skills, which are more essential for weaker students, the aforementioned tools emphasize more on programming language features. Thus, since learning to solve problems algorithmically contributes to learning how to program and because programming languages are merely a way of expressing solutions, greater focus should be directed toward students' problem-solving abilities (Conati & Van Lehn, 1999; Scott, 2010). Countless students are unable to develop flowcharts for simple problems and encounter difficulties in the preliminary learning stages. This may cause them to lose interest or give up, leading to dropout and failure.

In order to address the mentioned issue, the main aim of this thesis is to design and develop a flowchart-based ITS model which focus more on early stage concepts of computer programming along with visualizing the solution development for a programming problem by offering an adaptive and personalized flowchart development environment while engaging users in the process. Therefore, the proposed ITS could facilitates the creation of a mental model of execution for students and fulfill the gap in the area.

2.3 Intelligent Tutoring Systems and Gamification

Intelligent Tutoring Systems (ITSs) are effective in improving student performance and promoting overall learning gains (Woolf et al., 2009). However, these tools may

cause student disengagement and boredom, especially when timely guidance is lacking. Moreover, students also require extensive training and practice, which may be discouraging (Bell & McNamara, 2007; Gonz, Mora, & Toledo, 2014; McNamara, Jackson, & Graesser, 2010). Therefore, scientists in this field have been investigating possible ways to boost student motivation through ITSs. Gamification is basically the process of adding game-based features into a non-game based context with the purpose of increasing student engagement in a task, and therefore seems to be best fit to overcome this difficulty (Borges, Durelli, Reis, & Isotani, 2014; Deterding, Sicart, Nacke, O'Hara, & Dixon, 2011; Laamarti, Eid, & Saddik, 2014). Previous work has shown heightened student motivation and engagement in tasks when game-based features are applied in learning environments (Baker, Corbett, & Koedinger, 2004; Cordova & Lepper, 1996; Rai & Beck, 2012; Rowe, Shores, Mott, & Lester, 2011; Sabourin, Rowe, Mott, & Lester, 2011; Snow, Jackson, Varner, & McNamara, 2013). For example, Rai and Beck (2012) reported higher student enjoyment when game-based features were integrated into a complex learning environment. More recently, Snow et al. (2013) reported that students' interaction with personalized avatar features was positively related to the posttest measure of personal control and negatively related to the posttest measure of boredom. Therefore, a growing body of research has indicated that students' attitudes are positively affected by having game-based features added to ITSs. As an example of gamification in ITS, iSTART-ME, a game-based ITS, was developed with the aim of boosting students' understanding of science texts through teaching reading strategies (Jackson & McNamara, 2013). The experimental study findings revealed the increment in students' engagement, persistence and motivation (Jackson & McNamara, 2013; Snow et al., 2013). However, off all existing ITSs in the area of computer programming, none of them added game-based features into their environments for the purpose of increasing students' engagement in a task specially

when the timely guidance are lacking, as shown in Table 2.1. As a result, gamification is a motivational feature which should be added into ITS to address aforementioned issue of students.

2.4 Problem-solving Skill in Computer Programming

In computer programming, the term problem-solving describes the transition from a problem specification or requirement to a working computational program. Westphal, Harris, and Fadali (2003) note that the transition from a problem specification through to a working syntactical solution is a particular difficulty for novice programmers.

Problem-solving encompasses a range of skills that relate to the processes of program design. Problem-solving can be broken down into two activities, algorithmic problem-solving and debugging. Algorithmic problem-solving is taking the constructs and putting them together in meaningful and intentional ways to express a solution to a specified problem or program requirement. Debugging is the process of finding and fixing logic, semantic and declarative errors within the program's design or resultant code. Problem-solving is the most transferable skill set a novice will learn and is pertinent to programming in any language or paradigm. Fostering problem-solving skills should be the main aim of an introductory course in programming. Problem-solving can be broken down into 5 stages as follows, where stages 1 – 3 are algorithmic problem-solving and stages 4-5 are testing and debugging which are carried out repeatedly until the problem is solved.

1. Understand the problem to be solved;

2. Break problem down into manageable pieces;

3. Design a solution;

4. Code the solution in a programming language via an IDE;

5. Find and fix any problems, debugging ;

Many authors have indicated that within and upon completion of an introductory programming course, many students are not achieving the expected level of problem-solving skill that the course intended to teach. As indicated by Howel (2003), "since students are no longer focused on elementary education, they cannot master problem-solving skills. Students have become so weak in the development of problem-solving skills, resulting in negative effects in independent and meaningful learning in all disciplines". This supports the notion that students are deficient in problem-solving skills. M. McCracken et al. (2001) gained further evidence to back this claim. The results of a large-scale multi-institutional study in a CS1 level computer science course suggested that students lacked essential skills required to abstract a problem into a solution despite having attained acceptable knowledge levels of programming implementation and syntax. A review of research related to the current problems with learning and teaching programming conducted by Robins, Rountree, and Rountree (2003) also corroborates this point. In their research, the strategy employed by students when applying programming knowledge was deduced to be the difference between effective and ineffective novices. Mciver and Conway (2000) highlighted that a reason for this ineffectiveness is forcing students to deal with syntax and the individual elements of algorithms, which may cause excluding the broader picture of the problem at hand. While the results of one isolated study could be attributed to poor tuition, the large number of concurring studies provides sufficient evidence to back the claim that problem-solving is a key weakness of novices.

Within introductory programming the development of problem-solving skills is often overshadowed by the complexities of the development environment, the programming

language and the novices' sloppy approach to programming. More often than not novices do not approach their day to day (non assignment based) programming activities via the analysis, design, code, and test methodology even though instructors and text books stress the importance of this approach (Lakanen & Isom, 2015). The natural tendency of the novice is to focus on a coding by trial and error style, skipping the analysis and design phases, diving straight into the coding, before thinking the problem thorough and planning a solution (Dale & Weems, 2014; Perkins, Hancock, Hobbs, Martin, & Simmonds, 1988). Subsequently the development environment and language forces the programmer to wrestle with the syntax before gaining any feedback on the efficacy of the ad-hock algorithm they are trying to implement. This creates an unbalanced focus on syntax and overshadows the problem-solving process. This approach is made more difficult by the novices' fragile and incomplete comprehension of the constructs and how they interact; this makes decomposing solutions directly into code prohibitively difficult. Debugging requires the same prerequisite skill set as algorithmic problem-solving, therefore it is logical to assume that those with weak algorithmic problem-solving skills have weak debugging skills (Bouras & Ainarozidou, 2015; Kalelio, 2015).

From the research reviewed, it is clear that problem-solving skill is deficient in a large proportion of novice programmers and that this is a long standing issue of global concern. Various authors have shown that developing mental models and process flow in preliminary programming courses could highly determine and impact a learner's progress in computer programming (Ramalingam, LaBelle, & Wiedenbeck, 2004). The significance of such models in enhancing comprehension is further highlighted by Winslow (1996). With regards to the fact that generally learners are not granted with a cognitive or visual model in the context of learning, they will try to code the solution to a problem while holding no mental model for it. To transfer the concepts of

programming, educators mainly emphasize on pseudo codes despite the fact that the execution flow between a program components and its flowchart cannot be explained by such justifications. Due to the absence of a mental model of execution, learners experience critical complications in grasping the relationships between different individual programming components.

With the aim of visualizing programming constructs and translating a problem specification to its corresponding program code solution, the use of flowcharts has been broadly suggested and employed. Clear mental models of algorithms without the need of any prior training are proposed to learner programmers through flowcharts. Westphal et al. (2003) observed that "Even with having pseudo code, it is so hard for novices to communicate the flow of a program, unless using flowcharts or diagrams". But for complex problems, UML activity diagrams are considered as a more proper and contemporary visualization compared to flowcharts for novice programmers; flowcharts are well-adapted for non-complex programs and for transferring the fundamental concepts to novice programmers. In addition, considering its visualization and functionality, UML activity diagrams are not beneficial compared to flowcharts for non-complex novice programs. By employing visualization-based tools and environments, a flowchart's performance can be enhanced to help learners in problem-solving and program development, as pointed out by Bassat Levy et al. (2001). An actual model of execution which is required by the majority of novice programmers in order to learn about algorithms and programming concepts is given through such environments. Limiting novice programmers from engaging in the statements of programming problems possibly can result them in facing critical challenges when they are confronted with advanced programming problems. Moreover, it should be mentioned that novice programmers should find the connection between the statement of programming problems and their corresponding flowcharts. This helps the programmers to connect a

programming problem statement to a flowchart form more productively, helping shift from a problem text specification to a corresponding flowchart.

There are several flowchart-based tools designed which acts as a visual aid in programming to improve problem-solving skills of novice programmers. As stated by D. Hooshyar et al. (2015), there are 17 flowchart-based programming environments developed with the aim of improving problem-solving skills of novice programmers and they all indicated to flowchart as an effective representation for learning the basics of programming. Whilst these systems possessed useful features in isolation, their other features were either limited, or poorly designed. More importantly, none of these tools could provide their users with an adaptive and personalized flowchart development environment and engage them in the process of flowchart development at the same time with the aim of improving their problem-solving skills. Comparative analysis of flowchart-based programming environments is shown in Table 2.2.

Table 2.2: Comparative analysis of flowchart-based programming environments

Features	BACCII	FLINT	EC	FCI	Raptor	SFC	SICAS	VL	IP	DF	A&Y	B#	PG	G&G	CVF	Web-flowchart	Progr ani-mate
Year Published	1992	1999	2002	2003	2004	2004	2004	2004	2005	2006	2006	2006	2007	2007	2009	2010	2010
Flowchart	✓	✓			✓		✓				✓			✓		✓	✓
Flowchart based programming	✓	✓			✓		✓									✓	✓
Flowchart generates code	✓				✓		✓										✓
Structural rules enforce	✓	✓			✓						✓						✓

d																	
Color used to fully differentiate components and structures	✓	✓															
Code	✓				✓		✓				✓			✓			✓
Code based programming																	
Code generates flowchart																	
Code and flowchart displayed concurrently											✓			✓			✓
Synchronized highlighting of flowcharts and code										✓	✓			✓			✓
Synchronized Visual Execution of Flowcharts and code											✓			✓			✓
Non visual execution																	
Visual execution		✓			✓		✓				✓			✓			✓
Variabl	✓	✓			✓		✓				✓						✓

e inspector																	
Error feedback	✓	✓			✓						✓			✓			✓
Java support					✓		✓										✓
Variables	✓	✓			✓		✓				✓			✓		✓	✓
Sequence	✓	✓			✓		✓				✓			✓		✓	✓
Selection	✓	✓			✓		✓				✓			✓		✓	✓
Iteration	✓	✓			✓		✓				✓					✓	✓
Arrays	✓				✓		✓									✓	✓
Procedures	✓	✓			✓		✓									✓	
Empirically Evaluated	✓	✓			✓												✓
Associated Pedagogy		✓												✓			✓
OS Dependency	win	win	win	win	win	win	win	win	win	win	any	win	win	win/ Mac	win	win	Independent

In this section, in addition to summarizing all features of related works in the form of table, the most significant characteristics of some of the flowchart-based programming environments are pointed out. Regarding language, MicroWorlds Pro, Visual Flowchart, and A&Y are founded on specific national languages, so English speaking users are not able to use these tools and additional learning is required in order to use these environments. In terms of notations, some of the reviewed works are not in line with standard flowchart symbols. In some cases, symbol shapes are similar and characteristic icons are employed, such as in BACCII, BACCII++ and B#, whose developers prefer to use symbols with a term-iconic programming language. In a few environments, different colors are applied, to distinguish between same-shape symbols for diverse

structures, such as in Progranimate. Concerning object oriented support, to introduce novice programmers to programming, flowchart and algorithms are broadly applied and imperative-procedural programming technique is supported by majority of flowchart-based programming environments. According to Scott (2010), using object-first method may cause some inconveniences in the imperative approach. Though, in BACCII++ and RAPTOR object-oriented programming techniques and procedurals are supported. It should be noted that in BACCII++ it is difficult to recognize how exactly object-oriented paradigms are covered as this tool is not accessible free of charge. A UML class diagram can be designed by RAPTOR users by applying manipulation methods and interaction. More importantly, the majority of object-oriented concepts can be depicted with this tool. With reference to execution, amongst all reviewed tools, only three of them named BACCII, BACCII++ and SFC editor do not provide users with flowchart execution. However, according to the SICAS designers, backward step-by-step execution is also supported by this tool. In B#, the automatic generated code from flowchart can be executed, whereas flowchart and code operation are provided in many languages in Progranimate (Scott, 2010). Similar to IP, Progranimate uses explanatory visualization to show explanatory message in natural language by giving information about flowchart components while it is being executed. These futures can be considerably helpful in understanding flow of control and the semantic of the programming structure for novice programmers. In relation to the automatic generation of source code, SFC editor produces a type of pseudo-code that needs additional modification. However, the code is generated in an automatic manner as in other cases such as in BACCII/BACCII++, RAPTOR, SICAS, etc. without modification. It is important to mention that RAPTOR has the potential to generate code from its flowchart in various programming languages such as Ada, C#, C++ and Java. The execution of the source code is offered by B#, whereas both executions of the flowchart

and the source code at the same time is supported by Programinate. The automatic creation of source code that is syntactically correct may offer novices some help in understanding the connection between algorithms that they used in the form of flowchart and a programming language. The power of generating source code in different languages shows clearly that an algorithm which is properly designed has the option of being translated into different programming languages with ease. Finally, to perceive an algorithm is more important than to think about the programming language employed for implementing it.

2.5 Flowcharts -- Effective Visualization for Novice Programmers

Mental models are very important for comprehending programming and other technical computing subjects. A student's success in preliminary programming courses is extremely dependent on and affected by the development of process flow representations and mental models (Ramalingam et al., 2004). The importance of these models in gaining understanding was also emphasized by Winslow (1996). Students mostly attempt to code a solution while not having an appropriate mental model of the solution to a problem; in addition, most are not even provided with a visual model in the learning context. Instructors often employ pseudocodes to convey programming concepts, as they offer good explanations for instruction in English to some extent. Nonetheless, pseudocodes cannot show the program execution flow between the program components and flowchart. Thus, flowcharts serve to visualize the programming structure and overcome difficulties in translating a problem's specification to its corresponding program code solution.

Flowcharts have a graphical representation to describe a set of rules or the detailed logic of a process. In the programming domain, flowcharts are typically used to visualize and communicate the design of programs, algorithms and procedures by

visually depicting the respective concepts of sequence, selection and iteration. Flowcharts do not necessitate prior training and can be understood easily, as they can express any program through only five basic flowcharting symbols (see Figure 2.2). Furthermore, flowcharts are used in our day-to-day lives and various disciplines such as science and mathematics. For this reason, it is likely that a student who is learning programming for the first time will be familiar with flowcharts to some degree. Therefore, flowcharts are considered very cognitive-efficient.

Flowcharts are designed to aid with program composition, comprehension and the transition from problem specification through to syntactical solution -- skills in which novice programmers seem to be quite weak. They are also very appropriate in a lecture setting as the instructor can convey and discuss the concepts of sequence selection and iteration and how they are implemented within a program or algorithm without the distraction of language syntax (Khalife, 2006).

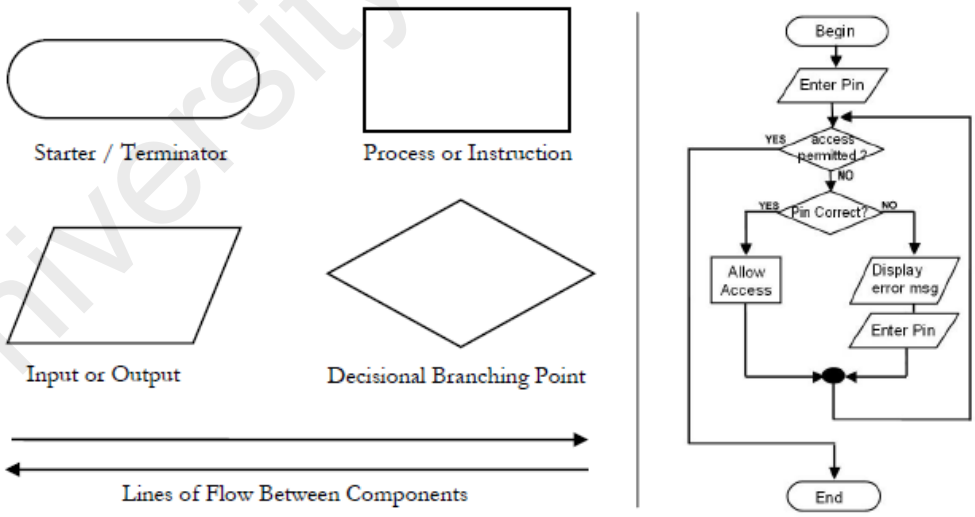

Figure 2.2: The simplicity of flowchart notation

Flowcharts can help students form accurate mental models, a crucial factor in novices' comprehension of programming. Students can form inappropriate or misleading mental models if not guided otherwise. Therefore, within programming

instruction, it is important to incorporate instructional support that minimizes the likelihood of triggering inappropriate mental models. A flowchart can afford novices with accurate and useful visual metaphors of a program and the semantics of individual program components. Therefore, flowcharts are useful instructional aides for promoting the formation of accurate, appropriate and useful mental models of program logic. While flowcharts may not be ideal for modeling large and complex programs of professional programmers, they are very appropriate for modeling and visualizing relatively small-scale procedures, simple programs and the foundational, imperative programming concepts for novice programmers of all programming paradigms. Novices can envisage the semantics of conditional structures, the flow of execution, and how different pieces of a program interact to form higher level concepts through flowcharts. Flowcharts also limit the syntactic overhead of a programming language. As indicated by Westphal et al. (2003), "Even with having pseudocode, it is so hard for novices to communicate the flow of a program, unless using flowcharts or diagrams".

Even though UML activity diagrams are more appropriate modern visualizations than flowcharts, for conveying basic concepts and simple programs to novice programmers, flowcharts are better-suited than UML activity diagrams. As indicated by many researchers such as Bassat Levy et al. (2001), visualization-based tools are capable of extending the effectiveness of flowcharts with the aim of assisting novices with problem-solving and program development. A concrete model of execution, which is essential for most novices to understand algorithms and programming, is offered via these environments. Novice programmers may face serious difficulties as a result of being prevented from engaging in the statement of a programming problem, specifically once they encounter new programming problems. Thus, observing the relation between the textual forms of a programming problem and its relevant flowchart is necessary for novices, as this enables relating a programming problem statement to flowchart design

more effectively. As a result, such environment could facilitate the creation of a mental model of execution for students through helping visualize the solution development for a programming problem by converting the given problem statement into a relevant flowchart, all while engaging users in the process.

2.6 Dynamic Flowcharts

The capability of modern computers entails that visualizations, such as flowcharts, no longer need to be static. The dynamic nature of a computer program and its flow of execution can be animated via computer-based flowchart visualization. An executable flowchart is a significant improvement over static flowcharts, leaving the process largely up to the mind of the viewer. Because of this, a dynamic flowchart is more likely to trigger the formation of appropriate and accurate mental models of execution and decrease the chances of conceptual inaccuracies. Since learners are not required to manually follow up with the state of program data using dynamic flowcharts, the amount of information the learner must hold in their working memory or write down is minimized.

Not only can a dynamic flowchart serve to visualize program execution, it can also be dynamically constructed. A criticism of flowcharts is that they are difficult to update, since new ideas and modifications are added, in which case a traditional flowchart needs to be completely redrawn. To a large extent this is also true of computer-based graph drawing systems like Visio (Microsoft, 2009), whereby adding or removing items requires the user to significantly rearrange the existing diagram. This can take time and makes the flowchart a less than compelling design methodology. However, computer-based flowchart visualization can overcome this requirement by tidily restructuring and redrawing a flowchart automatically as items are added or removed (auto structuring). Novices thus have access to an effective mental model of the constructs and how to

compose them whilst reducing syntactic overhead through flowchart-based representation.

This places greater emphasis on the underlying abstractions of programming, problem-solving and program composition, i.e. how the pieces fit together to form a solution to a problem or specification. The dynamic flowchart execution will allow novices to evaluate the appropriateness of their solutions, gain a deeper understanding of the programs they have composed and foster a mental model of program execution. The product of this thesis (as discussed in Chapter 5) is to employ a dynamic flowchart to great effect for visualizing both program construction and execution.

2.7 Flowchart-based Learning Approach for Improving Students' Problem-Solving Skills

Although it is valuable to guide students in identifying and correcting errors and misconceptions by simulating their own programs, weaker students do not benefit from this. The reason is their incapability to develop initial solution propositions. Thus, as learning to solve problems algorithmically contributes to learning how to program, more focus should be directed toward problem-solving skills rather than programming language. With the hope of addressing the above matters, the Flowchart-based Multi-agent System (FMAS) was developed with the benefits of an automatic text-to-flowchart conversion approach. With FMAS, novices should be able to create their initial solutions for simple problems and improve their problem-solving skills accordingly. Novices are provided with a dialogue system chat that guides them through solution formation. Moreover, synchronization of the programming problem statement with its corresponding flowchart, hints, errors and extra information is available in order to encourage step-by-step solution construction.

The rationale behind integrating the automatic text-to-flowchart conversion approach into FMAS is to assist students visualize the relation between the problem statement and its pertinent flowchart as they become engaged in the process of flowchart development that can subsequently be improved (Hooshyar, Ahmad, & Nasir, 2014). Thus, the novel proposed text-to-flowchart conversion approach is intended to simulate the human mechanism of programming problem-solving, in English, by prompting novices to focus on the solution rather than on the programming syntax as they become involved in meaningful planning activities and solution design so as to enhance their problem-solving skills. As a result, users can observe how the programming problem statement maps onto a flowchart through the proposed automatic text-to-flowchart conversion approach.

2.8 Discussion

Based on the principles of human learning, to learn effectively students must (i) be involved, engaged and active in the learning process, (ii) be able to learn at their own learning pace, and (iii) be provided with material that is in accord with the state of their current knowledge (Woolf, 2008). In the programming domain, one strategy that can be used to learn effectively is for students to learn to design a solution for a programming problem by developing a flowchart themselves and learning the material through writing programs themselves as well (Felder, 1988; D. Hooshyar et al., 2015; Lahtinen, Ala-Mutka, & Hannu-Matti, 2005). Students should have access to help when encountering problems in flowchart development or when writing programs (Anderson & Reiser, 1985).

Of all ITSs discussed in this chapter, none is fully designed to support the first principle of human learning, which makes students become engaged in the learning process. In addition, students may become demotivated when using the majority of the

aforementioned ITSs due to the lack of timely guidance and because motivation and enthusiasm need to be actively raised. Besides, all existing programming ITSs focus more on programming language features and syntax and less on early-stage computer programming concepts or visualizing the solution development for a programming problem by offering an adaptive and personalized flowchart development environment while engaging users in the process.

Although it is valuable to provide guidance by identifying and correcting errors and misconceptions by having students simulate their own programs, students who are weaker do not benefit from this. The reason is their incapability to develop initial solution propositions (Kalelioğlu, 2015). It is worth mentioning that rather than focusing on problem-solving skills, which are more essential for weaker students, the aforementioned tools emphasize more on programming language features. Countless students are unable to develop flowcharts for simple problems and encounter difficulties in the preliminary learning stages. This may cause them to lose interest or give up, leading to dropout and failure. In order to address the mentioned issues, the main aim of this thesis is to design and develop a flowchart-based ITS model. The model emphasizes more on early-stage concepts of computer programming along with visualizing the solution development for a programming problem by offering an adaptive and personalized flowchart development environment that engages users in the process. Therefore, the proposed ITS may facilitate the creation of a mental model of execution for students and accordingly fulfill the gap in this area. Moreover, with the aim of raising student motivation and engagement in a task, a game-based feature is added to the proposed ITS to prevent student demotivation.

2.9 Summary

This chapter described overview of Intelligent Tutoring System (ITS) and reviewed existing ITSs in the programming domain; ITSs and Gamification were then briefly explained. Moreover, the concept of problem-solving skill in programming is thoroughly then explained along with some studies that have been done on improving problem-solving skill using flowchart. This chapter ends with a discussion of the shortcomings of the existing ITSs in the programming domain and the research work in this thesis that aim at addressing those shortcomings. In the next chapter, the research design and architecture will be explained.

CHAPTER 3: RESEARCH DESIGN AND SYSTEM ARCHITECTURE

This chapter outlines the design and methodology of the research. It starts with an overview of the research methodology that is used in this research. This chapter then continues with the information about the research design and the instruments used, the procedure and timeline, the participants of this research, and the ethics and limitations. The last two sections in this chapter describe the requirements of the system and its architecture.

3.1 Research Design

3.1.1 Methodology

This thesis involves a rigorous process of designing an artifact to solve observed problems, making research contributions and evaluating the design, and communicating the results to appropriate audiences. For this reason, the science design research methodology (SDRM) is the chosen research methodology. SDRM is often used to create and evaluate IT artifacts aimed at solving identified organizational problems (Peffers et al., 2007). Artifacts may be algorithms, human/computer interfaces, design methodologies (including process models) and languages. Artifact application can be found in many disciplines and fields, the most notable being engineering and computer science. Creating an innovative and purposeful artifact for a special problem domain is must in design science research, as indicated by (Hevner, March, Park, & Ram, 2004). Afterwards, to ensure the usefulness of the designed artifact, it must be evaluated. It is worth mentioning that the artifact should either offer a more effective solution than other existing solutions or it must solve a problem that has yet to be solved. At the end, the proposed artifact should be developed and evaluated, and the findings and results must be presented to both management and technology-oriented audiences. SDRM comprises six steps: 1) problem identification and motivation, 2) defining the

objectives, 3) design and development, 4) demonstration, 5) evaluation and 6) communication.

3.1.1.1 Problem identification and motivation

In this phase of SDRM, the research problem is defined along with a justification of the solution's value. The conceptual atomization of the problem is useful in this step of SDRM, as the problem definition will be used in constructing an artifact that can successfully offer a solution. Justifying a solution's value can motivate researchers and the audience of the research to pursue the solution and help to recognize the reasons behind the researcher's understanding of the problem. Knowledge of the state of the problem and the significance of its solution are two resources required for this activity.

3.1.1.2 Definition of the objectives for a solution

Defining the objective of a solution is the second phase of SDRM. Some scientists implicitly attempt to transform the problem into system objectives, i.e. part of programming and data collection, while other scientists' effort to transform the problem into system objectives is explicit. Due to the fact that the design process is essentially one of partial and incremental solutions, the identified problems should not essentially be translated into objectives of the artifact. Thus, following the problem identification stage, the solution objective should be rationally inferred from the problem specification. The objective can either be qualitative or quantitative. Describing how a new artifact can support solutions to problems that have yet to be addressed is an example of a quantitative objective, whereas an instance of a qualitative objective is the term in which a desirable solution can be better than current solutions. Having knowledge of the state of the problems and current solutions entails two resources required for this activity.

3.1.1.3 Design and development

Artifact design and development is the third phase of SDRM and to which most researchers pay more attention. As aforementioned, artifacts could be methods, models and constructs or new properties of technical and social resources (Pertti, 2007). In some research, this stage of SDRM is divided into different activities, where researchers are more interested in the nature of the iterative search process (Hevner et al., 2004). Theoretically, any designed object in which research contribution is embedded can be considered a design research artifact. It comprises determining the artifact's functionality and then actually creating the artifact. Knowledge of theory that can be brought to yield a solution is a resource required for this activity.

3.1.1.4 Demonstration and evaluation

Demonstration is the fourth phase of SDRM, in which artifact workability is investigated using one or more examples. Therefore, in this stage, the artifact is used in simulation, case study, experimentation, or other suitable activity with the aim of solving one or more instances of the problem. Effective knowledge of how to use the artifact is the necessary resource for this activity. In the fifth, artifact evaluation phase of SDRM, how well the artifact supports a solution to the problem is measured and evaluated. The objectives of a solution and the actual observed results from using the artifact are compared in this activity. Knowledge of analysis techniques and relevant metrics is required in this stage. Evaluation could be done in different forms based on the nature of the artifact. Theoretically, the evaluation could comprise any proper logical proof or empirical evidence. Whether to continue on to communication and leave further improvement to subsequent projects, or reiterate step three to try and improve the artifact's effectiveness, are two possible courses of action that researchers can take at the end of this stage. It should be noted that in some research, not all, the iteration may be feasible.

3.1.1.5 Communication

In the final, communication phase of SDRM, the problem and its importance as well as the artifact, its usefulness and its effectiveness to researchers and other relevant audiences are communicated. Scientists may use the structure of this process in their research publications as the nominal structure of an empirical research process. In applying SDRM, the researcher may not always proceed in sequential order through the 6 activities and may begin at any step. For example, in a design and development-centered approach, the researcher may start with activity three. The presence of an artifact that has not yet been formally considered as a solution for the explicit problem domain may result in this. In case of an objective-centered solution, the researcher may begin at activity two, potentially as a result of an industry or research need that can be addressed by developing an artifact. Concerning the problem-centered approach, a researcher usually starts with activity one, perhaps if the research is a result of problem observation.

3.1.2 Research Design and Instruments

Both quantitative and qualitative methods are used in this research. The type of the quantitative method used is experimental. It is "experimental" in that a certain computer system is built to be used by students. Its impact on the students will be measured by comparing the students' pre-test and post-test results. The students' pretest and post-test are performed before and after the students use the system, respectively. The qualitative method in this research is performed by distributing a questionnaire to the students after they use the system. Some independent variables that may affect the result of the measurements were captured and used to determine their correlation with the measurements. The instruments used for this purpose are a questionnaire and the FITS database. The independent variables investigated in this research are:

- The student's previous programming knowledge (Have they learnt programming before?).
- The length of the time the student uses FITS.
- The number of time the students played the game in FITS.
- The number of tasks completed by the student (number of flowchart completed along with number of programming concepts learned).

The questionnaire (see Appendix A: FITS Questionnaire) is also used to measure the performance of FITS itself. There are 19 closed questions and one open question in this questionnaire. In the closed questions, the students can express their agreement or disagreement by using a five-level Likert item. The 19 closed questions are used to find the students' opinions about:

- The easiness of using FITS.
- The usefulness of the assistance in FITS.
- The problem-solving approach (flowchart development).
- The satisfaction in using FITS.
- The usefulness of FITS in improving problem-solving skills.
- The appropriateness of the number of tasks that must be done. Unlike the other questions, the answers for this question starts with "too few" as the first item and "too many" as the last item. "Just right" is provided in the middle of the answers.

The open question at the end of the questionnaire gives the students a chance to express their opinion on the issues that may not have been covered in the closed question. The students are asked to mention whatever comments or suggestions they have regarding FITS.

3.1.3 Procedure and Timeline

In 2015, FITS was released for the course named Programming I class. The students must perform a pre-test that is displayed by the FITS application. This pre-test is used to measure the students' knowledge before they use FITS. There are 20 multiple choice questions in this pre-test (see Appendix B: Pre- and Post-Test Questions). The students can do the pre-test only once. After the pre-test, the students could see the learning materials of basic and imperative programming concepts by clicking on the traffic light on the navigational menu. Also, after going through the learning materials, they could enter their desire programming problem to develop the respective flowchart with the help of FITS. There are currently 16 tasks (programing problem) defined inside FITS. However, the number of tasks should be done by the students varies depending on the correctness of the students' answers to the pre-test and their performance during learning materials and flowchart development. The students are asked to reach their programming knowledge to higher than 50 % (by clicking on different unknown concepts) and then proceed to other tasks which are aimed for problem-solving skills improvement. After completing the tasks, the students are asked to do a post-test. The questions in this post-test are the same as the questions in the pre-test but displayed in random order. As in the pre-test, the students can only do the posttest once.

The last step that must be done by the students is to fill-in the questionnaire that asks for their opinion about FITS and the approach used by it. The questionnaire could be filled online for all students. The timeline for the students to use FITS ranges from one to three weeks. After being released, the students have a chance to use FITS as much as they like. The procedure described above was an improvement of the procedure that was used in the first release. In its first release, in 2014, the pre-test in FITS was performed by asking the students to do all the tasks in FITS, with no help feedback, or guidance provided. After the pre-test, the students could use FITS with all of those functionalities.

Their work was saved to be used as a posttest and was compared with the pre-test results to evaluate their performance. More details about this is described in section 6.1.1.

3.1.4 Participants

The targeted participants for FITS are university students who take "Programming I" class. This class is offered in both Semesters 1 and 2 at University of Malaya (UM). FITS was released for the first time in 2014. There were only six students who participated in this first release. The result and feedback from those students was used to improve FITS before it was released for the second time in 2015. There were 30 students who participated in this second release.

3.1.5 Ethics and Limitations

As described above, the participants of this research are UM students who took the Programming I class. Ethical problems would have arisen if only some students were allowed to use FITS and others were not. If FITS gives an advantage to students who are learning to program, then preventing some students from using FITS, while allowing others to use it, would be ethically unacceptable because of the possible impact on the students' final grades (Weragama & Reye, 2013). Therefore, it was not possible to create a control group in this research. All students were given the opportunity to use FITS.

3.2 System Requirements

The system requirements defined here are based on the requirements defined in some studies in ITS and pedagogical field. These system requirements were used as a basis to develop the prototype system. The prototype system was then released to the students and their feedback was used to improve the second version of FITS.

3.2.1 Functional Requirement

In order to achieve the goal of this research, the system developed should have the functionalities that:

- Make the student improve their problem-solving skills in an adaptive and personalized environment.
- Let the student learn to design solutions for programming problems by engaging them in flowchart development activities.
- Capable of determining the student current knowledge.
- Provide tasks that appropriate to the student current knowledge.
- Allow the student to select the task.
- Enable the student to use the current environment to develop their flowchart.
- Check the correctness of the student flowchart in the development environment.
- Provide incremental help and feedback that are relevant to the student error.
- Provide the solution for the given programming problem.
- Enable the student to continue their work in another time.

3.2.2 Non-Functional Requirement

There are two non-functional requirements that are identified in this research. The first one is the response time of the system to deliver the feedback or help. Ideally, this response time should be fast enough to prevent the user from being annoyed of the waiting. For ITS application, two or three second response times are still considered as acceptable.

The second non-functional requirement identified is that the ITS should provide a mechanism to extend the tasks easily. To add a new task, the teacher should just add a new task title, task description, the goal of the task, and the help and feedback for that

task. The teacher should also set the difficulty of the task and the knowledge needed to solve it. All of these should be performed without modifying the ITS code. An authoring system is required to help the teacher add new tasks more easily. The output from the authoring system can be stored in a database to be read by the ITS. Therefore the code in the ITS does not need to be changed when the teacher adds or modifies the tasks. Although the authoring system is very helpful to the teacher, this authoring system is not part of the ITS.

3.3 System Architecture

As described in section 2.1, there are four components that usually exists in an ITS. They are Domain, Student, Teaching, and Communication modules. As an ITS, FITS contains all of those modules in its system architecture. The purpose of the modules and their relations in the FITS system architecture are follow:

3.3.1 Domain Module

This module is responsible to retrieve the knowledge about the instructional domain. In FITS, the domain knowledge contains information about the tasks and the goals that must be accomplished for the tasks. Feedback and help are also stored in this knowledge and linked with the specified goals. In this module, FITS also check the correctness of the student program based on the specified goal. Details of how this process is performed are described later in section 5.4.

3.3.2 Student Module

This module is responsible to record and update information about the students' knowledge. In this module, the initial belief about the students' knowledge is calculated based on their answers in the pre-test. This knowledge is then updated based on the tasks designed for them. Additional information related to the students is also processed by this module, such as the starting and ending time of using FITS, the time when the

students are really learning programming concepts and developing their program, the tasks or sub tasks that give them, the need to ask for help when they have problem, and the number of help requests that they make. Details of how this process is performed are described later in section 5.2.1.

3.3.3 Teaching Module

Through this module, FITS provides suitable feedback and help to the student. The feedback and help is customized to the student need based on the information from the student module. This module is also responsible to give recommendation about the next suitable tasks for the student. Details are described later in section 5.2.4.

3.3.4 Communication Module

This module is responsible to provide communication between student and FITS. FITS needs to present the task, feedback, and help to the student. The student needs to tell their needs to FITS. In this relation, FITS needs to check how students are doing with learning material tasks as well as monitor the student flowcharts and in some occasions modify the student flowcharts automatically. All of these needs must be facilitated with as easy and simple communication means as possible to avoid overwhelming the student with too many new things at the same time. To achieve this, some processes are handled automatically by this module, such as providing students with the percentage of their programming knowledge, showing how much they know about each specific concept, automatically generating flowchart for the entered programming problem, offering them feedbacks and helps, changing one task to the next task, and monitoring student actions. As a result, FITS only needs one rich text box, flowchart development environment, and some buttons to communicate with the student. Section 5.2.3 describes the details of the communication module in FITS.

Figure 3.1 shows the structural relationships between these modules. This figure shows how the conceptual ITS structural relationships (as described in Figure 2.1) have been adopted in FITS.

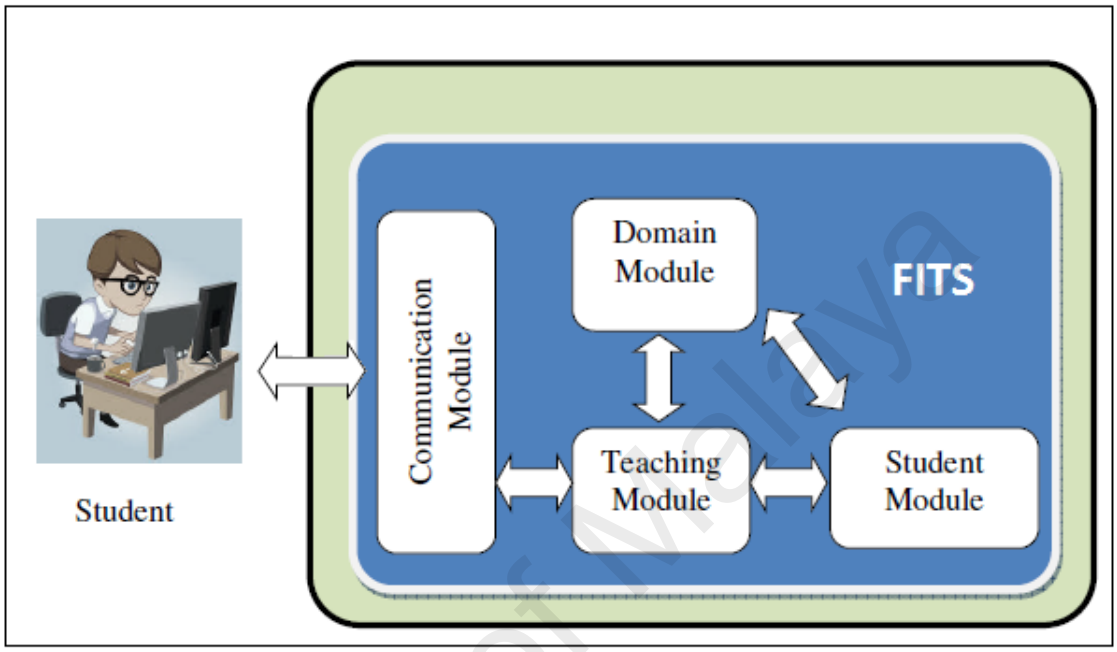

Figure 3.1: Structural relationship among modules in FITS

3.4 Summary

This chapter described the research methodology used in the research. It also explained the instruments used, the procedure, timeline, and the participants. The system requirements were then described in two categories, the functional and the non-functional requirements. In the last preceding section, the architecture of the system is described. In the next chapter, this research design and architecture will be used as the foundation for the details of the analysis and design of the system.

CHAPTER 4: THE FLOWCHART-BASED LEARNING APPROACH

As explained in **c**hapter 3, the general ITS architecture contains four components, namely the domain module, student module, teaching module and communication module, the most important of which is the knowledge module. This chapter offers an overview of the design of the flowchart-based multi-agent system (FMAS), which is incorporated to fulfill the knowledge module of FITS. Subsequently, the different FMAS components along with the various flowchart development options, feedback, error detection and help offered to students are provided.

4.1 FMAS's Architecture

Two distinct scenarios have been defined in FMAS, namely when a keyword is found and when it is NOT found. Concerning the former, six agents and five sub-agents have been employed to convert the textual form of programming problem into its relevant flowchart, while the latter involves four agents along with two sub-agents to generate the corresponding flowchart. All agents and sub-agents are explained and elaborated in below. FMAS architecture is shown in Figure 4.1.

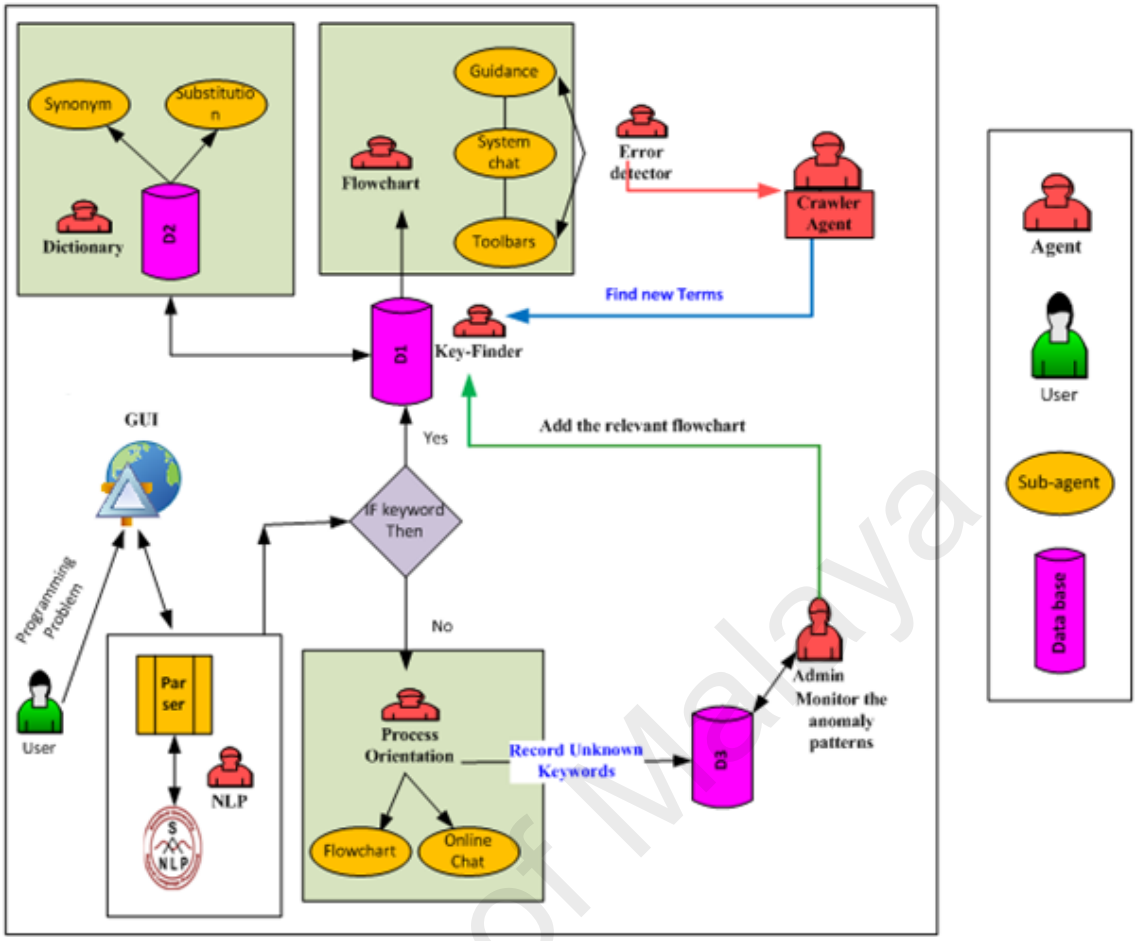

Figure 4.1: FMAS architecture

4.1.1 The First System Scenario

The interface that a user employ to interact with other agents named GUI. This software component takes the entered programming problem, in English text, and sends it to the NLP agent. The flowchart and process orientation agents use the GUI to pass the drawn flowchart or sub-flowchart along with instant feedback and messages to users using this component.

4.1.1.1 NLP agent

The semantic and syntactic analysis of the statement of programming problem will be performed by this agent. The main task of NLP agent is part-of-speech tagging and parsing in addition to sentence segmentation. Besides, the NLP agent also does noise

removal, removing prepositions, conjunctions, etc., and pass the main keywords to the key finder agent.

Example 1: *Write a program to calculate the factorial of a given number.*

After parsing we have: *write/VB a/DT program/NN to/TO calculate/VB factorial/NN of/IN given/VBN number/NN*

As illustrated, an online parser, Stanford Parser (Stanford, 2014), processed the entered text and the noise in the parsed sentence is removed automatically. The output of example 1 after the noise removal is:

Calculate/VB factorial/NN given/VBN number/NN

4.1.1.2 Key finder agent

The main words after the noise removal will be cross-checked with keywords stored in database 1 (D1) using key finder agent. It refers the keywords to flowchart agent for further processing in case a word matches a keyword. Otherwise, it refers the keywords to the dictionary agent for further checking. If no synonym or substitution is found, the second scenario of FMAS will automatically start working, see 4.1.2.

4.1.1.3 Dictionary agent

The main task of dictionary agent is to search the database 2 (D2) for any existing substitutions and synonyms. In case any synonym or substitution is found, it will be returned to the key finder agent again for further action. Dictionary agent consisted of two sub-agents, namely synonym and substitution.

- ***Synonym Sub-agent***

After the noise removal, the synonym sub-agent cross-checks the main words with its repository to find the possible words. In case any found, it will be returned to dictionary agent.

Example 2: *Write a program to find the largest among three numbers.*

For above example, if the key finder agent cannot find any keyword match for its question, it refers to the dictionary agent which, using a synonym sub-agent, will find synonyms such as biggest, maximum, and max for the main word 'largest.'

- ***Substitution Sub-agent***

The substitution sub-agent has the same function as synonym sub-agent and it will return the substitution to the dictionary agent, if there is any. However, unlike the synonym, this substitution may be a single word, a phrase, or a sentence.

4.1.1.4 Flowchart agent

The flowchart agent uses a draw-able representation module and the found keywords from D1 to provide the GUI with a user workspace, a system chat for step-by-step guidance, and the flowchart template. The system chat is used to offer guidance to student during the process and to generate the appropriate feedbacks. This agent comprises of three sub-agents as follow:

- ***Toolbar Sub-agent***

A flowchart template, workspace, different flowchart notations, and brief feedback next to each shape after flowchart completion are provided by toolbar sub-agent. For Example 1, the generated workspace, different shapes to complete the flowchart, system error upon flowchart completion along with the system chat are illustrated in Figure 4.2. As a classical way of making users think after a failure is to limit the amount of feedback, this sub-agent offers brief feedbacks to users on their tasks after the

completion of drag and dropping and not immediately. Moreover, as a user should enter a text into each shape in the flowchart template, this option does not fully guide the users.

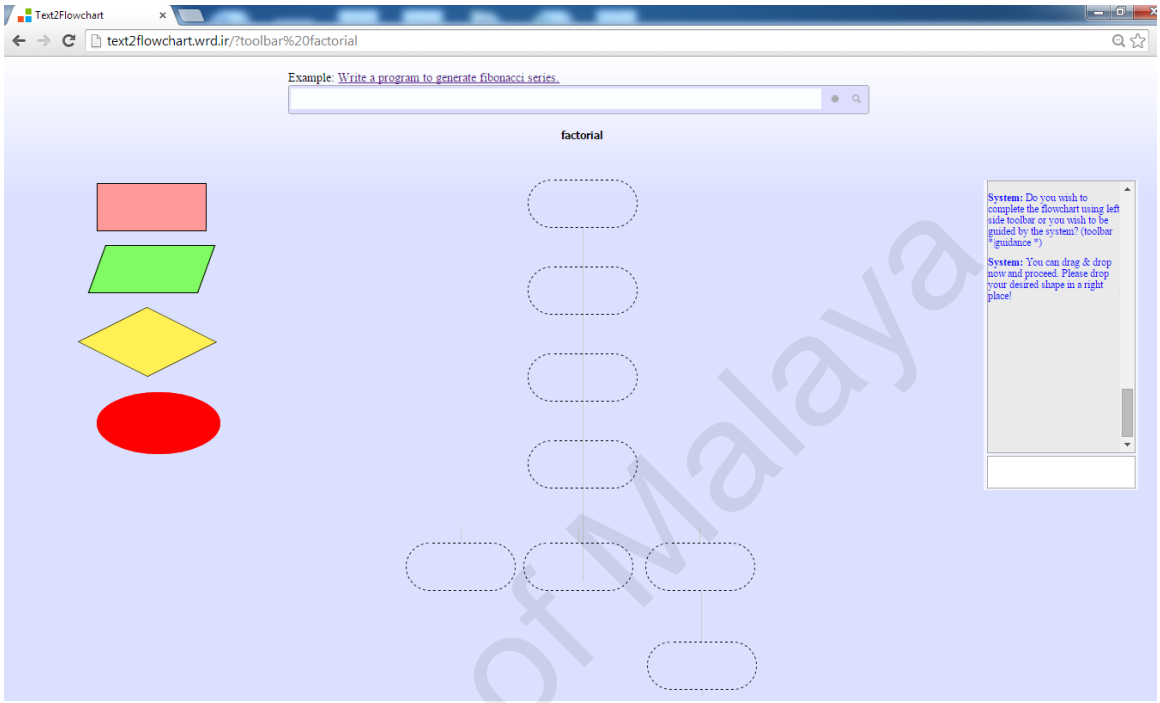

Figure 4.2: Workspace provided by the toolbar sub-agent

The automatically generated questions that ask whether the user wish to be guided step-by-step through the guidance option and the feedback upon flowchart completion are presented in Figure 4.3.

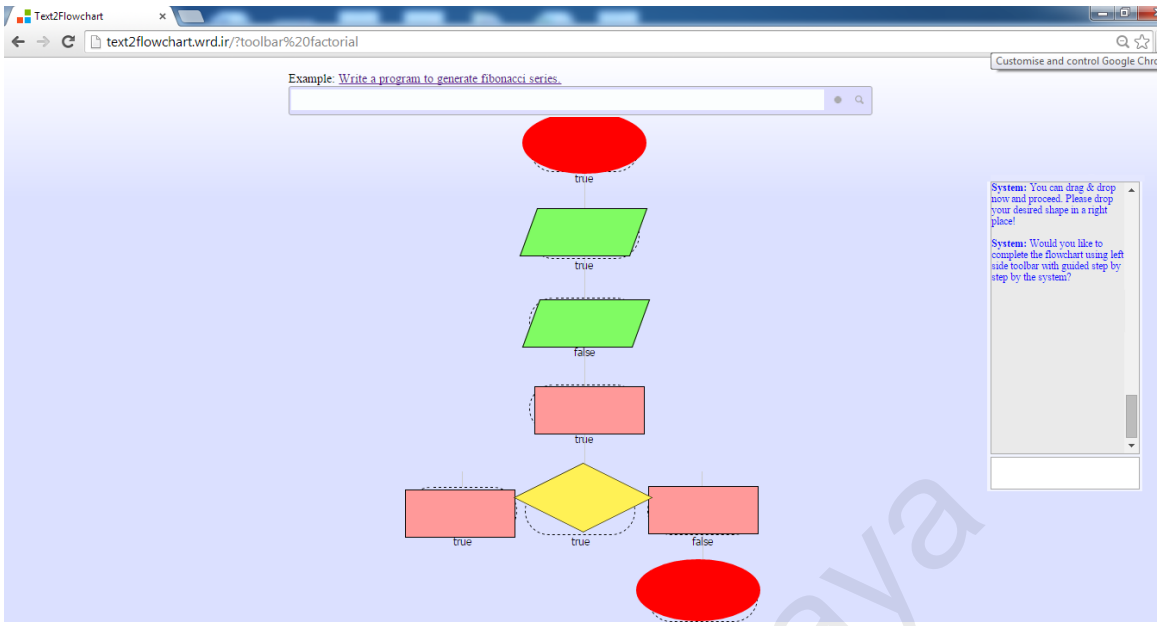

Figure 4.3: Toolbar sub-agent workspace with brief feedback

- ***Guidance Sub-agent***

A workspace, a flowchart template with sub-flowcharts located in the right places, various flowchart notations, instant feedback, extracting the correct text and content of each shape in the flowchart from D1 and placing it in the shape dropped by users, and showing the full flowchart to be compared and traced by users are provided for user by this sub-agent. The workspaces generated for Example 1 along with all aforementioned components are shown in Figure 4.4. It should be noted that this sub-agent provides users with an instant error and feedback and it does not allow the wrong shape or notation to be dropped in the flowchart template. While dropping the right shapes in the template, the relevant text extracted from D1 will be placed in the shape dropped to facilitate full user guidance.

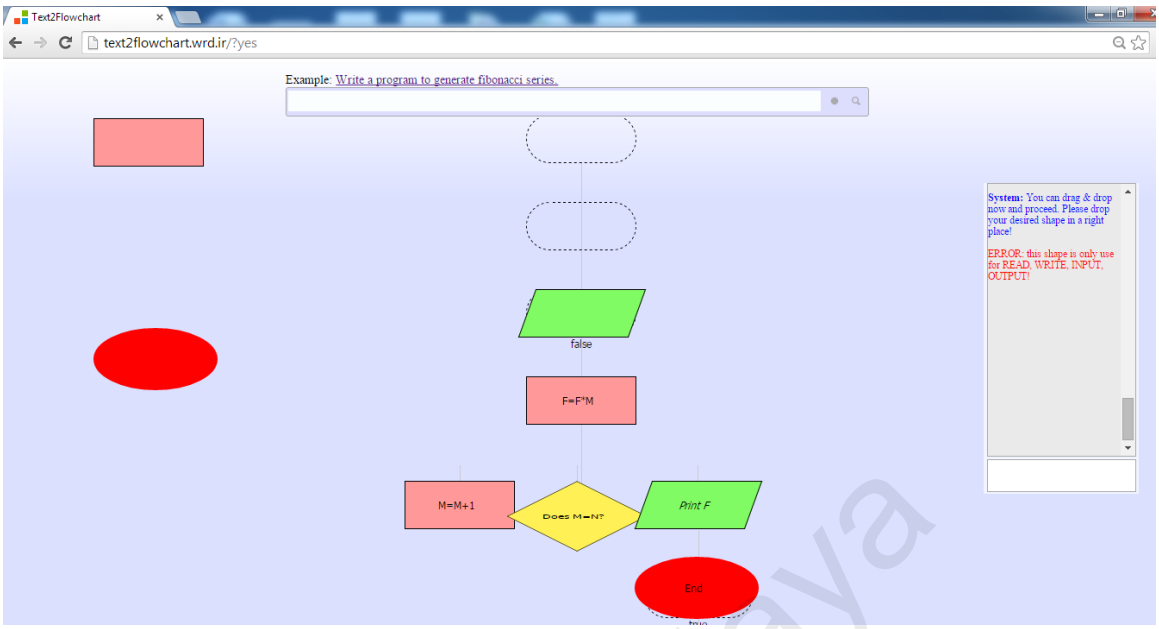

Figure 4.4: Workspace provided by the guidance sub-agent

As illustrated in Figure 4.5, the user will be asked whether they want to have a full flowchart extracted from the Internet. It enables them to observe the correct flowchart from the Internet, including arrows, next to the developed flowchart by the users. Afterwards, they could compare and trace flowcharts together.

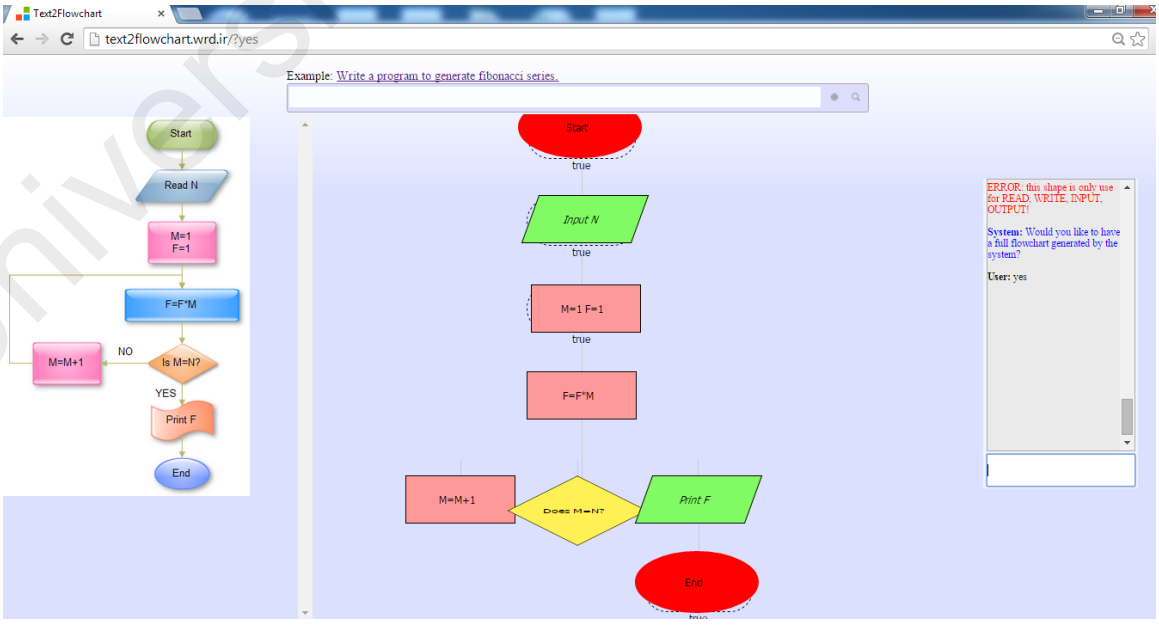

Figure 4.5: Workspace provided by the guidance sub-agent along with full flowchart from the Internet

4.1.1.5 System chat agent

The immediate and delayed feedback, errors and recommendations have been provided to users while they are completing the flowchart (Figures 4.4 and 4.5).

4.1.1.6 Error detection agent

This agent detects and stores any errors which occur during flowchart execution using the toolbar and guidance sub-agents. It will then send them to crawler agent who's responsible for finding additional and relevant information to improve the database for subsequent users without human intervention.

4.1.1.7 Crawler agent

This agent receives an unrecognized keyword from the error detection keyword and crawls relevant websites to find relevant information for automatically improving the D1 database. Once the additional information are extracted using this agent, they will be automatically added to D1 and shown for the next user who enters the same question before proceeding to flowchart completion; this added information are shown on top of the page in Figure 4.6, showing the function of these two agents in Example 1.

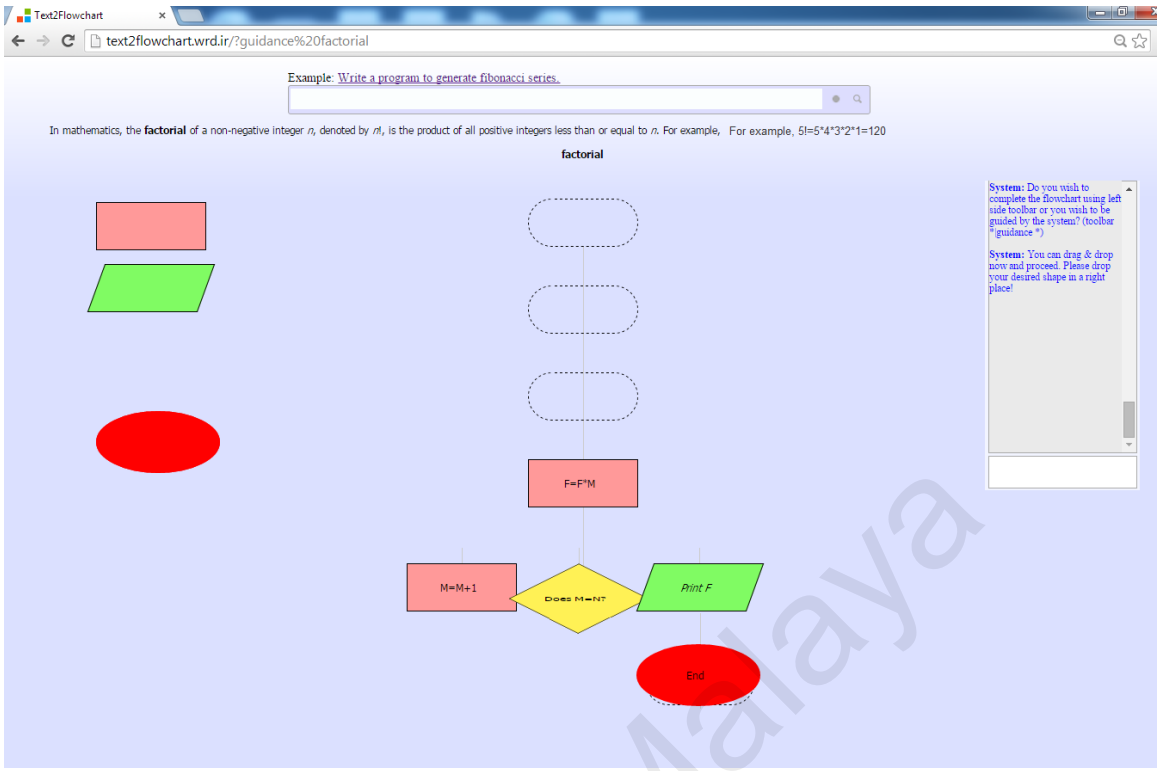

Figure 4.6: Workspace of the guidance sub-agent with extra information added by the crawler agent

4.1.2 The Second System Scenario

4.1.2.1 NLP agent

The GUI software component and NLP agent are the same as described in Section 4.1.1.

Example 3: *Write a program that asks the user to type an integer, and write _you win_ if the value is between 56 and 78.*

After parsing, there is*: write/VB a/DT program/NN that/WDT asks/VBZ the/DT user/NN to/TO type/VB an/DT integer/NN and/CC write/VB _/VBG you/PRP win/VB _/NNS if/IN the/DT value/NN is/VBZ between/IN 56/CD and/CC 78/CD*

As shown above, an online parser processed the entered text and the noise removal is applied on the parsed sentence. It should be noted that the noise removal stage in the second scenario of FMAS varies from the first scenario. Therefore, the output for example 3 after noise removal is:

Asks/VBZ user/NN type/VB integer/NN write/VB _/VBG you/PRP win/VB _/NNS if/IN value/NN is/VBZ between/IN 56/CD 78/CD

4.1.2.2 Key finder agent

The main words after the noise removal stage are cross-checked with keywords stored in D1and D2 by this agent. In case no match is found, the main words will be sent to the process orientation agent.

4.1.2.3 Process orientation agent

In case no keyword is found, this agent obtains the main words from the NLP agent, refers each related word to its corresponding flowchart notation (for example, if there is '?' or "If" in the problem statement, a diamond will be drawn, in which the flowchart sub-agent will place relevant words) and then sends them to the flowchart sub-agent for drawing. The process orientation agent includes two sub-agents as follows:

- ***Flowchart Sub-agent***

The flowchart sub-agent is responsible for generating sub-flowchart by referring each specific word and keyword to its corresponding shape. Similar to the toolbar option, this sub-agent provide the GUI with a workspace to complete the flowchart, a flowchart template and a system chat. However, unlike the toolbar option, this sub-agent offers an online system chat to guide users step-wise during the flowchart development. The relevant sub-flowchart for Example 3 is generated by this sub-agent and shown in Figure 4.7. In addition, it highlights the relevant sub-flowchart when a user holds a mouse curser on the statement of programming problem and vice versa so as to show the relationship between the flowchart and its corresponding text.

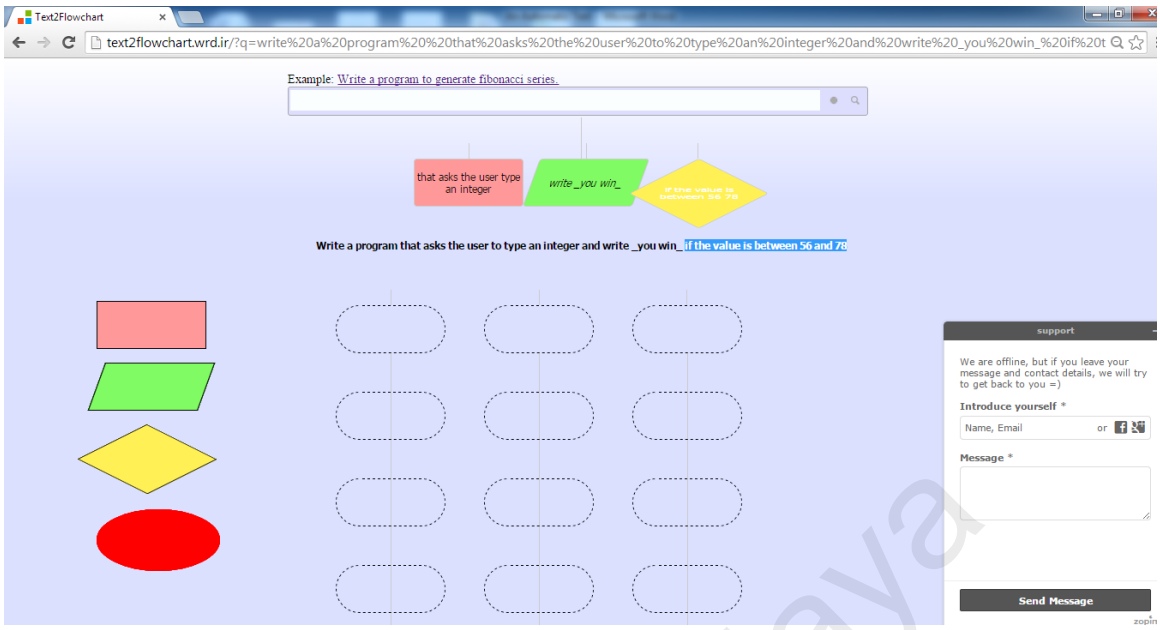

Figure 4.7: Workspace of the process orientation agent

- ***Online Chat Sub-agent***

As it frequently occurs that a user get stuck on a certain problem, this sub-agent is employed to get timely assistance from a tutor with the purpose of continuing the work. Otherwise, they may give up or not have sufficient time remaining. Thus, this sub-agent offers online chat with the system's admin; as illustrated in Figure. 4.7. However, it is worth mentioning that in case the D1 and D2 are fed properly with enough basic and imperative exercises aimed at novice programmers, this option will not be used frequently and this sub-agent is merely used in the worst-case scenarios.

4.1.2.4 Admin agent

This agent is employed to improve the system's database. It first request the system admin to develop appropriate flowchart for the unknown programming problems which are stored in database 3 (D3) and then will automatically add them into D1.

4.2 Summary

In this chapter, the rationale behind integrating a flowchart-based learning approach into the FITS knowledge domain was pointed out by considering the most challenging

part of FITS. This chapter started with the importance of mental modeling and visualization in the early programming stages before coding. The reasons why dynamic flowcharts could improve students' problem-solving skills were also highlighted. The flowchart-based approach integrated in FMAS development was then explained along with the main FMAS architecture. The main architecture of FMAS, which offers three different flowchart development options, i.e. toolbar option, guidance option, and worst case scenario, and some examples for each option were then elaborated. The next chapter provides details about the different FITS components, i.e. the student model and its implementation.

CHAPTER 5: DESIGN AND IMPLEMENTATION

This chapter provides an overview of the FITS design to fulfill all requirements defined in the previous chapters. The chapter first presents the tasks given in FITS to facilitate student problem-solving skills. It continues with the main architecture of the proposed ITS and the different components. This section discusses features that allow students to continue their work, the integration of a multi-agent system into FITS with the aim of providing a personalized and adaptive flowchart development environment for students according to their knowledge level, the user interface and FITS extensibility. The succeeding section demonstrates how different students can continue their work at a later time using FITS. Subsequently, the implementation details of FITS including software and tools are illustrated. The last section explains updating the student current knowledge.

5.1 The Given Tasks

The tasks given to student are grouped into three levels. The first level tasks are given when the student is doing the entrance test and the next two level tasks are given after the student has already answered the pre-test questions (the entrance test). Students are either learning programming concepts by clicking on light icons in the navigational menu or entering the given programming problem into the system and trying to develop its relevant flowchart with the help of FITS (see section 4.4).

5.1.1 The Entrance Test

The entrance test contains 20 multiple questions that are all connected to the programming concepts shown in Figure 5.3. The designed questions fall into three categories: variables, assignments and control structures. Each student is required to answer the questions to get the Bayesian network updated and start the next tasks based

on their answers. Thus, the subsequent tasks may be different for each student according to their programming knowledge.

5.1.2 Navigational Support Task

A navigational menu is presented to enable users to navigate through the concepts. Each concept is denoted by a specific color which represents the user's knowledge level. Red indicates that a user does not know about a specific concept and is not ready to learn, yellow shows that a user already knows the concept, while green indicates that the user does not know the concept but is ready to learn (see section 5.2.4). The next task is to navigate through the programming concepts. Based on the result from the first task explained in section 5.1.1, the navigational menu shows which concepts the student should approach first. Therefore, the second task is to turn as many green or red lights as possible to yellow so as to increase the student's knowledge of basic, imperative programming concepts. More yellow lights indicate greater knowledge of basic concepts that are useful for the next task.

5.1.3 Test Flowchart Development Task

The final task is to develop a flowchart for the given programming problem by completing the sub-flowchart (some parts of the flowchart, i.e. a diamond, rectangle, etc.) generated by FITS. For example, the student is asked to develop a flowchart for a programming problem. Meanwhile, the navigational menu shows a color related to Control Structures as red or green, meaning the student does not know the Control Structure concepts. The sub-flowcharts generated by FITS are different and more so for students who know less about Control Structure concepts. Afterward, the student will be asked to complete the flowchart by dragging and dropping the proper shapes. FITS automatically gives feedback and corrects the flowchart using different options (see

section 4.4). Thus, the result of each given task directly affects the next task in FITS. Except for the first task, the next two may differ for each student.

As previously discussed, the tasks presented to the student depend on his/her current knowledge (one of the processes performed by the teaching module – see 5.2.4). If the student can answers many pre-test questions correctly, it is possible the student already has a good understanding of the topics covered in FITS. In this case, the student can skip several tasks related to the topics that they understand. If FITS deems that the student does not have a good understanding of the topics related to the task, then FITS will extend the task based on that student's profile.

5.2 The Architecture of FITS

The engagement of leaners in solution designing activities from the very first stages of computer programming leads to improve their problem-solving skills. Therefore, in this thesis, an online game-based formative assessment is incorporated into a flowchart-based ITS with the purpose of boosting problem-solving skills and learning imperative concepts of programming. This results into two advantages in comparison with other advanced ITSs for learning computer programming. Firstly, incorporating a multi-agent system in FITS's domain model with the purpose of flowchart generation in an adaptive manner, see 5.2.4, extends the traditional model of ITS, contributing to more focus on problem-solving activities in the form of solution designing. Secondly, it applies an online formative assessment game to improve learning effectiveness and enhance student's motivation. The ability to navigate through the online learning materials in a game form in addition to recommend learning goals and generates appropriate reading sequences is offered by FITS. Moreover, using Bayesian network and a multi-agent system which benefits from an automatic text-to-flowchart conversion approach, see 4.4, in development of FITS provides learners with different flowchart development

options according to their level of knowledge. This feature, in addition to the online navigation of the learning materials in the form of a tic-tac-toe based game, significantly advances FITS in comparison to other related works in problem-solving skill improvement of students in the area. The FITS's architecture and entry page are shown in Figure. 5.1 and 5.2.

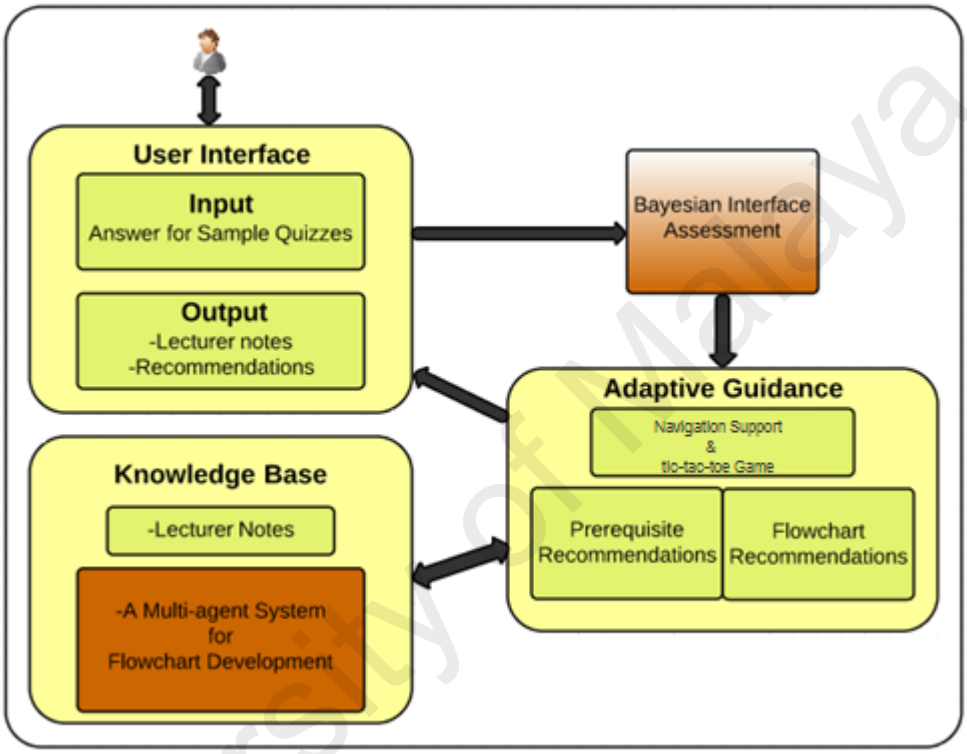

Figure 5.1: FITS's architecture

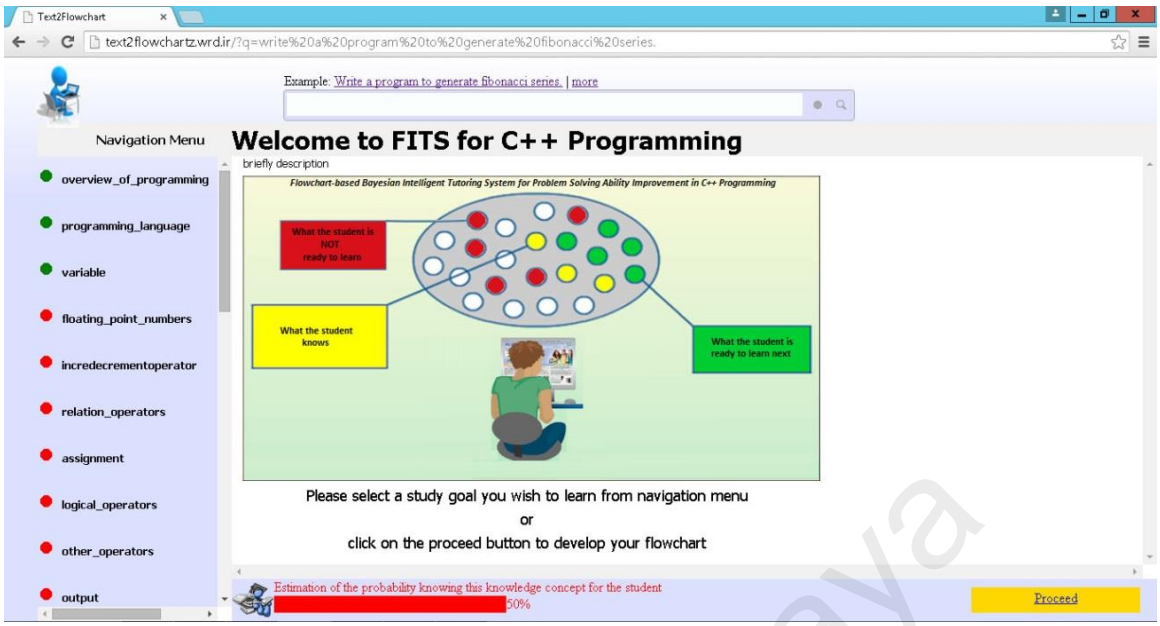

Figure 5.2: The Entry page of FITS

FITS is a completely web based system that can be accessed through a web browser. In order to use the system, each student must create a user name. They then login to the system using this user name and the relevant password. When a student logs in for the first time, he/she is required to complete a pre-test to gauge their current knowledge. A set of multiple choice questions form the pretest (see appendix B). In case the students do not know the answer to the question, they can leave it blank. It is even possible to not answer any questions if the student has no relevant knowledge. Once a student has completed the pre-test, she/he is directed to the main page, see figure 5.2. This page is also directly displayed on each subsequent login since the pre-test is only permitted once per student. The student selects a specific concept to learn and then enters the programming problem. The system provides students with appropriate feedbacks. The student is also permitted to abandon the current task and return to the task page at any time. The main advantage of this system is that it is web based. The system has currently been tested in various web-browsers, such as Opera and Google Chrome browsers. Therefore, it makes it possible for a multitude of users to access the ITS from different platforms.

5.2.1 FITS's Student Module

5.2.1.1 Determining the student current knowledge

The students' current knowledge in C++ programming will differ from one student to another. Presenting tasks that fit to their knowledge can encourage the students to do the tasks because they can see that the tasks are not too difficult or too easy to do. The first step that has to be done is to decide how to model each student's current knowledge or student model. The students' knowledge about C++ programming is represented by the probabilities of their understanding of the covered topics in FITS. The topics that are covered in FITS are Variables, Assignments, and Control Structures, see 1.4.

Various approaches in Artificial Intelligence have been proposed for uncertainty reasoning (Nilson, 1998), including rule-based systems (Buchanan & Shortliffe, 1984), fuzzy logic (Klir & Yuan, 1995), DempsterShafer theory of evidence (Zadeh, 1986), and neural networks. Bayesian networks (Pearl, 1988) are a powerful approach for uncertainty management in Artificial Intelligence (Wong & Butz, 2001; Wong et al., 2000). Overall, the reason for using this method so extensively is that it can accurately handle uncertainty. The theoretical basis for Bayesian Belief Networks (BBNs) is also highly developed and therefore it is expected to provide a relatively accurate student model, when sufficient observations are available. Therefore, a Bayesian Network is used to estimate the probability that the students know each of these topics. For that, the topics above are modeled as variables in Bayesian Network. Twenty multiple choice pre-test questions are used to measure the students' current knowledge before they use FITS. The questions in the pre-test can be seen in Appendix B.

All of these variables are represented as nodes in the Bayesian Network graphical model. Directed edges are then added to represent the relationship between the

variables. Figure 5.3 shows the Bayesian Network in FITS that represents the pre-test questions and the covered topics.

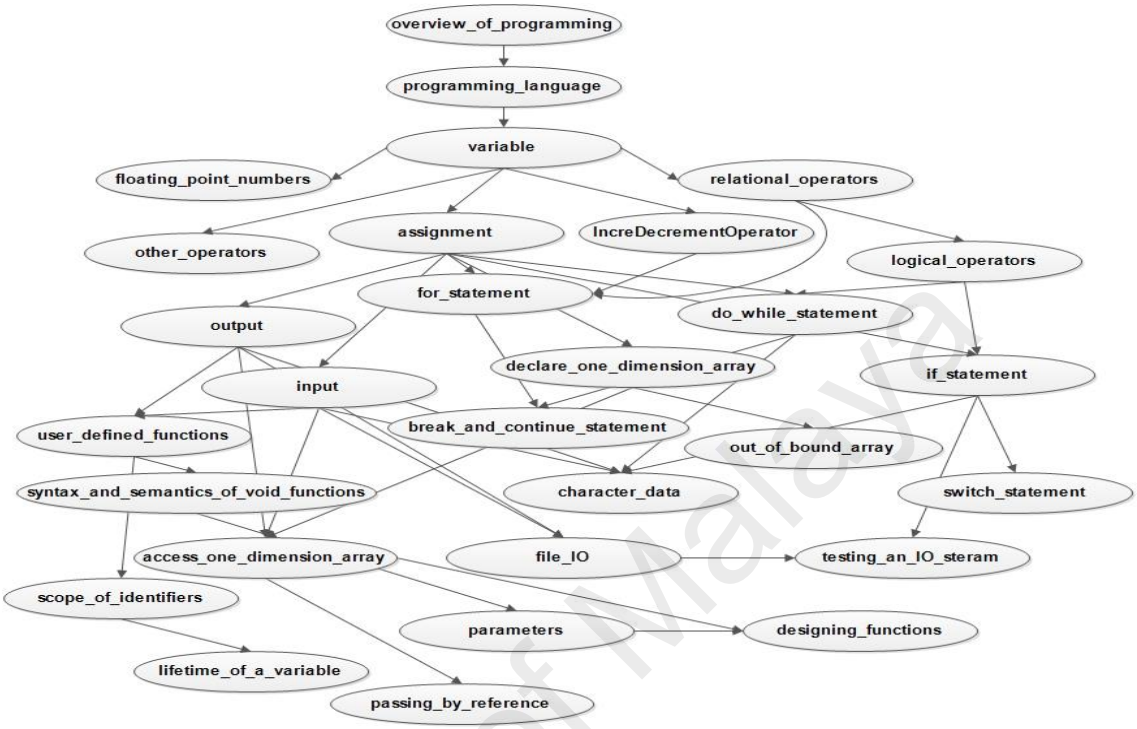

Figure 5.3: All programming concepts in FITS

The next step in designing a Bayesian Network is assigning the probability for each state that exist in a variable or node. All variables in FITS have two states only, known and not known which are used for the variables that represent knowledge of covered topics. The total probabilities for the states in a variable must be 100%. Assigning the probabilities value to the states of a node can be differentiated into two types depending of the type of the node, that is a parent and a child node. A node that does not have an incoming edge is called a parent node. There is only one node in Figure 5.3 that is set as parent nodes, that is "Overview_ of_ programming".

The probability of a parent node can be set directly without depending on the condition of the other nodes. In FITS, the known state of all parent nodes is assigned a 50% value. This number means that we 50% believe that the student knows the

overview of programming and 50% believe that the student does not know the overview of programming. The second type of the node in a Bayesian Network is a child node. For a child node, its conditional probability distribution has to be set, based on all the possible combinations of the parent(s) states. Table 5.1 shows the conditional probability distribution that is set for the child node “If_statement”. The parents of the node “If_statement” are “Assignments” and “Logical_operators” node. The conditional probability value for the child node must be created from the combinations of all states in its parent. Because each parent has two states, that is known and not known, there are four possible combinations that can be made from these states.

Table 5.1: Conditional probability for node "“If_statement”

Parent Nodes		**If_statement**	
Assignments	Logical_operators	Correct	Incorrect
Known	Known	0.8	0.2
	Unknown	0.4	0.6
Unknown	Known	0.4	0.6
	Unknown	0.2	0.8

The conditional probability in Table 5.1 shows that:

- It is 80% believed that the student knows “If_statement” concept correctly if the student already knows the “Assignments” and “Logical_operators “.
- It is 40% believed that the student knows “If_statement” concept correctly if the student already knows the “Assignments” but does not know “Logical_operators “. The same percentages of belief are used when the student does not know “Assignments” but already knows “Logical_operators “.
- It is 20% believed that the student knows “If_statement” concept correctly if the student does not know both “Assignments” and “Logical_operators “.

Based on the formula for a Bayesian Network, the system can update its belief for the variables in the network based on the occurrence of new evidence. For example, the belief that the student knows a particular topic can be updated based on the correctness or incorrectness of his/her answers to the pre-tests questions that are related to that topic. The changing of the belief probability in one variable will also influence the belief probability in other variables that are related to this variable. After the student has completed the pre-test, the probabilities of the student understanding of all topics will be known. This information is regarded as the student initial knowledge before using FITS.

Bayes' Theorem is used to update the probability value of a node given the occurrence of evidence. The theorem states that:

$$P\,(M \mid E) = \frac{P\,(\mathrm{M} \mid \mathrm{E})}{P\,(\mathrm{E})} \,.\, P\,(M) \quad \text{(Eq. 1)}$$

Where:

P(M) : the probability of M before E is observed

P(M|E) : the probability of M given E (after E is observed)

P(E|M) : the probability of observing E given M

P(E) : model evidence

We can calculate the value of P(E) by enumerating all possible state of M such as:

$$P\,(E) = \sum_m P\,(E|M_m) P\,(M_m) \quad \text{(Eq. 2)}$$

For example, let's say we have the probability value for the parent node "Variable" and a conditional probability distribution for node "Floating point numbers" (see Table 5.2 and Table 5.3)

Table 5.2: Probability value for node " Variable"

Variable	
known	unknown
0.5	0.5

Table 5.3: Conditional probability distribution value for node " Floating point numbers"

Parents Nodes	Floating point numbers	
Variable	Correct	Incorrect
Known	0.8	0.2
Unknown	0.3	0.7

According to the values in those tables, the probability that the student can answer question 1can be calculated correctly using the formula in equation 2 such as:

$P(\text{Floating point numbers}_t) = \sum_{m=(t.f)} P(\text{Floating point numbers}_t \mid \text{variable}_m)\, P(\text{variable}_m)$

$P(\text{Floating point numbers}_t) = P(\text{Floating point numbers}_t \mid \text{variable}_t)\, P(\text{variable}_t) + P(\text{Floating point numbers}_t \mid \text{variable}_f)\, P(\text{variable}_f)$

$P(\text{Floating point numbers}_t) = 0.8*0.5 + 0.3*0.5$

$P(\text{Floating point numbers}_t) = 0.55$

The correctness or incorrectness of the student answer to the question which is related to “Floating point numbers” in the pretest can be used as an evidence to increase or decrease our belief that the student already knows the topic about variables.

For example, if the student answers the question which is related to “Floating point numbers” correctly then using equation 1, we can calculate the new probability that the student already knows the topic about variables such as:

$P(\text{variable}_t \mid \text{Floating point numbers}_t) = \frac{P(\text{Floating point numbers t} \mid \text{variable t})}{P(\text{Floating point numbers t})}.\ P(\text{variable}_t)$

$P(\text{variable}_t \mid \text{Floating point numbers}_t) = \frac{0.8}{0.55}.\ 0.5$

$P(\text{variable}_t \mid \text{Floating point numbers}_t) = 0.73$

The result of this calculation shows an increasing belief (from 55% to 73%) that the student already knows the topic about variables given that he can answer the question which is related to "Floating point numbers" correctly. We can also use equation 1 to calculate our new belief that the student already knows the topic about variables if he answers the question which is related to "Floating point numbers" incorrectly, such as the following:

$$P(\text{variable}_t \mid \text{Floating point numbers}_f) = \frac{P(\text{Floating point numbers f} \mid \text{variable t})}{P(\text{Floating point numbers f})} . P(\text{variable}_t)$$

$$P(\text{variable}_t \mid \text{Floating point numbers}_f) = \frac{0.2}{(1-0.55)} . 0.5$$

$$P(\text{variable}_t \mid \text{Floating point numbers}_f) = 0.22$$

In this case, our belief that the student already knows the topic about variables drops from 55% to 22%, given the evidence that he cannot answer the question which is related to "Floating point numbers" correctly.

If we have a set of independent observations E = (e_1, …, e_n) for an event M then we can change the equation 1 to become:

$$P(M \mid E) = \frac{P(E \mid M)}{\sum_m P(E|Mm) P(Mm)} . P(M) \quad \text{(Eq.3)}$$

We can also find P(E|M) using the following formula:

$$P(E \mid M) = \prod_k P(ek|M) \quad \text{(Eq.4)}$$

For example, let's see the probability that the student already knows the topic about variables given the evidence that the student can answer the question which is related to "Floating point numbers" and the question which is related to "Other operators" correctly. Both of these questions are the child nodes of the node that represent variables.

To calculate this probability, we will use the probability value shown in Table 5.2, Table 5.3, and Table 5.4. The value in Table 5.4 shows a conditional probability distribution of the node "Other operators". By applying the values in these tables to the formula in equation 4 we will get a 91% probability value that the student already knows the topic about variables (see below). This number shows that our belief that the student already knows the topic about variables is increased highly (from 55% to 91%), given the evidence that the student can answer questions related to "Floating point numbers" and "Other operators" correctly.

Table 5.4: Conditional probability distribution of node " Other operators "

Parents Nodes	**Other operators**	
Variable	Correct	Incorrect
Known	0.75	0.25
Unknown	0.2	0.8

$$P(\text{variable}_t \mid \text{Floating point numbers}_t \text{ Other operators}_t) = \frac{P(\text{Floating point numbers t} \mid \text{variable t}) P(\text{Other operators t} \mid \text{variable t}).P(\text{variable t})}{P(\text{Floating point numbers t}|\text{variable t}) P(\text{Other operators t}|\text{variable t}).P(\text{variable t}) + P(\text{Floating point numbers t}|\text{variable f}) P(\text{Other operators t}|\text{variable f}).P(\text{variable f})}$$

$$P(\text{variable}_t \mid \text{Floating point numbers}_t \text{ Other operators}_t) = \frac{0.8*0.75*0.5}{0.8*0.75*0.5+0.3*0.2*0.5}$$

$$P(\text{variable}_t \mid \text{Floating point numbers}_t \text{ Other operators}_t) = 0.91$$

After doing the pre-test, the student is ready to use FITS. In this state, their current knowledge model is updated from the capability of completing the tasks (In this case, after reading the learning materials of concepts which are marked with green, meaning unknown but ready to learn, he/she should play the game and once he/she wins, the green light will automatically turns to yellow, meaning the concept is known). By doing so, the students unknowingly updates the Bayesian network by only playing the game.

5.2.1.2 Personalized learning

To satisfy the aim of suggesting one-to-one environments in a structure, the interaction of the users with the system needs to be supervised in order to implement a

profile for every one of them and to bring the Bayesian network up to date. In FITS, two different methods are employed in order to gain the profile from the users. The first approach is to answer the given questions before the entry page of FITS (pre-test). The second approach is to play tic-tac-toe game after reading the relevant learning materials. In Figure. 5.4, the learning materials of the "Assignment" concept are exhibited as well as the button for the game. If the user wins the game, the Bayesian network is updated and the navigational menu is displayed one more time with the concept labeled as known (yellow light), see 5.2.4; but if the user does not win the game, the concept will be labeled as unknown (red or green light) and he/she will be advised to revise the learning materials related to the concept. Meanwhile, the Bayesian network will be updated. Hence, the Bayesian Network algorithm will constantly be updated and by referring to the degree of the knowledge of learners, they receive the recommendation of respective learning materials and sub-flowcharts. Regarding sub-flowcharts, the learners are asked to complete the generated sub-flowcharts by FITS while they are provided with workspace and editor, system-chat, visualization notations, instant feedback, and the necessary guidance.

Figure 5.4: Learning materials and the game button

5.2.2 FITS's Knowledge Module

The knowledge base is classified into two distinct parts. The first section consists of the lecture notes in web page form, a repository of sample quizzes and questions along with their key solutions; while the second section includes a flowchart-based multi-agent system, see 4.4. The learning materials, including lecture notes, quizzes, and their key solutions, are presented when a user learns a new concept, whereas tic-tac-toe game and quizzes are introduced to decide if the user has understood a certain concept or not. It is worth mentioning that these learning materials are stored by their specific programming concept which refers to the relevant nodes in the Bayesian network, see Figure. 5.3. By doing so, knowledge concepts from actual instructional contents are separated which brings some advantages such as: 1) different instructors can work independently and write various parts of the instructional materials, allowing them to include information resources located anywhere on the World Wide Web; 2) the student's knowledge state can be estimated by applying the Bayesian networks as the inference mechanisms; 3) the changes to the content of learning materials in the tutoring system could be facilitated by this separation mechanism, allowing the easy modification of the instructional contents without having to reformulate the whole system. Regarding the flowchart-based multi-agent system, the leaners should enter the statement of their target programming problem into the system, and then they will be provided with three different flowchart development options based on their level of knowledge, namely the worst case scenario, the guidance option, or the toolbar option, see 4.4.

5.2.3 FITS's Communication Module

This section describes the design of FITS's user interface. This user interface is used as a communication module between user and the system. The design of the user

interface in FITS is aimed at encouraging students to use the application. Therefore, this user interface should be simple (to avoid overwhelming the student with many new things), self-explanatory, and requiring not many steps to operate it. Although simple, the design must be able to deliver all the functionality designed into FITS. These functions are:

1. Knowing who is the user that uses FITS
2. Allowing the student to continue their work after login
3. Providing the pre- and post-test
4. Providing tasks to the student
5. Differentiating between recommended and not recommended tasks
6. Differentiating between completed and not completed tasks
7. Allowing the student to select the task
8. Providing feedback and help

All of these functions can be accessed through FITS's windows that are discussed in the following sections.

5.2.3.1 Login window

The login window is used to identify the user who uses FITS. Details of the user's previous work are retrieved after the user logs in successfully. This login window also allows each new user to register into the system by selecting an ID and password. Figure 5.5 shows the design of this window. Because this window has two functions (to register and login), the text on some controls is adjusted to represent the active function. The information from the check box "Register" is used to determine the active function. After a new student registers successfully, the next window that is displayed is the pre-test window, see Figure 5.6. However if an existing user enters the system by logging-in, the next window that is displayed is the FITS's main window.

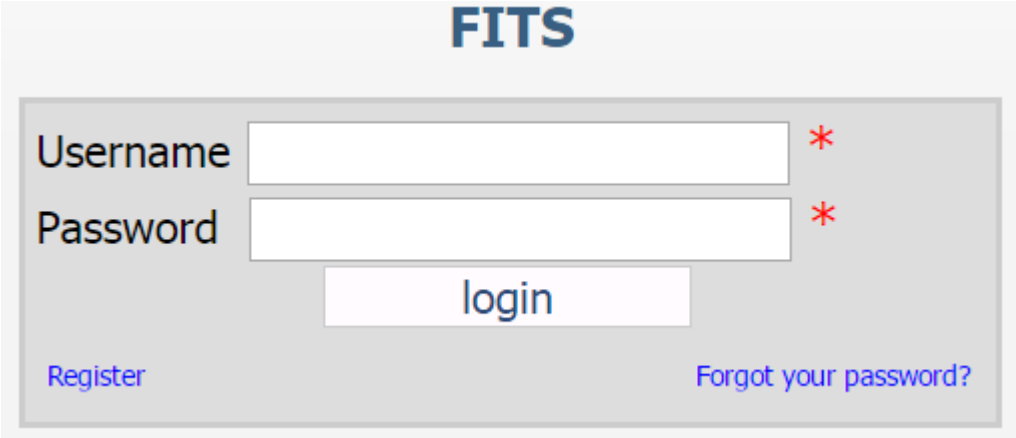

Figure 5.5: Login window

5.2.3.2 Pre-test and post-test window

The pre- and post-test windows are used to show the pre- and post-test questions. Because the pre- and post-test questions are the same – except for the ordering of the questions, then these tests are shown using the same window. The pre- and post-test questions are not static. The tests are stored in a database and so can be added or removed by a teacher, by modifying that database. However, adding or removing a test will affect the Bayesian Network student model. Therefore, the teacher must also modify the Bayesian Network model. This can be done through MSBNx, a software packages that support reasoning in Bayesian Networks, see 5.3. The questions and answers are added programmatically to this window, based on the data stored in the database. A rich text box is used to present the questions and a collection of radio buttons for the answers. Figure 5.6 shows the pre- and post-test window after being filled with some questions and answers.

Only one pre-test and one post-test can be performed by the student. The pretest window is shown automatically whenever a new student completes registration. On the other hand, the post-test window is displayed when the student presses the “Do the post test” button in the FITS main window.

1. _____ is the code with natural language mixed with some program code

- A program
- A statement
- Pseudocode
- A flowchart diagram

2. To improve readability and maintainability, you should declare _____ instead of using literal values such as 3.14159

- variables
- functions
- constants
- classes

3. Which of these data type requires the most amount of memory?

- long
- int
- double
- long double

4. What is the correct value to return to the operating system upon the successful completion of a program?

- -1
- 1
- 0
- Programs do not return a value

5. What is the only function all C++ programs must contain?

- start()
- system()
- main()
- program()

6. What punctuation is used to signal the beginning and end of code blocks?

- { }
- -> and <-
- BEGIN and END
- (and)

Figure 5.6: Window for pre- and post-test

5.2.3.3 FITS's main window

FITS's main window is used to deliver all the remaining functionality that is not handled by the login window or pre-test window. When the main window is displayed, all information related to the student's work is retrieved and FITS will start at the point

where the student left FITS last time. The information itself is saved automatically based on events that occur in FITS, that is when the student asks for help, when the student learns a concept by playing the game, and when the student completes a task. Figure 5.7 shows the design of this window. In the left side of this window, a navigational menu that is used to present the knowledge level of each user in that particular concept. The information that is displayed in this navigational menu is:

- Yellow light represents the student already know the concept
- Red light represents the student does not know the concept and is not ready to learn
- Green light represents the student does not know the concept but ready to learn

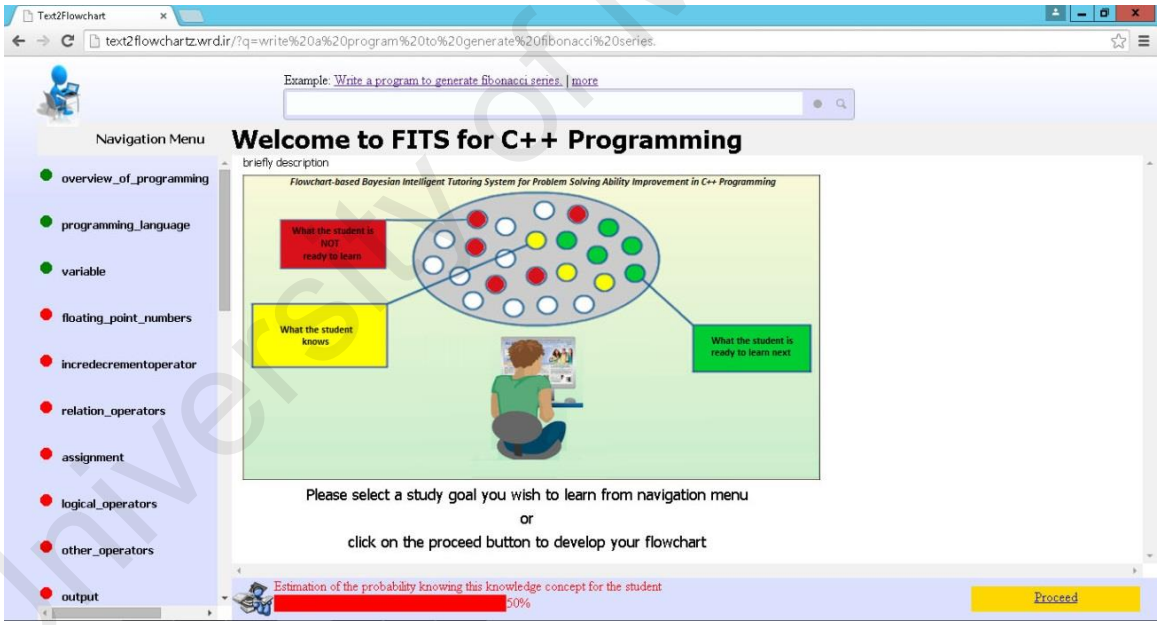

Figure 5.7: FITS's entry page with a navigational menu, offering study goals

At the bottom of the main window, each student can see the estimation of the probability knowing the knowledge concepts. At top of the main window, a text box is improvised where each student could enter the statement of programming problem and get involved in the process of flowchart development for the entered problem by the help of FITS. After entering the programming statement in the txt box, the student could

click in the ‘Proceed’ button to see FITS’s suggestion for flowchart development, see 5.2.4. By clicking on the light icons on the navigational menu, the student may see two different pages. In case the student clicks on the yellow or green lights, the learning materials of that specific concept will be shown along with the game button, see Figure 5.4. After going through the learning materials and reading them, the student clicks on the game button and game begins. If he/she wins, the light color will be turned to yellow, meaning the student knows the concept, see 5.2.4. And, in case the student click on the red light, he/she will be given prerequisite recommendations rather than learning materials, see 5.2.4.

5.2.4 FITS’s Teaching Module

Navigational support & tic-tac-toe game, pre-requisite recommendations, and flowchart development for problem-solving skills are three types of adaptive guidance offered to users by FITS.

5.2.4.1 Navigational support and tic-tac-toe game

Each concept in the navigational menu is marked with a different color. Yellow represents ‘already know the concept’, red represents ‘the concept is unknown and not ready to learn’, and green represents ‘unknown concept but ready to learn’, showing the knowledge level of each user on that particular concept. When the learner clicks on the lights (Navigation menu), the corresponding learning materials will be provided for the learner as well as a tic-tac-toe quiz for single-player (TRIS-Q-SP) game button (Figure.5.4). After pursuing all learning materials related to the concept, the user can start the game. The game-based formative assessment is shown as step 1 in Figure. 5.8. The basic regulations of TRIS-Q-SP are identical to the conventional tic-tac-toe game: whoever locates three adjacent tokens in a row, column or diagonally, wins the game. Regarding the TRIS-Q-SP procedure, once a user places the first token, he/she will

receive a random multiple-choice test question chosen from the database according to the learning content (Figure. 5.9). The user should then click on the correct answer. However, in order to raise the playfulness and complexity of the game, the regulations are somewhat modified from the traditional tic-tac-toe. If the player gives the right answer, they can place their token (Figure. 5.10); otherwise they will place the opponent's token (Figure. 5.11). Hence, the new game regulation prompts users to respond to the questions with more effort. Additionally, TRIS-Q-SP integrates three types of formative feedback. As seen in Figure. 5.10 and 5.11, a window will pop up once the user has finished responding to the questions and begins to move in the TRIS-Q-SP. Immediate knowledge of results (IKR) feedback is indicated on the right of the screen, notifying the user whether they had correctly responded to the prior question. The previous answered question and Immediate Elaborated Feedback (IEF) comment appear on the left side of the screen, showing corresponding information and hints to the last question. Moreover, Delayed knowledge of results (DKR) will be provided to students when they use the query answering history function after completing a game. DKR can be seen at left side of Figure. 5.10 and 5.11.

Figure 5.8: The TRIS-Q-SP

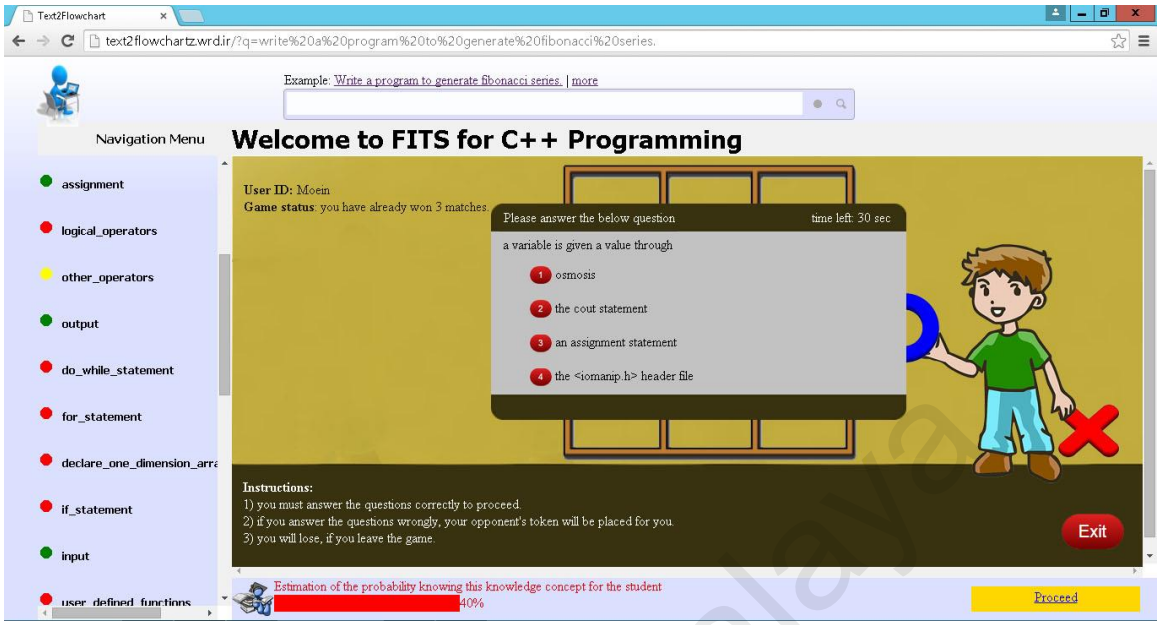

Figure 5.9: The random multiple-choice test question chosen from the database

Figure 5.10: The player gives the right answer

Figure 5.11: The player gives the wrong answer

5.2.4.2 Pre-requisite recommendations

In case the user clicks on the concepts with red, unlike green and yellow light icons, by considering the parent set of certain concepts in the Bayesian network, FITS recommends appropriate learning materials to the learners. For instance, since "Assignment" and "IncreDecrementOperator" concepts are parent nodes of the "For statement" concept in the graph, FITS proposes both to the user before permitting them to proceed to the "For statement" concept (Figure. 5.12).

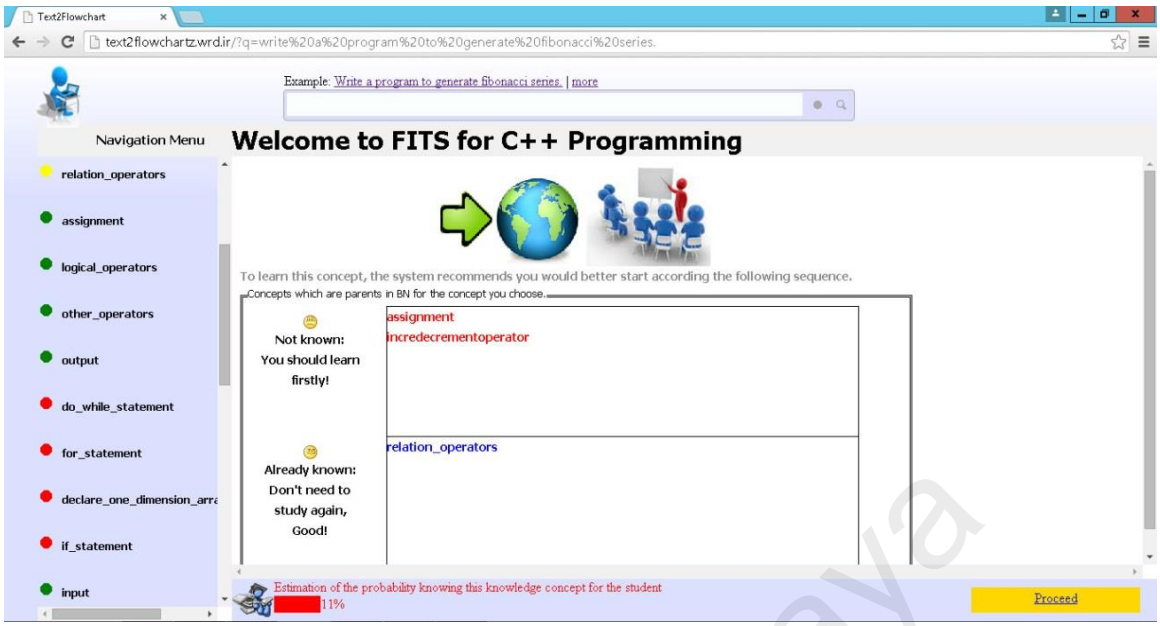

Figure 5.12: A pre-request recommendation generated by FITS

5.2.4.3 Adaptive flowchart development

The last and the most important adaptive guidance that FITS provide their users with is the personalized environment for flowchart development. It should also be noted with regards to the obtained profile and knowledge level evaluation by the Bayesian network for every programming concept, FITS is able to manage the guidance and sub-flowcharts in order to offer users an intelligent, customized and active setting for problem-solving improvement. Based on the estimation of the probability that a concept is known, different flowchart development options with various guidance levels based on the user's level of knowledge are proposed by FITS. For example, the guidance provided for user A for a programming problem such as "Factorial" is distinct from another user B; in Figure. 5.14, a sub-flowchart and guidance for user B are displayed, while in Figure. 5.13, a sub-flowchart for the *same* programming problem is shown, indicating more guidance for user A based on their profile.

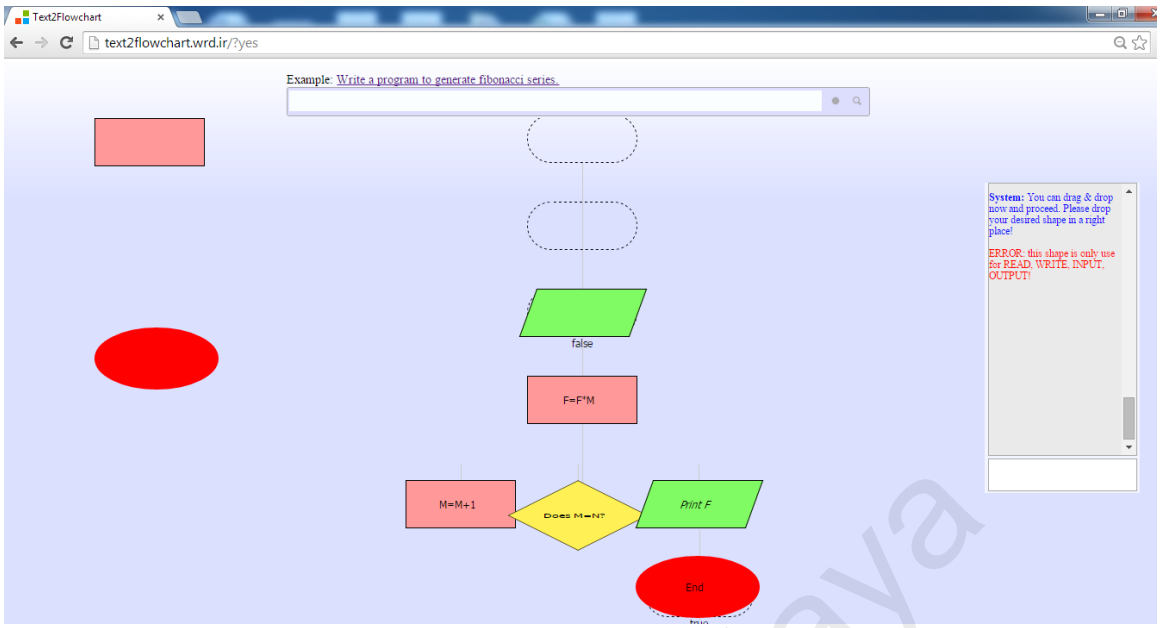

Figure 5.13: Workspace offered to user A based on the taken profile

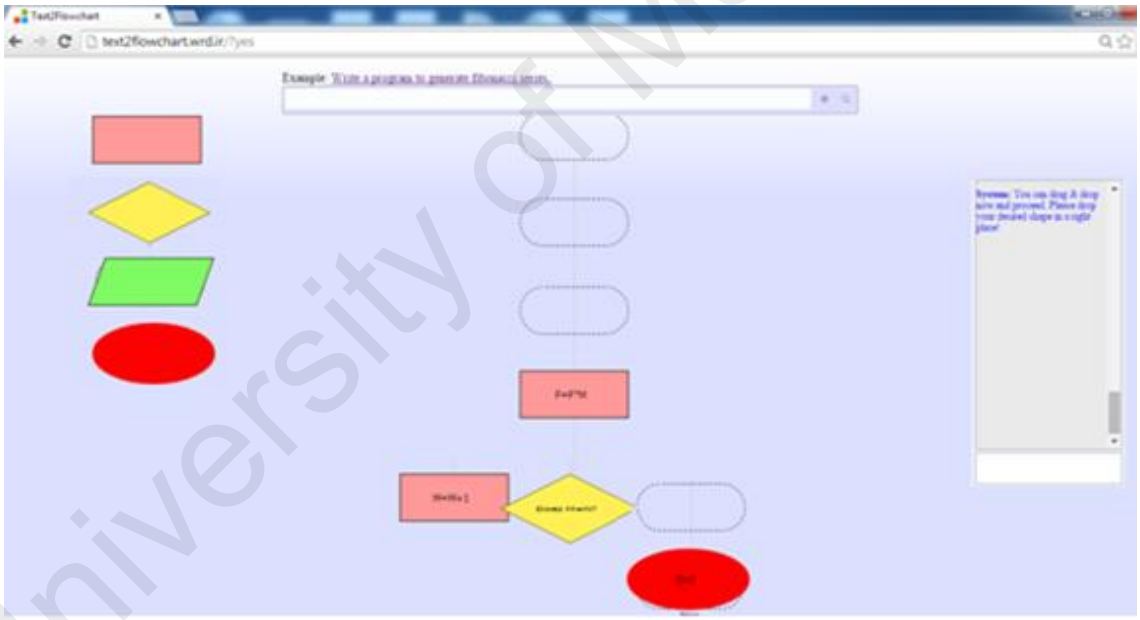

Figure 5.14: Workspace offered to user B for the same programming problem

5.3 Continuing to Work at a Later Time

The student's learning pace will differ from one student to another (and from time to-time for each student). One student may be able to finish all the tasks in FITS in one go but another student may need several days to finish the tasks. For the latter case, FITS is

required to remember the student knowledge model when they logout and reload this information when they login again to FITS.

A Bayesian Network model that represents the student knowledge is stored in a file that is created using an existing software package named MSBNx (Microsoft, 2013). This file contains the variables used in the model, the relationships between variables, and the probability values. When the student logs in, this model is read into memory from the file. Unfortunately, the items of evidence (about student knowledge) that are gathered by FITS cannot be stored in this file, because its format is specified by MSBNx. Therefore a database, see Figure 5.15, is used to store this information. When the student logs in, the structure and the initial probabilities values are read from the MSBNx file and the existing items of evidence are read from the database file (stored in the Student entity). The student model can be rebuilt thoroughly by recalculating the probability values of all variables using the formulae defined in section 5.2.1.

Having only the student model is not enough to bring FITS back to the same state as when the student left FITS previously. FITS also needs information that shows the student's current level and the tasks that have been completed by them. The student's current level and tasks are stored in Student Model entity. By combining the information from this entity and the information about the student's current knowledge model, FITS can return back to the same state as when the student left FITS.

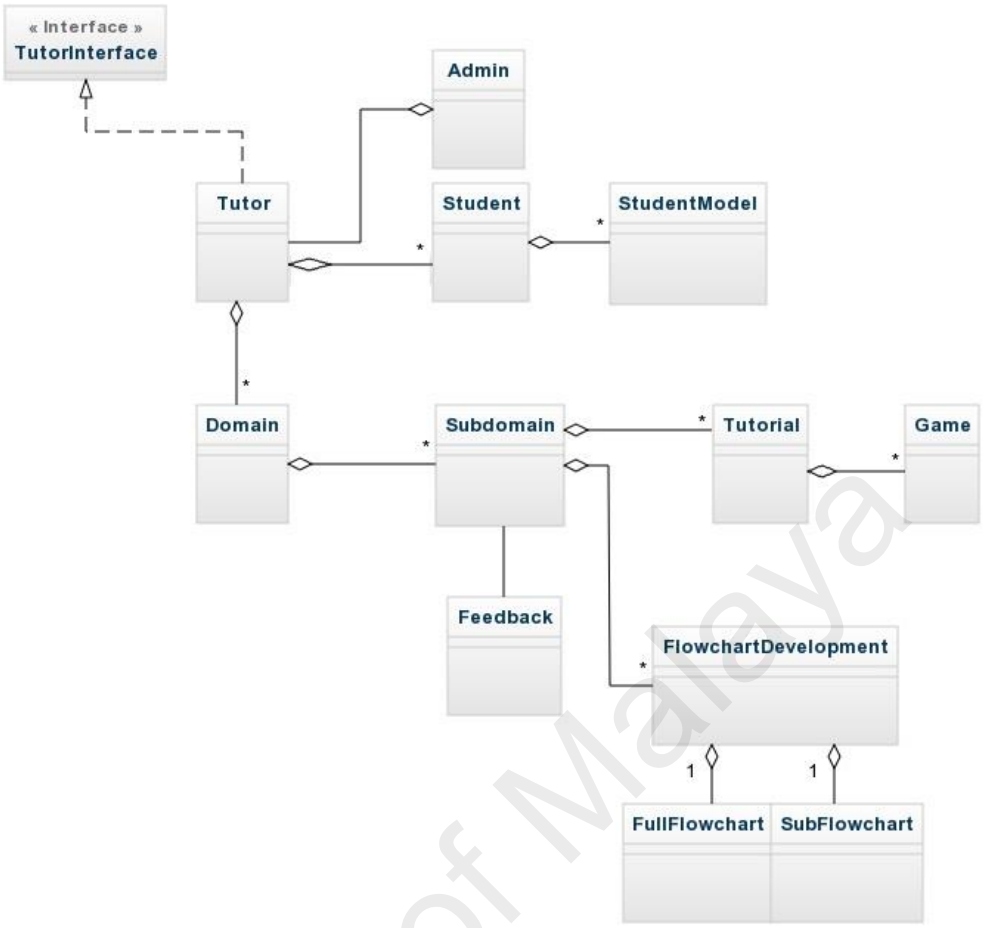

Figure 5.15: Database diagram

5.4 Implementation Details

This section presents the system implementation along with the software architecture. Figure 5.16 depicts the software architecture used for FITS development. As described earlier, the proposed ITS is a Web-based system and was therefore created using Web development languages. HTML was used to create the Web pages and CSS served to maintain consistent styles across the system. jQuery was used for processing the pages at the client side. The dynamic aspect of the Web pages was developed using C#/ASP.NET. C#/ASP.NET was selected over PHP owing to the fact that it is more efficient and reliable than some simple dynamic Web pages that could be developed using C#/ASP.NET. Moreover, ASP.NET provides built-in tools, functions and

controls that assist with faster code development compared to PHP. In addition, C#/ASP.NET was also used for flowchart development along with the Bayesian network in the student module. Regarding the Domain module, in order to develop flowcharts, it is necessary to have user inputs. The user should enter the commands related to the topic they want to learn. Before starting the process of generating the appropriate flowchart displayed by HTML, they should be processed. First of all, it is crucial to recognize if the user input is correct or not and inform the user in case the input is incorrect. For this purpose, regular expressions are used, where proper regular expressions are defined as language rules.

After using regular expressions to ensure the user's input grammar, a parser algorithm should be applied to extract different keywords and categorize them in defined groups to be used in the next step. Thus, the Stanford Parser is applied here to work out the grammatical structure of sentences with the aim of finding the keywords of the given programming problem statement. The result of this parsing process comprises defined system objects and shapes. In this step, a specialized markup language, Markdown, is defined to present the system calculation results graphically using HTML tags to make it easier for users to understand. Moreover, in text processing, the relevant flowchart components are found and linked by regular expressions. At the end, Markdown is applied to generate shapes adaptable and structured for HTML.

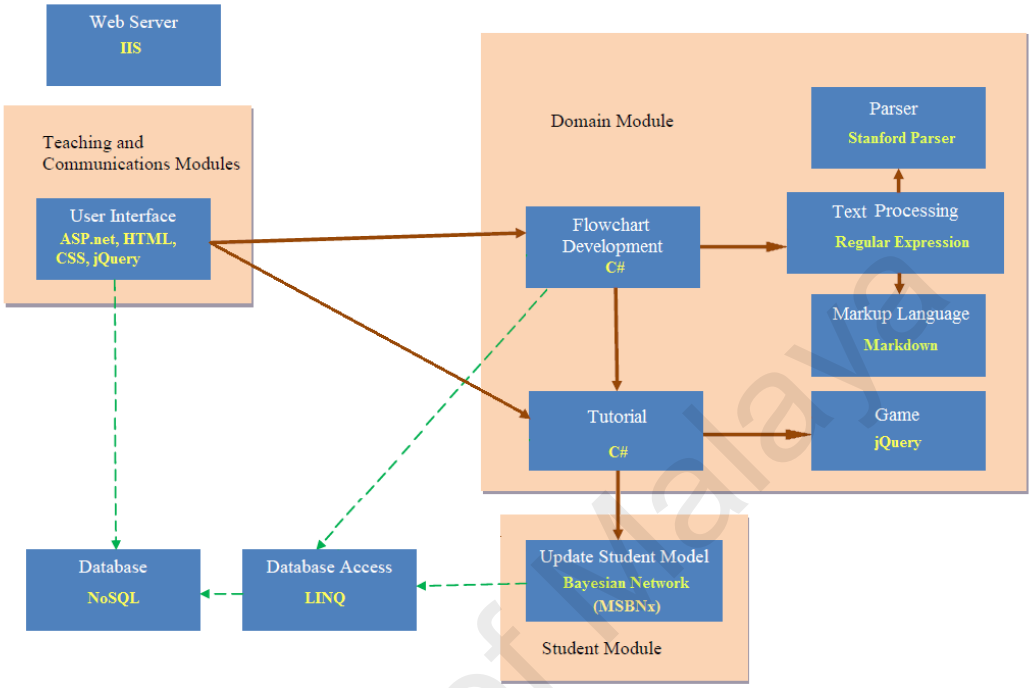

Figure 5.16: Software architecture used in FITS

In addition to flowchart development, an important part of FITS is updating of the student model. As explained earlier, this is done at the time of the pre-test and when the student completes the tasks. The model is also updated when the game is played and the student wins (section 5.2). Each student has a model in the Bayesian Network defined using MSBNx software. This model is dynamic and updated using C# codes while the user is utilizing the system. In one way, the user interface module, which is developed in the ASP.Net environment, is able to change the related Bayesian network. Also, the tutorial module can update the student model in order to make the necessary changes. This means that the program to update the student model needs to be called from both the interface (ASP.NET) and from the tutorial (C#). To ease access to this program from both these languages, MSBNx, which was used to create the Bayesian Network model to represent the student knowledge, was written in C#/ASP.net.

Several C# language versions are available and their functions slightly differ from each other. Here, the 2013 version is used through the powerful Visual Studeio.Net 2013 Integrated Development Environment (IDE). The main reason is that it allows working with many programming languages and tools employed in this project simultaneously. It is possible to open C#, ASP.net, HTML and jQuery with the languages in a single environment.

An imperative element of any software system is a database. NoSQL (non-relational) is used in FITS as the database management system. It includes different database technologies developed with the aim of responding to the rise in user data volumes stored, the frequency with which the data is accessed, objects and products, and performance and processing needs. Relational databases, on the other hand, were not designed to cope with the scale and agility challenges that modern applications face, nor to benefit from the economical storage and processing power available today. Thus, providing a mechanism for storage and retrieval of data that is modeled by means other than the tabular relation used in relational databases is the reason why NoSQL was chosen for FITS over a relational database. Moreover, design simplicity and finer availability control are other reasons for applying NoSQL.

Since Visual Studio.Net 2013 is already a matured IDE, it is easy to harness the power of different database technologies that use it. It is possible to connect to NoSQL through LINQ that is applied in C#. Furthermore, LINQ was developed by Microsoft in order to increase database-related codes' readability, which is the main reason it is used here. However, the direct connection between the user interface and database is through jQuery and the indirect connection is through C#. Visual Studio.net includes a native interface to connect directly to NoSQL. Therefore, the database was accessed directly from Visual Studio.net. However, it was also necessary to access the database through

the C# programs for flowchart development as well as for student model updating. LINQ was used for both flowchart development and student model updating in order to access the database in FITS.

Since FITS is a Web application, it needs to be deployed on a Web server. The Web server employed during development is Internet Information Service (IIS), which is installed on a Windows 2012. IIS is the main service recommended by Microsoft for developing ASP.Net applications. Furthermore, it is easy to manage Web sites, services and databases using this service.

5.5 Finding and Updating the Student Current Knowledge

As described in section 5.2.1, the initial student model is calculated by using Bayes theorem after the student completes the pre-test. This model is then updated whenever the student completes a task in FITS.

MSBNx is used as a software packages that support reasoning in Bayesian Networks. This software package is a component-based Windows application developed for creating, assessing, and evaluating Bayesian networks. Users can incorporate the components of MSBNx into their programs and benefit from inference and decision making under uncertainty. The diagnosis and troubleshooting inference that considers both observations and repair operations are provided by MSBNx. During diagnosis and troubleshooting, according to the value of information (VOI) computations, MSBNx can recommend what evidence to gather next. Besides from the components that this software package offer, the new add-in components can be created and be used within MSBNx by researchers and developers.

In FITS, the Bayesian Network model was created in MSBNx. This includes creating the nodes that represent the variables, setting the links between nodes, and entering the

probability values for all nodes. When the student complete the pre-test, the correctness or the incorrectness of their answers is used as evidence to update the probabilities values in the nodes, as described in section 5.2.1. The MSBNx is used to do the calculations. Unfortunately the current probabilities values in the nodes that were updated cannot be saved by MSBNx. Because of this, instead of saving the current probabilities calculation, FITS saves all the pieces of evidence that have occurred for the student, during their use of FITS. When the student logs in again, the model created in MSBNx is read along with the existing pieces of evidence from the database. The MSBNx is used again to recalculate the probabilities in all nodes to show the latest state of the student knowledge.

5.6 Summary

This chapter described in detail the system to be built. First, the different tasks designed in the proposed ITS for users were presented. Then the main architecture of FITS was elaborated, including the four different components. Continuing work at a later time was described, as learning pace differs from one student to another. In section 5.4, the proposed ITS implementation was explained, followed by determining and updating the student's current knowledge. The next chapter will address FITS evaluation.

CHAPTER 6: EVALUATION

The first section in this chapter gives a description of how the evaluation was performed, for both the first and second releases of FITS. Because we had a sufficient number of respondents for the second release only, this chapter reports the results from that second release. As stated in section 1.3, the main goal addressed by this research is to investigate if flowchart-based ITS model can help students to improve their problem-solving skills and attract their interest in learning basic concepts of computer programming effectively. The student pre- and post-test results were used to see if there was an improvement in the student knowledge before and after they used FITS.

Section 6.2 investigates this goal by evaluating the effectiveness of FITS across all respondents. The degrees of correlation between each of four variables (i.e. the length of time spent in FITS, the number of help requests, the number of times game played, and the number of tasks completed) with the student increase in score are evaluated. In section 6.3, the next goal of the evaluation which is to show which one of the student group that gets the highest benefit from FITS is illustrated through refining the evaluation by dividing the respondents into two categories: the respondents who had any background in programming; and the respondents who had no background in programming. This was done to see if either category got more benefit from FITS. The next section, 6.4, explains about the next goal which is to show attractiveness of FITS when learning to program. It describes the evaluation that was performed to see if FITS attracts student interest in learning programming. There are three issues that were used to measure if FITS can do this, namely its ease-of-use, its ability in helping students learn programming, and its learning approach attractiveness. In section 6.5, FITS's focus on problem-solving skills is described as the next goal and the final goal, section 6.6, describes student recommendations on possible extensions to FITS.

6.1 The Evaluation Process

The evaluation of FITS was performed according to the methods described in section 3.1. The instrument, procedure, timeline, and the participants are discussed respectively in sections 3.1.2 – 3.1.5; the evaluation process of FITS is shown in Figure 6.1. As described in section 3.1.5, it was not possible to have a control group, for ethical reasons.

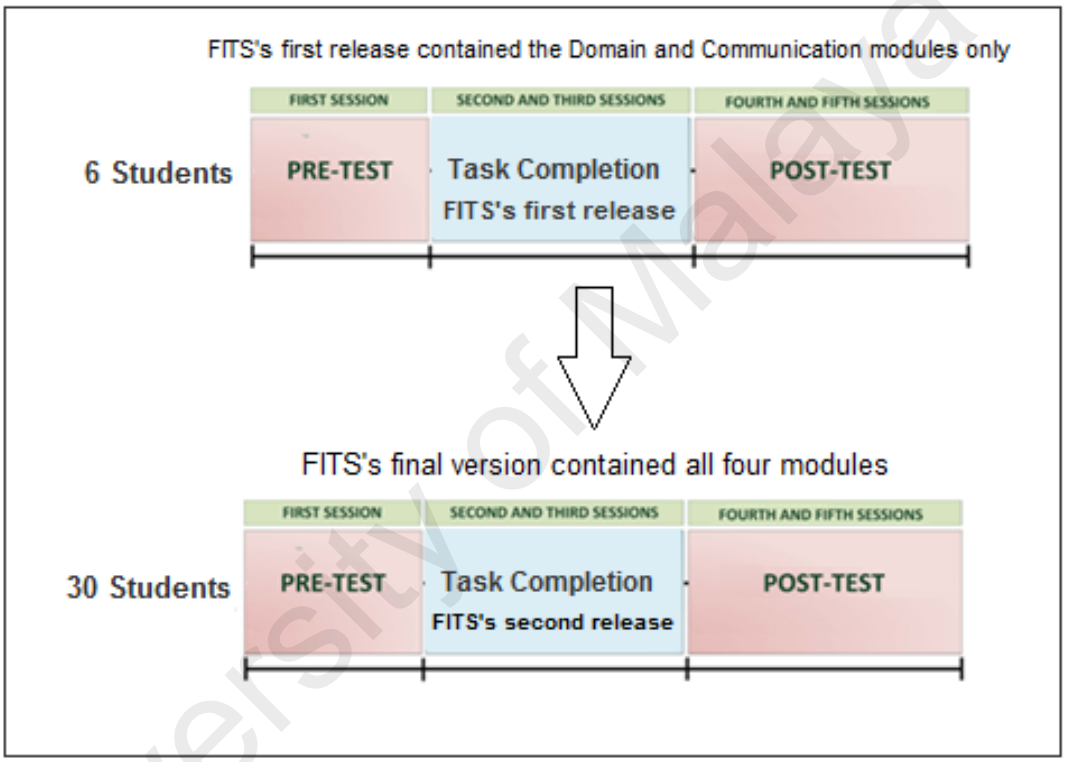

Figure 6.1: The evaluation process of FITS

The effectiveness of FITS in helping students learn the basic programming concepts and improving their problem-solving skills is measured from the performance of the participating students, before and after they use FITS (see section 3.1.2). This measurement is cross-referenced with data from the students' work history, such as the length of time the students spent in FITS, the number of exercises completed, the number of time game played, and the number of help requests. When that data is aligned with the performance measurements, it can be determined whether FITS can effectively help the students learn to program or not. A separate, qualitative evaluation

from the students' questionnaire is also used as part of the evaluation. The details of how and when the questionnaire was administered are discussed in section 3.1.3. The use of two evaluation types in this research is considered to be strength because one evaluation can validate or invalidate the result from another evaluation. On the other hand, the evaluation may contain some limitations including, the usage of positive phrases for all items in the questionnaire and the small number of respondents. Due to these limitations this evaluation can be enhanced in future research. As stated in section 3.1.4, FITS was released twice and the detailed evaluation that is discussed in this chapter is mainly based on the second release of FITS.

6.1.1 The Evaluation of the First Release of FITS

Before discussing the evaluation of FITS for the second release, the evaluation of FITS for the first release has been described and how it affected the enhancement of FITS for its second release. When it was released for the first time, FITS contained the domain and communication modules only. The student and teaching module were not developed at that time. Therefore, the students had to do all the tasks (develop flowchart for all given programming problem) in FITS because the student model did not exist and so no method was available to select specific tasks for each student.

The pre-test and post-test in the first release of FITS was performed differently from the second release. In the first release of FITS, the pre-test was performed by asking the student to do all the tasks in FITS without any help from FITS. The different options of flowchart development (i.e. Guidance and Toolbar option, see section 4.4) were also not fully presented in the pre-test. The student could skip any task in the pre-test, if he does not know the answer. The student works were marked automatically by FITS and stored as the pre-test values in FITS database. After the student completed all tasks (skipping as many as they liked), FITS started its tutoring mode. In this mode, the Guidance and

Toolbar option were presented and the student was asked to do the same tasks again but this time with help provided. The student could not skip any tasks this time. The student works in this tutoring mode were marked and stored as the post-test values.

As described in section 3.1.4, only six students participated in this first release. From the six students, only four completed all the tasks in the pre-test and in the tutoring mode of FITS. Although the effectiveness of FITS in this first release cannot be measured, several valuable comments from the students have been received, see Table 6.1.

Table 6.1: Participants' comments of the first release of FITS

Number	Student's comments
1	"This would be a great program for students as it can be used while the Tutor may be busy with someone else. 10/10 would learn by this again."
2	"It is extremely clever. Being able to engage in the process of solution designing and flowchart development while observing relationship between the programming problem statement and its relevant flowchart is a great way to learn programming."
3	"The feature that I like most is Automatic text-to-flowchart conversion approach. If it was implemented more completely, it could be more effective."
4	"The worst case scenario option to draw some parts of the flowchart for all entered programming problem was very impressive. i.e: I liked the way this option highlights the problem statement while you click on the flowchart and vice versa."
5	"Microsoft should pay you lots of money to incorporate your program into theirs."

The students' comments were very encouraging and strengthened my belief that the research was going in the right direction. In addition, it had been also increasingly believed that a system like FITS would be enjoyed by the students and could improve their performance.

6.1.2 The Evaluation of the Second Release of FITS

Before the second release of FITS, some technical problems that were commented-on by the students were fixed. Additional features such as the selection of some possible methods for specific students, showing the percentage of each student knowledge for each specific concept and total for all concepts, giving students different basic and imperative programming concepts and ask them to learn them by game playing, showing their knowledge through interesting way (i.e. traffic lights), providing them with prerequisite recommendation option to learn which concept they should go through first, and connecting each student knowledge (traffic lights) to the flowchart development section in order to offer them a personalized and adaptive environment for flowchart development. The student and teaching modules were also added.

The way the pretest and posttest were performed in the first release was also reconsidered and completely modified for the second release. In the first release, the pretest was performed by asking the students to do all tasks in FITS (e.g. to develop a flowchart for all tasks). No help was provided by FITS. The students' task was checked, marked and stored as their pretest score. Then the students were asked to do the same task again, but this time they could ask for help from FITS (using the guidance and toolbar options). Their flowcharts with answers to the posttest were checked, marked and stored as their posttest scores. When the tests were designed, it was thought that giving the same tasks to the students in the pre-test and post-test was good. Therefore the students were provided with the same tasks in both tests. The tests also check the correctness of the students' flowchart because they used the same module in FITS.

The problem with this kind of pre- and post-test is that the students get too many tasks to be completed. The students become bored and too exhausted to do all the tasks in post-test. One of the students who used the program in the first release commented

that he did not do some tasks in the post-test because he thought that his flowchart in the pre-test for the same task was correct already. Therefore, there is a possibility that the student becomes demotivated to do the post-test because he gets bored or too tired to do it again.

In the second release, 20 multiple choice questions were provided as the pre-test. After answering the pre-test, the student starts using FITS in its tutoring mode directly (as described previously in section 3.1.3). The students first could see their prior knowledge by observing traffic lights and then carry on to complete the tasks and draw their flowcharts for the task provided (programming problem given). The student can ask for help if they do not know how to develop the flowchart for the given task. With the existence of the student and teaching modules, the number of tasks that must be completed by the student depends on his developing knowledge. When all of these tasks are completed, the same 20 multiple choice pre-test questions (but in random order), are presented as the post-test. There were 30 students who used the second release of FITS. Although it would have been better to have more students, only this number showed an interest and satisfied the necessary requirements for participation.

6.2 The Effectiveness of FITS

As stated in section 1.3, the main goal addressed by this research is to investigate if flowchart-based ITS model can help them to improve students' problem-solving skills and attract their interest in learning basic concepts of computer programming effectively. The student pre- and post-test results were used to see if there was an improvement in the student knowledge before and after they used FITS. The data were distributed normally, because the skewness values were from -1.77 to 1.68, and the kurtosis values were from -2.61 to 2.11 for all variables. Byrne (2010) stated that if the skewness value is between -2 and +2, and the kurtosis value is between -3 and +3;

multivariate normality of the data could be assumed. The average student score in the pre-test, shown in Table 6.2 is 36.33% and the average student score in the post-test is 59.50%. The students' pre-test scores were higher than anticipated for student who mostly did not have programming background. Because some of students did the pre-test in different weeks, it was initially thought that the students' pre-test scores were influenced by the material that was taught during the semester. An effect size of 0.52 was obtained which is considered as high effect size based on Cohen's Rules-of-Thumb (1988), whereby an effect size of about 0.20 is considered "small"; about 0.50 is considered "medium"; and about 0.80 is considered "large."

Table 6.2: The average student score in the pre-test, post-test, and increase in scores

Respondent number	Pre-Test	Post-Test	Increase in score
1	50%	70%	20%
2	20%	45%	25%
3	45%	65%	20%
4	50%	55%	5%
5	30%	25%	-5%
6	60%	80%	20%
7	25%	60%	35%
8	15%	45%	30%
9	20%	55%	35%
10	45%	65%	20%
11	50%	70%	20%
12	15%	55%	40%
13	35%	50%	15%
14	40%	35%	-5%
15	55%	75%	20%
16	45%	55%	10%
17	20%	75%	55%
18	50%	70%	20%
19	15%	60%	45%
20	55%	55%	0%
21	25%	50%	25%
22	20%	65%	45%
23	15%	45%	30%
24	55%	65%	10%
25	20%	60%	40%
26	55%	80%	25%
27	25%	45%	20%
28	30%	60%	30%
29	60%	80%	20%
30	45%	70%	25%

Average	36.33%	59.50%	23.17%

To measure this correlation, a Pearson's R test between the week when each student did the pre-test and their pre-test scores was performed. A Pearson's correlation is the most widely used when you want to find a linear relationship between two normally distributed (IBM SPSS Statistics 20 was used to obtain the results of all statistical tests in this research.) Pearson's R value of 0.240 (close to 0) with p-value of 0.202 (not less than 0.05) in Table 6.3 show that there is no correlation here. The students who did the pre-test in an early week may have as good as (or as bad as) pre-test scores as the students who did the pre-test in a later week. The multiple-choice type of the pre-test questions –enabling the students to make good guesses at the answers based on logical reasoning – may be the reason behind their relatively high pre-test scores.

Table 6.3: Correlation between week when each student did the pre-test and their pre-test score

Correlation: Week when each student did the pre-test – pre-test score

		Pretest_score
Week_pretest	Pearson Correlation	.240
	Sig. (2-tailed)	.202
	N	30

Another Pearson's R test was performed to see if there was a correlation between the week when each student did the post-test and their post-test result. Pearson's R value of -0.338 (not too far from 0) with the p-value of 0.06 (not much less than 0.05) in Table 6.4 shows that the relationship here is not strong. The negative Pearson's R value shows

that there is a tendency that the students who do their post-test in early week have higher post-test scores than the students who do their post-test in later week.

Table 6.4: Correlation between week when each student did the post-test and their pre-test score

Correlation: Week when each student did the post-test – post-test score

		Posttest_score
Week_posttest	Pearson Correlation	-.338
	Sig. (2-tailed)	.067
	N	30

After confirming that the students' pre-test and post-test scores were not significantly influenced by the week when they did the test (i.e. not significantly influenced by the material that was covered in class), the next step was to see if the students' post-test scores were significantly better than their pre-test scores. In order to answer this question, a two-tailed paired t-test with a 95% confidence interval was performed because the results are interesting in either direction, whether the total area of α is placed in one tail or divided equally between the two tails. Null hypothesis was employed to answer the research hypothesis in order to decrease Type I error (Cohen, 1988). In statistical hypothesis testing, a type I error is the incorrect rejection of a true null hypothesis (a "false positive"). The null hypothesis for this test is:

Hypothesis number 1: The student scores in post-test are not higher than the student scores in the pre-test.

The p-value of 0.0000 shown in Table 6.5 is the result of two-tailed paired t-test in SPSS. This p-value shows strong evidence against the null hypothesis. Therefore from the result of this t-test, it can be concluded that the scores in the post-test are significantly higher from the scores in the pre-test.

Table 6.5: Paired t-test of post-test and pre-test

Paired Samples Statistics

		Mean	N	Std. Deviation	Std. Error Mean
Pair 1	Posttest_score	59.5000	30	13.41319	2.44890
	Pretest_score	36.3333	30	15.75312	2.87611

Paired Samples Correlations

		N	Correlation	Sig.
Pair 1	Posttest_score & Pretest_score	30	.530	.003

Paired Samples Test

		Paired Differences					t	df	Sig. (2-tailed)
		Mean	Std. Deviation	Std. Error Mean	95% Confidence Interval of the Difference				
					Lower	Upper			
Pair 1	Posttest_score - Pretest_score	23.16667	14.29271	2.60948	17.82968	28.50365	8.878	29	.000

The student scores in the pre- and post-tests can also be displayed in a boxplot chart – see Figure 6.2. This boxplot chart easily shows that the student score in the post-test is better than the student score in the pre-test. The median in the post-test score is 60 and the median in the pre-test score is 37.5. The chart also shows that the first and the third quartile post-test scores are also better than the ones in the pre-test. After finding out that the student scores in the post-test were significantly better than the student scores in the pre-test, the factors in FITS which influence the student scores should be identified. There are several candidate variables that were investigated to look for correlations with increase in student scores. These variables are:

- The length of time spent by the students when using FITS.
- The number of help requests made by the students.
- The number of game played by students.
- The number of tasks completed by students.

The degree of correlation of each of these four variables with the increase in student scores is described in each of the following four subsections.

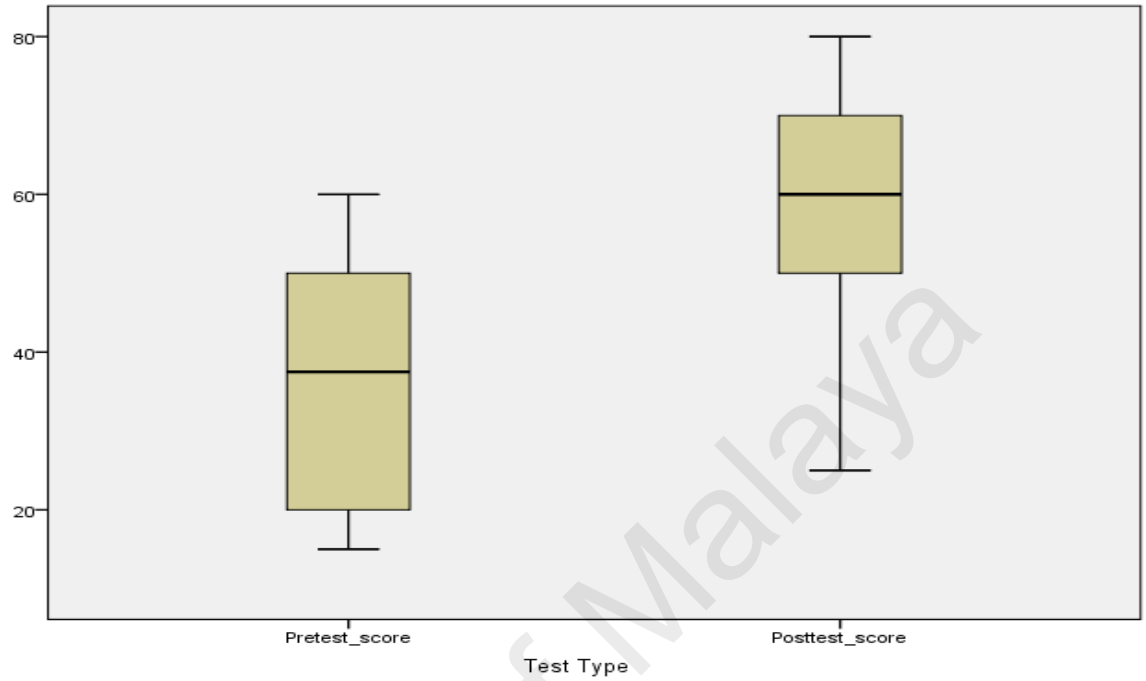

Figure 6.2: Boxplot chart of pre- and post-test scores

6.2.1 The Degree of Correlation between the Length of Time with the Increase in Student Scores

The length of time spent by the student using FITS seems to become a good candidate variable. There is a tendency to think that if a student spends more time in FITS then his increase in score will be higher than the student who spends less time in FITS. Pearson's R test was used to see if there was a correlation between the total lengths of time spent by each student with their increase in score. Pearson's R value of 0.062 in Table 6.6 shows a very weak relationship between the lengths of time used by the students to their increase in score.

Table 6.6: Correlation between length of time spent by students and their increase in score

Correlation: Length of Time Used – Increase in Score

		Increase_in_score
Time	Pearson Correlation	-.345
	Sig. (2-tailed)	.062
	N	30

As described in chapter 1, all students learn at their own pace. Therefore the length of time spent by one student cannot be compared with the length of time spent by another student. One student may spend two hours to get a 10% increase in score, another student may spend one hour to get the same increase in score, and another student may spend three hours. Based on this thought, it is not surprising that the lengths of time used by the students do not correlate with their increases in score.

6.2.2 The Degree of Correlation between the Number of Help Requests with the Increase in Student Scores

The next test was performed to measure the correlation between numbers of help requests made by the students with the students' increase in score. Pearson's R test was used again for this case. The correlation between the numbers of help requests to the increase in score is shown in Table 6.7. Pearson's R value of 0.106 (close to 0) with p value of 0.577 (bigger than 0.05) in that table shows a weak correlation.

Table 6.7: Correlation between number of help requests and the increase in score

Correlation: Number of Help Requests – Increase in Score

		Increase_in_score
Help	Pearson Correlation	.106
	Sig. (2-tailed)	.577
	N	30

The scatter graph in Figure 6.3 shows a more detailed view of the relationship between the number of help requests by the students and their increase in score. The weak relationship possibly comes from the existence of outliers in the data. The graph in Figure 6.3 shows that there are three students who can be categorized as help abusers and one student as a quick learner. The help abusers students in this case requested help more than 100 times (much larger than the average numbers of help requested by the other students) but only obtained an increase in score in the range of 0 – 30%. On the other hand the quick learner obtained an increase in score of about 55% with only four help requests.

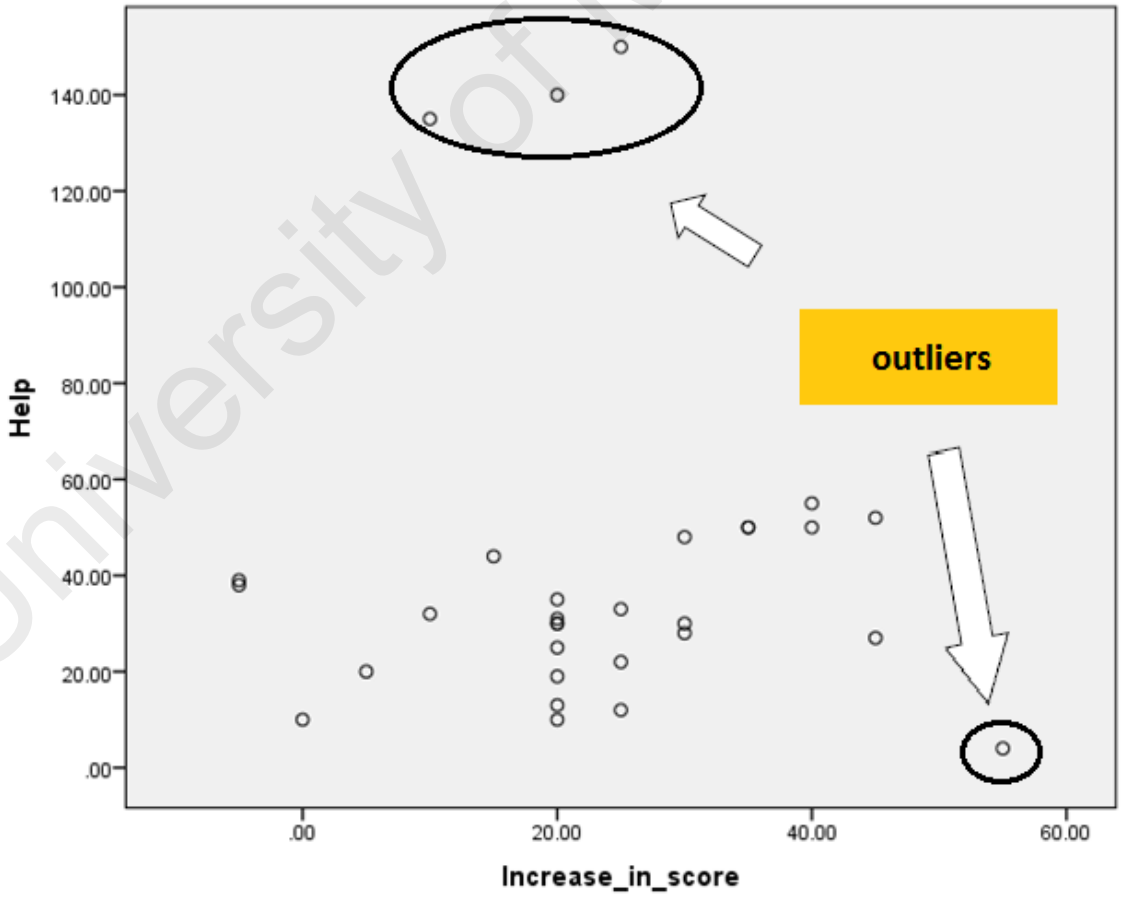

Figure 6.3: Scatter graph of increase in score vs. number of help request

When the data from these students are treated as outliers and the correlation between the help requests by the students and their increase in score is calculated again without them, then the correlation result is shown in Table 6.8. Pearson's R value of 0.411(far from 0) with p value of 0.037 (much less than 0.05) in that table shows a strong relationship between the number of help requests by the students and their increase in scores. (The positive Pearson's R value shows that the more help requests by students, the better their increase in scores.)

Table 6.8: Correlation between the number of help requests and increase (without outliers)

Correlation without outliers: Number of Help Requests – Increase in Score

		Help
Increase_in_score	Pearson Correlation	.411
	Sig. (2-tailed)	.037
	N	26

A more detailed test was performed to find the relationship between the student's initial knowledge and the number of help requests. The pre-test scores were used to represent the students' initial knowledge, that is the lower the score achieved, the less initial programming knowledge that the students have. The result of Pearson's R test for this correlation (without outliers) is shown in Table 6.9. Pearson's R value of -0.602 (very far from 0) with p value of 0.001(much less than 0.05) in that table shows a strong relationship between the students' initial knowledge and the number of help requests in FITS. The negative R value shows that the students with less initial knowledge ask more help in FITS.

Table 6.9: Correlation without outliers between the pre-test score and the number of help

Correlation: Pre-test score – number of help request

		Help
Pre-test score	Pearson Correlation	-.602**
	Sig. (2-tailed)	.001
	N	26

Recapping the results from Table 6.8 and Table 6.9:

- The lower the students' initial knowledge (the lower pre-test score), the more help requests they made (from FITS).
- The more help requests they made (from FITS), the better the students' increase in score (the better the students' final knowledge).

Based on these two statements, it has been concluded that FITS can help to improve the students' knowledge.

6.2.3 The Degree of Correlation between the Number of Times the Game Played with the Increase in Student Scores

The third variable to be checked is the correlation between the numbers of time the student play the game with the increase in student scores. The number of time that student needs to play the game varies based on their initial and current knowledge. As explained in section 5.2.4, if a light in the navigational menu is green, meaning the concept is unknown but the student is ready to learn, and student click on it, he/she will be given learning material and the game bottom. In case after reading the learning materials the student start paying the game and wins, the light on the navigational menu turns to yellow, meaning already know the concept. However, some students may click on the game and lose the game or withdraw for any reasons and the light still remains green, therefore, the relation between number of time game played and increase in

student score seems to become a good candidate variable. There is a tendency to think that if students play more game in FITS then his increase in score will be higher than the student who played less in FITS.

The result of Pearson's R test to measure this correlation is shown in Table 6.10. Pearson's R value of 0.835 in that table shows a strong relationship between the number of time game played by the students and their increase in score. The p-value of 0.000 – much smaller than 0.05 – shows that the numbers of game played by students have a significant correlation with their increase in score. It shows if student click on the game button after going through the learning materials, he/she would be able to complete and win the game and turn the light to yellow which leads to increase in students score.

Table 6.10: Correlation between numbers of time game played and increase in score

Correlation: Number of Time Game Played – Increase in Score

		Increase_in_score
Number of Time Game Played	Pearson Correlation	.835**
	Sig. (2-tailed)	.000
	N	30

6.2.4 The Degree of Correlation between the Number of Tasks Completed with the Increase in Student Scores

The final variable to be checked is the correlation between the numbers of task completed with the increase in student scores. The number of tasks that need to be completed by each student varies, based on their initial and current knowledge. FITS indicates the tasks that need to be completed by traffic lights which are either green or red, meaning the student does not know these concepts and they need to be turned to yellow color by reading the given learning materials and playing the respective game

for that specific concept, meaning the student know the concept now. Another way is to complete the given sub-flowcharts (generated by FITS for an entered programming problem) and turn them to a full flowchart.

The result of Pearson's R test to measure this correlation is shown in Table 6.11. Pearson's R value of 0.590 in that table shows a strong relationship between the number of tasks completed by the students and their increase in score. The p-value of 0.001 – much smaller than 0.05 – shows that the numbers of tasks completed by students have a significant correlation with their increase in score.

Table 6.11: Correlation between the number of tasks completed and the increase in score

Correlation: Number of Tasks Completed – Increase in Score

		Increase_in_score
Number of Tasks Completed	Pearson Correlation	.590**
	Sig. (2-tailed)	.001
	N	30

This correlation can be seen clearly in the scatter chart shown in Figure 6.4. This chart shows that students who completed more tasks in FITS have a tendency to get a higher increase in score. The information in Table 6.11 and the chart in Figure 6.4, strengthen the previous conclusion that FITS is effective in improving student knowledge of programming and improving their problem-solving skills by learning to develop flowcharts.

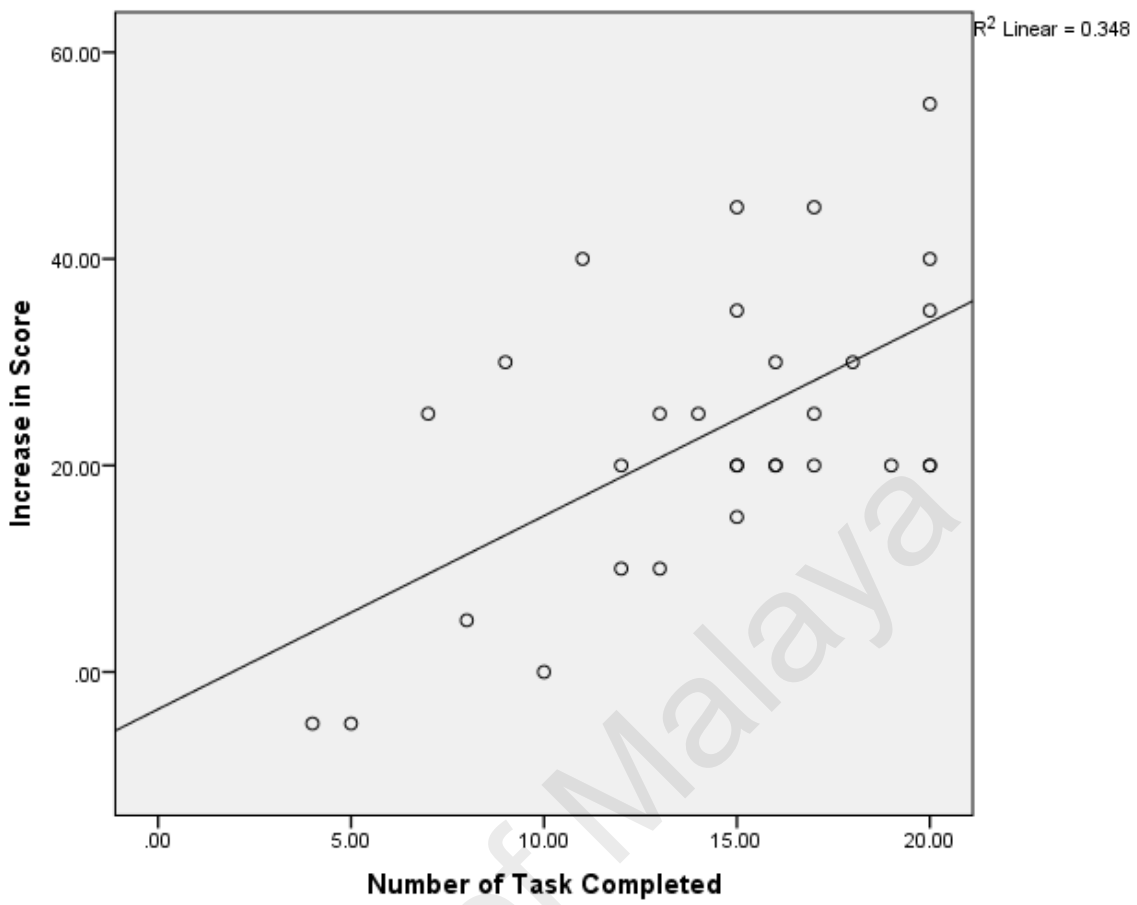

Figure 6.4: Scatter chart of number of task completed to the increase in score

6.3 The Student Group that Gets the Highest Benefit from FITS

After finding that FITS is effective for learning basic programing concepts and improving problem-solving skills, any student groups that get the highest benefit from FITS should be identified. The students who use FITS can be separated into two groups: the group of students who have learnt programming before and the group of student who have not. There are 16 students (53.33%) in the first group, and 14 students (46.67%) in the second group (see Figure 6.5). An independent samples two-tailed t-test was performed to see if there is a relationship between the students' programming background and the results of their:

1. pre-test
2. post-test, and
3. their increase in score.

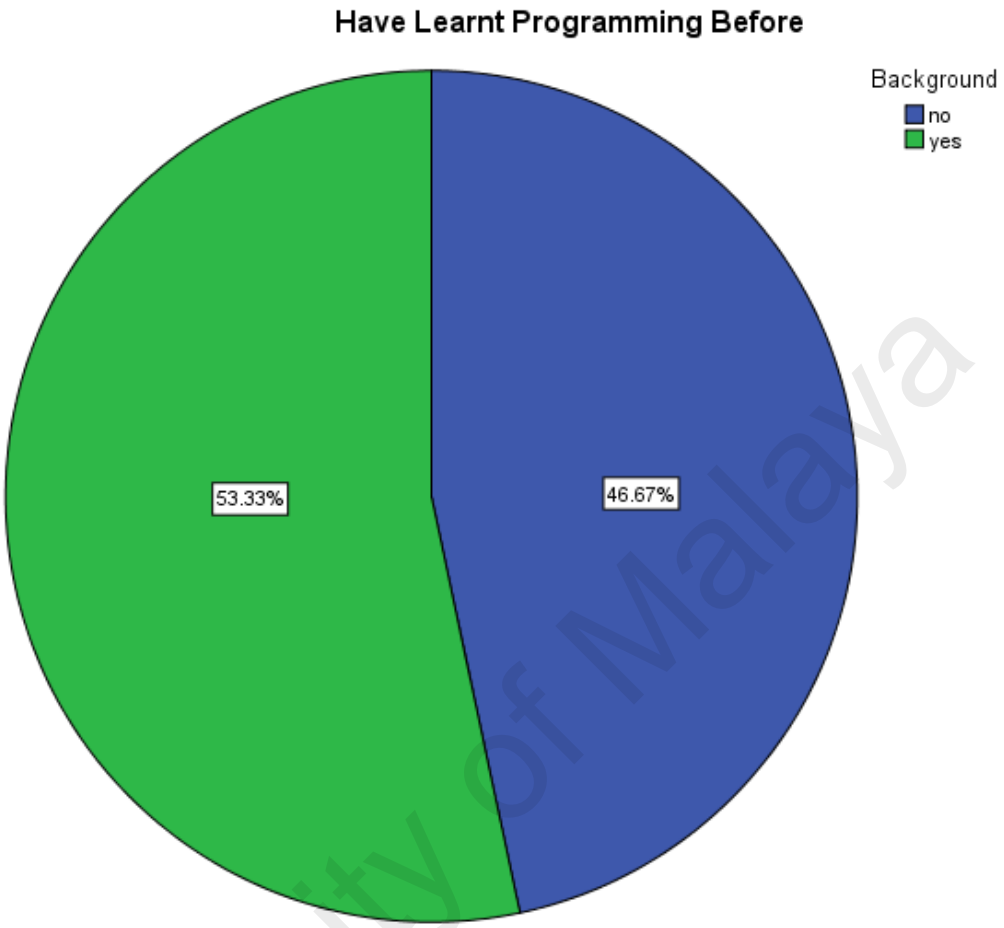

Figure 6.5: The number of participants grouped by the programming background

The null hypothesis for the first test is:

> *Hypothesis number 2: The average pre-test score of the students with a programming background is not better than the average pre-test score of the students without a programming background.*

The result of the independent samples two-tailed t-test of the students' programming background to the pre-test score is shown in Table 6.12. The p-value of 0.000 (less than 0.05) shows strong evidence against the null hypothesis. By looking at the average pre-test scores for both groups of students in Table 6.12, 47.81% for student with a programming background and 23.21% for students without a programming background,

it can be concluded that the average pretest score of the students' with programming background is significantly better than the average pre-test score of the students without a programming background. This result is not surprising. Students with a programming background should be able to answer some programming questions in the pre-test that are similar across multiple programming languages. Therefore their pre-test result should be higher than the students without a programming background.

Table 6.12: The students' programming backgrounds and their pre-test scores

Group Statistics

	Background	N	Mean	Std. Deviation	Std. Error Mean
Pretest_score	yes	16	47.8125	11.10086	2.77522
	no	14	23.2143	7.99210	2.13598

Independent Samples Test

		Levene's Test for Equality of Variances		t-test for Equality of Means						
		F	Sig.	t	df	Sig. (2-tailed)	Mean Difference	Std. Error Difference	95% Confidence Interval of the Difference	
									Lower	Upper
Pretest_score	Equal variances assumed	.269	.608	6.872	28	.000	24.59821	3.57954	17.26586	31.93057
	Equal variances not assumed			7.024	27.073	.000	24.59821	3.50203	17.41355	31.78288

The next independent samples two-tailed t-test evaluates the effect of the students' programming background on the result of the post-test. The null hypothesis for this test is:

> *Hypothesis number 3: The average post-test score of the students with a programming background is not better than the average post-test score of the students without a programming background.*

The two-tailed test in Table 6.13 shows the p-value of 0.729. Because the p-value is not less than 0.05 then there is no strong evidence to reject the null hypothesis.

Therefore the average post-test score of students with a programming background (i.e.: 60.31%) is not significantly better than the average post-test score for the students without a programming background (i.e.:58.57%). In other words, the students without a programming background can achieve post-test scores as good as the students with a programming background.

Table 6.13: The students' programming background and the post-test score

Group Statistics

	Background1	N	Mean	Std. Deviation	Std. Error Mean
Posttest	yes	16	60.3125	12.44572	3.11143
	no	14	58.5714	14.86200	3.97204

Independent Samples Test

		Levene's Test for Equality of Variances		t-test for Equality of Means						
		F	Sig.	t	df	Sig. (2-tailed)	Mean Difference	Std. Error Difference	95% Confidence Interval of the Difference	
									Lower	Upper
Posttest	Equal variances assumed	.327	.572	.349	28	.729	1.74107	4.98477	-8.46976	11.95190
	Equal variances not assumed			.345	25.521	.733	1.74107	5.04560	-8.63979	12.12194

The first and the second independent samples two-tailed t-tests above can be used to find out which group gets the highest benefit from FITS. However to make the conclusion clear, a third independent samples t-test was performed. This t-test evaluates the effect of the students' programming background on the students' increase in score in the post-test. The null hypothesis for this test is:

Hypothesis number 4: The increase in score of the students without a programming background is not higher than the increase in score of the students with a programming background.

The p-value of 0.000 in Table 6.14 shows the result of the two-tailed t-test. This value (that is less than 0.05) shows strong evidence to reject the null hypothesis. The average increase in score for both groups of students 32.50% for students without a programming background and 15% for students with a programming background shows that the increase in score of students without a programming background is significantly higher than the increase in score of students with a programming background. The results from Table 6.14 show that the students who get the highest benefit from FITS are the students who have not learnt programming before. The chart in Figure 6.6 gives a visual representation of this effect.

Table 6.14: The students' programming background and the increase in score

Group Statistics

	Background	N	Mean	Std. Deviation	Std. Error Mean
Increase_in_score	yes	16	15.0000	11.97219	2.99305
	no	14	32.5000	10.69687	2.85886

Independent Samples Test

		Levene's Test for Equality of Variances		t-test for Equality of Means						
		F	Sig.	t	df	Sig. (2-tailed)	Mean Difference	Std. Error Difference	95% Confidence Interval of the Difference	
									Lower	Upper
Increase_in_score	Equal variances assumed	.112	.740	-4.195	28	.000	-17.50000	4.17118	-26.04428	-8.95572
	Equal variances not assumed			-4.228	27.982	.000	-17.50000	4.13901	-25.97863	-9.02137

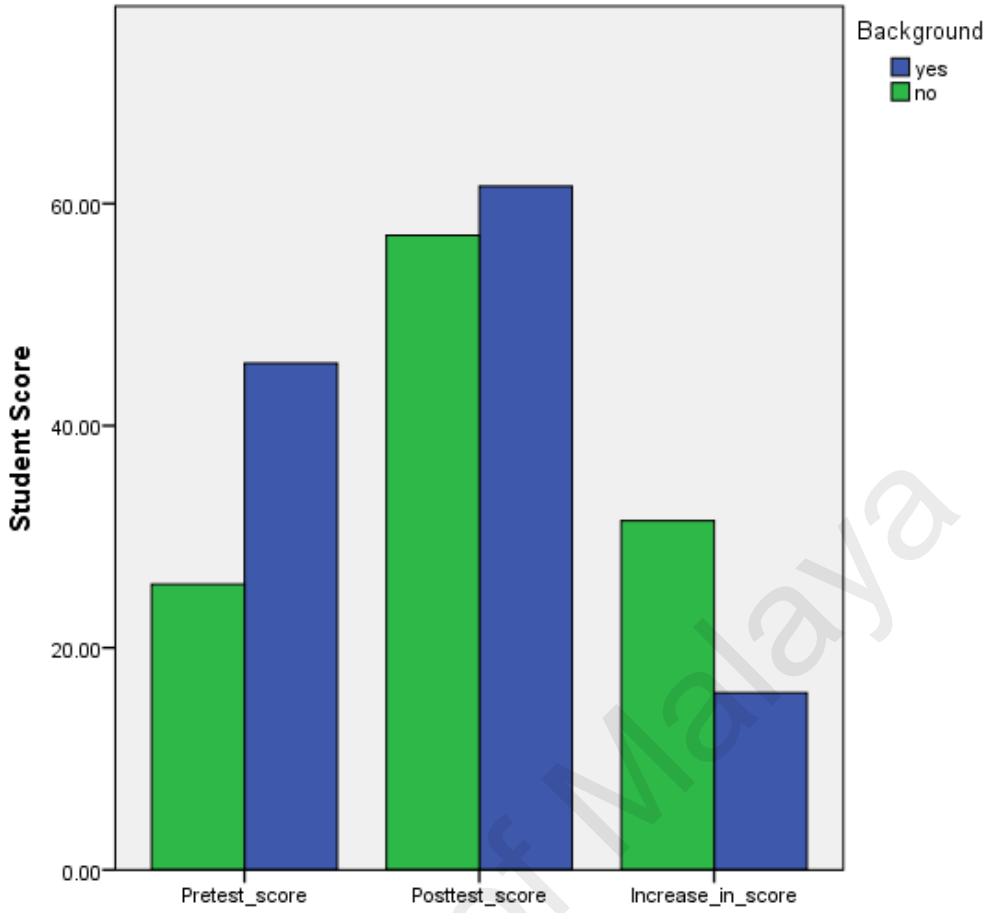

Figure 6.6: The effect of the student programming background to the student score

6.4 The Attractiveness of FITS when Learning Programming

As described in chapter 1, students should be encouraged to spend sufficient time on developing flowcharts and other solution designing activities as well as learning basic and imperative programming concepts. To facilitate this, FITS should be attractive to them.

Although there may appear to be a conflict between the effort to increase the time used by the students to develop flowcharts versus the nonexistence of a correlation between the length of time with the increase in student score (discussed in section 6.2.1), in fact there may no conflict between them. As explained in the last paragraph of section 6.2.1, the time used by each student to achieve the same performance may different from one student to another student. This evaluation did not state that the

length of time used by one student has no correlation with his/her performance. A student who spends two hours in FITS for example may get a better performance than if he/she spends only one hour in FITS. Therefore FITS should aim to make the students use FITS as long as possible. There are three factors that are intended to make FITS attractive to the students: its ease-of-use; its ability to help the students learn programming; and its learning approach appeal. In the following subsections, these three factors along with the students' actual opinions about them have been investigated more deeply and the students' responses have been used to the questionnaire (see Appendix A: FITS Questionnaire).

6.4.1 Is FITS Easy-to-use?

There are two questions that are used to measure if FITS is easy-to-use or not. In the first question, the students have been asked about the FITS user interface. A well designed user interface will enable them to use the application intuitively. The chart in Figure 6.7 shows that almost all students (90%) agree or strongly agree that the FITS user interface is well designed. The students' acceptance of the user interface is important because it is like a gate to the application. If the students dislike the user interface then it is possible that they would refuse to use the application in the first place.

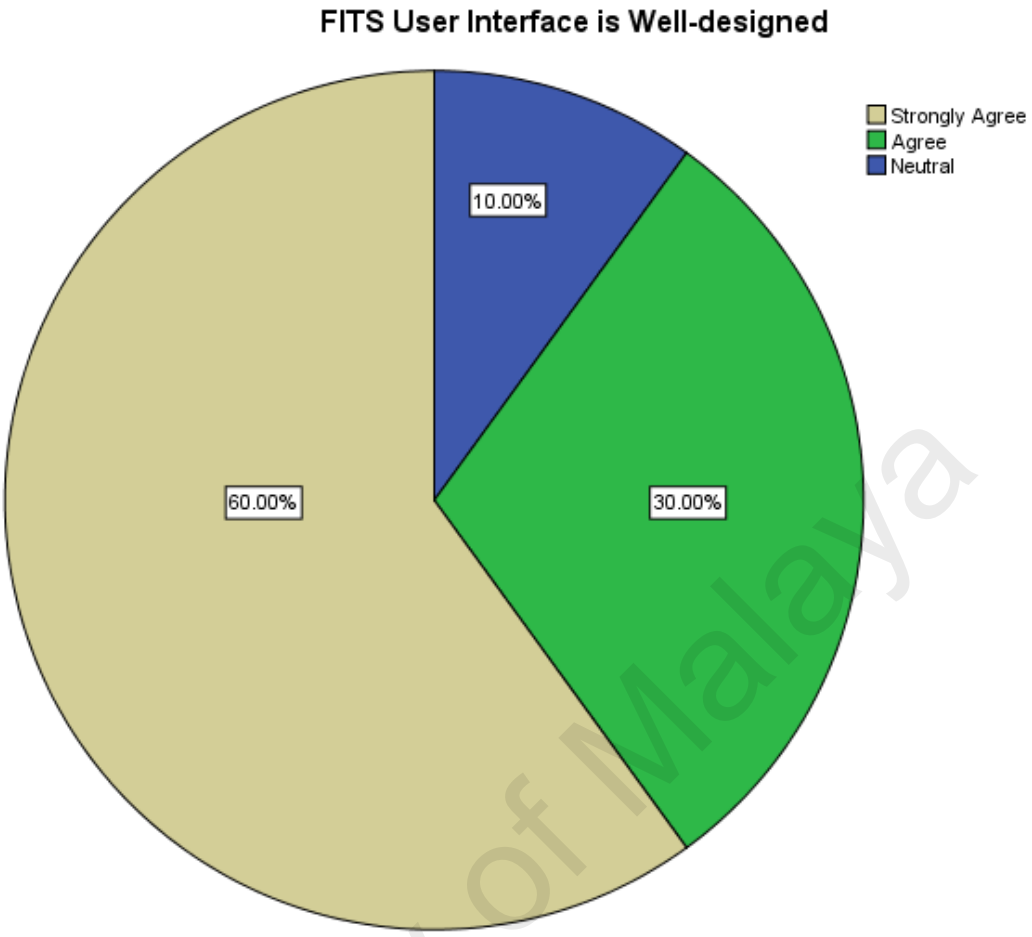

Figure 6.7: The user interface is well-designed

The second question is designed to see if overall the students think that it is easy to use FITS or not. The pie chart in Figure 6.8 shows that 86.67% of the students agree or strongly agree that it is easy to use FITS. This high percentage shows consistency of the students' answers with their previous answers. Therefore it can be concluded that the majority of the students (86.67%) consider that FITS is easy to use.

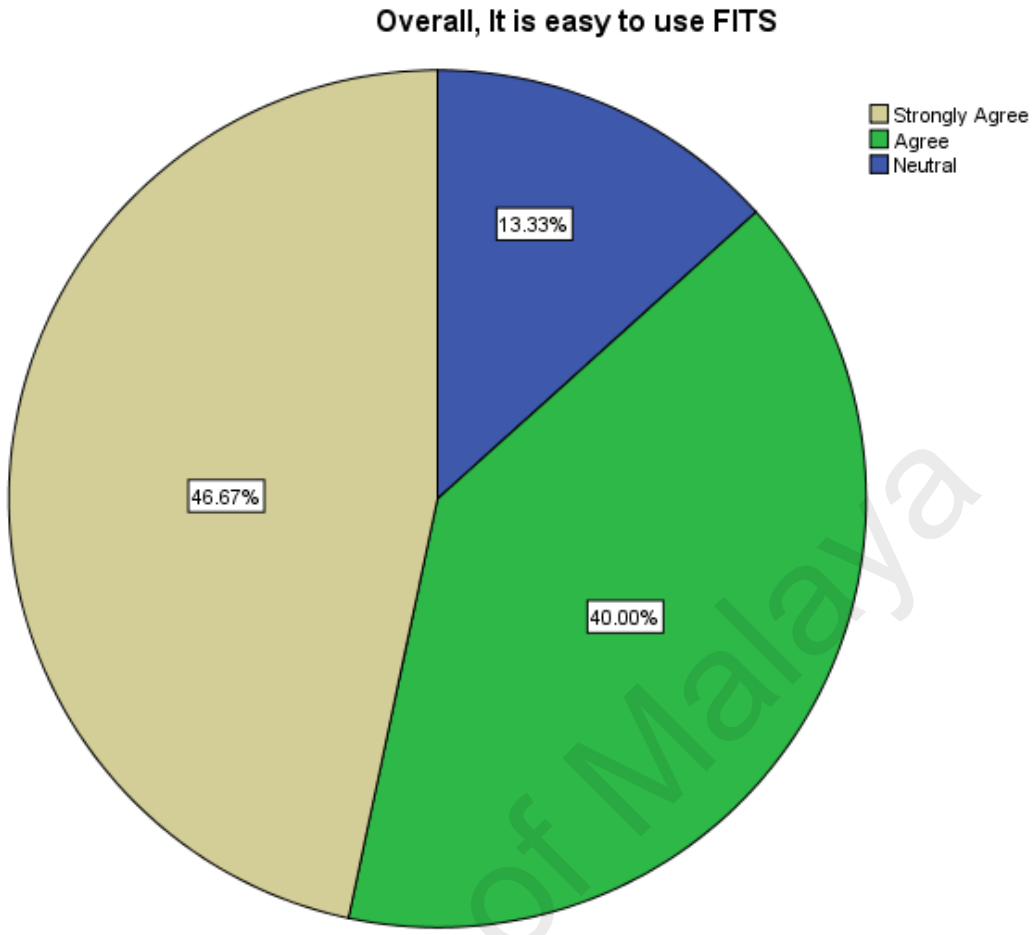

Figure 6.8: It is easy to use FITS

Some comments from the students are indicative of the students' satisfaction about this aspect, see Table 6.15.

Table 6.15: Students' comments about easy-to-use of FITS

Number	Students' comments
1	"Overall it is a very good idea and design for the students to learn programming."
2	"FITS taught me a lot of things right form the scratch."
3	"The traffic lights used to show the knowledge of each student about each concept is very impressive and I like it."
4	"It's a really good integrated pack with the flowchart development environment."

5	"The feature that i like most is the percentage bar at the bottom of page showing the specific and total knowledge of student."
6	"The conversion of text to flowchart automatically by FITS was a nice feature."

6.4.2 Does FITS help the student to learn programming?

In the next group of questions in the questionnaire, the students' opinions about FITS's functionality have been asked. Does FITS really help them learn to develop flowchart? What do they think about the help that is given by FITS, the number of tasks, and FITS's response time?

The first question in this group asks the student's opinion about the helpfulness of the FITS assistance as they develop their flowchart. This question is answered with mixed opinions. The chart in Figure 6.9 shows the students opinions. Although there are a small number of students who consider FITS assistance is not helpful, the majority of the students (73.34%) agree or strongly agree that FITS assistance is helpful. Afterward, a more detailed analysis of the group of students who considered the FITS assistance to be helpful had been done. The reason is to know if these students did really get benefit from FITS's assistance or not. The result in Figure 6.10 shows that 86% out of 73% of the students who considered FITS assistance to be helpful got an increase in their post-test score. This shows that the students who considered the assistance from FITS to be helpful did really get a benefit from it.

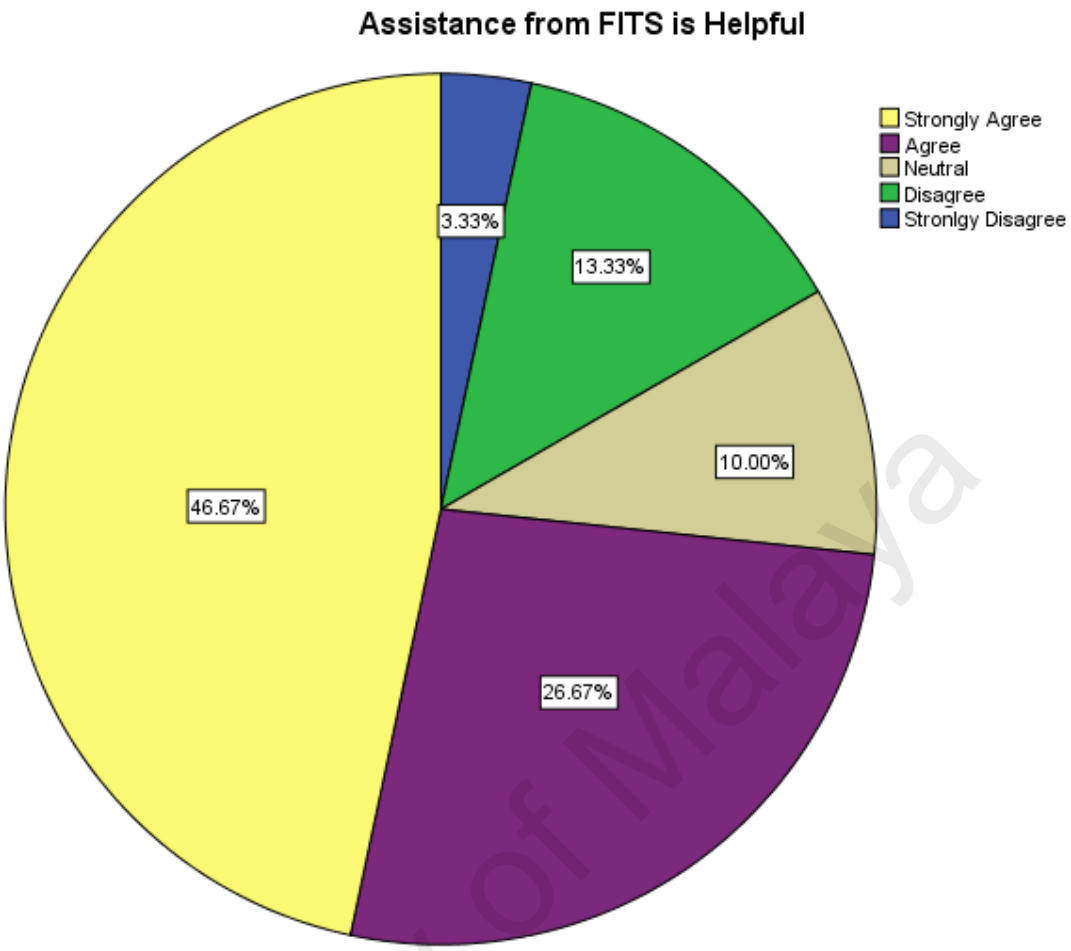

Figure 6.9: Assistance from FITS is helpful

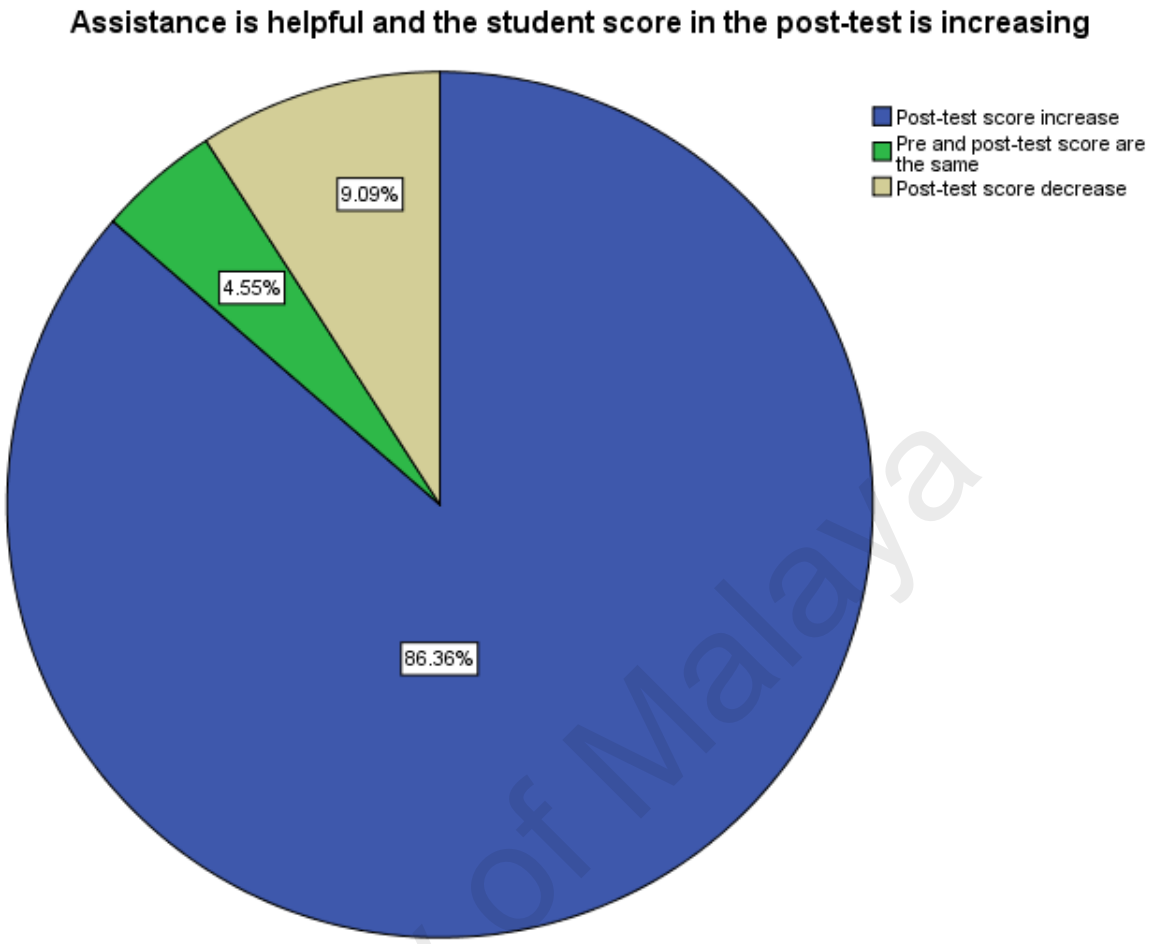

Figure 6.10: FITS is helpful and can increase the student score

For the next question on this issue, students have been asked about FITS's response time and the number of tasks that must be completed by students. For FITS's response time, 82.76% of the students agree or strongly agree that the response time in FITS is good (see Figure 6.11). None of the students disagree about this statement. A good response time is one of the requirements to make an application attractive.

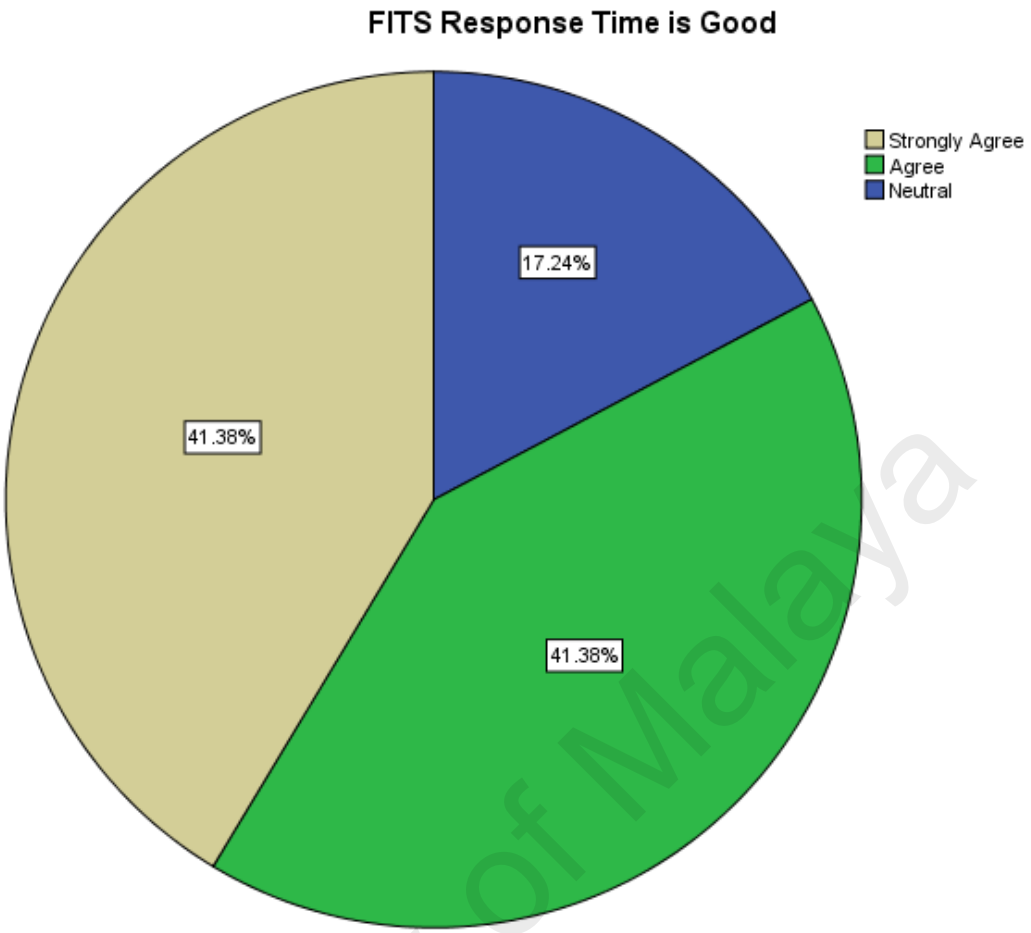

Figure 6.11: Response time in FITS

For the number of tasks that must be completed in FITS, 46.67% students consider that the number of tasks in FITS is already right 40% almost right, 3.33% too few, and 10% considers that they are a bit too many (see Figure 6.12). The answers for this question were numbered from one to five, with option 1 represents the opinion that the number of tasks are extremely too few, and option 5 represents the opinion that the number of tasks are extremely too many. Option 3 represents the opinion of the ideal number of tasks, and options 2 and 4 respectively represent the opinions that the number of tasks is a little bit too few or a little bit too many. 86.67% of students consider that the number of tasks is either already ideal (option 3) or slightly less ideal (options 2 and

4). Only 13.33% of students consider the number of tasks in FITS is extremely not ideal (options 1 and 5).

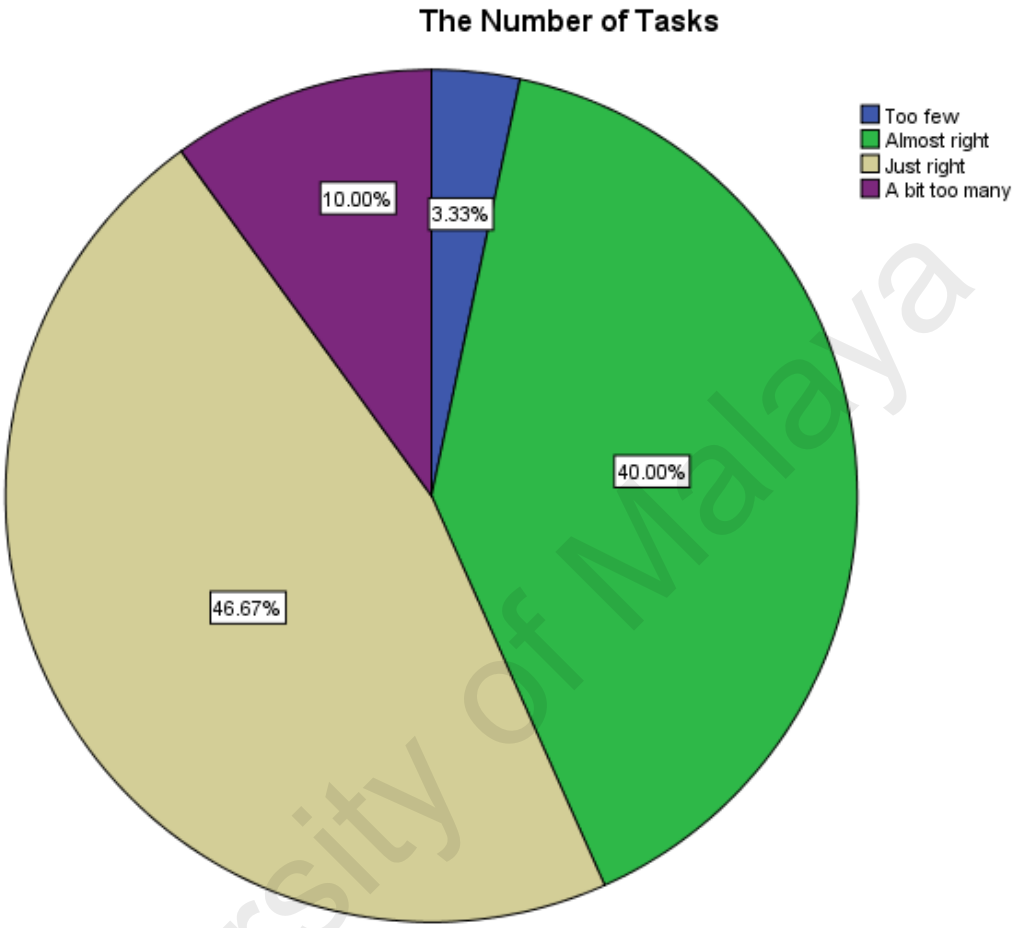

Figure 6.12: The number of tasks in FITS

Starting from this information, it has been tried to explore more deeply through the information from FITS's database to find out the ideal number of tasks that should be given to the students. Just as a reminder, there are 16 tasks that can be solved in the second release of FITS. However, the number of tasks that are actually completed by the students varies. This firstly is due to the fact that FITS assigns different number of tasks to the students depending on their initial and current knowledge (as described in section 5.2). Secondly, the students are allowed to do more tasks than the ones that are recommended for them. And, the students can do any exercise more than once. After

comparing the student's opinions about the number of tasks that should be given in FITS to the actual number of tasks that they did in FITS (see Table 6.16), it has been concluded that there is no correlation between these variables. The Pearson R value of 0.272 and the p-value of 0.145 show that there is no strong and significant correlation between those two variables.

Table 6.16: Correlation of opinions about the number of tasks to the actual number of tasks completed

Correlation: Opinion about Number of Task – Actual Number of Task Completed

		Task completed
number of tasks are ideal	Pearson Correlation	.272
	Sig. (2-tailed)	.145
	N	30

The final question that is related to the functionality of FITS is to ask the students if FITS is really helping them to learn programming. From the chart in Figure 6.13, 93.34% of the students agree or strongly agree that FITS helps them learn programming. This number shows that almost all of the students agree that FITS is really helpful. Thus, these students' opinions strengthen the claim in section 6.2 that FITS is effective at helping students learn to program.

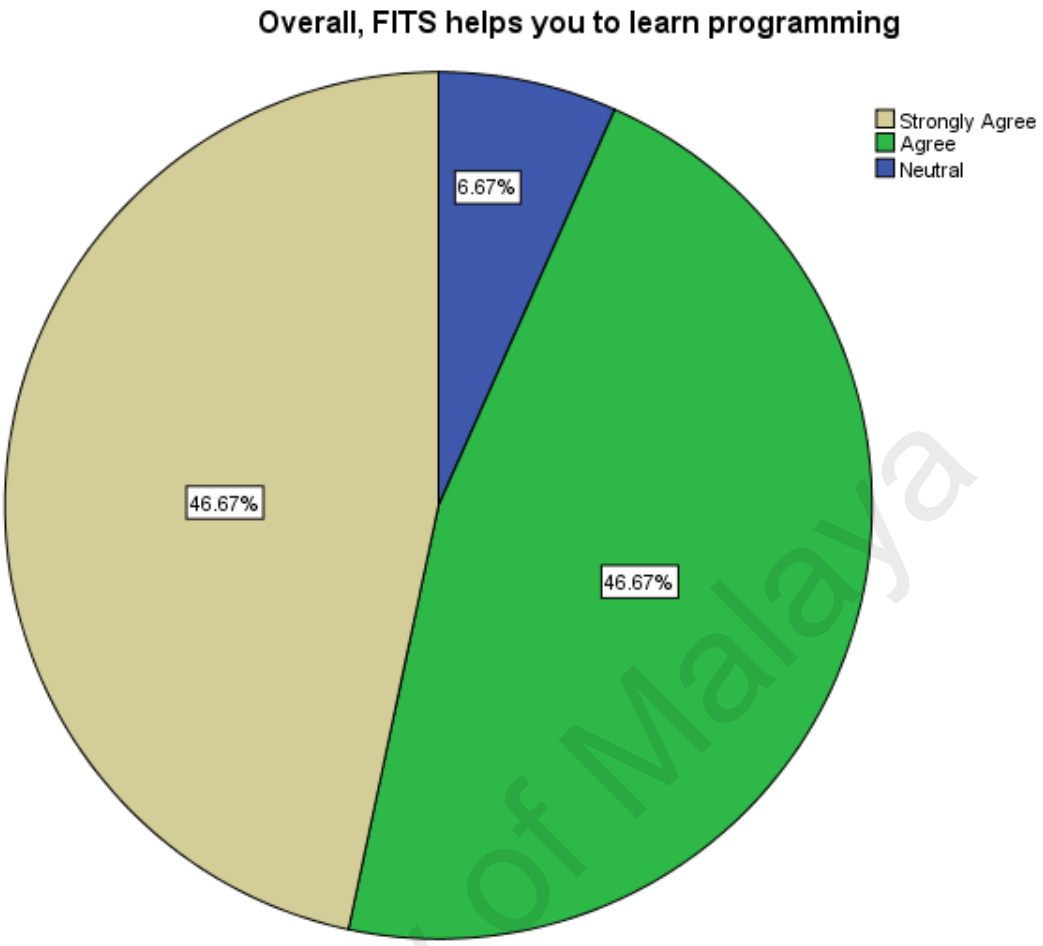

Figure 6.13: Overall, FITS helps the student learn to program

Some comments from the students indicate the students' satisfaction on this aspect, see Table 6.17.

Table 6.17: Students' comments about help provided by FITS

Number	Students' comments
1	"FITS is pretty good I like the way it is, I can't think of any suggestions that could make it better."
2	"No [I don't have any suggestion], this system is really helpful especially for programming beginners."
3	"Overwhelmed with how impressive FITS is, the rate of learning is remarkable."
4	"Overall I thought the FITS program was a good idea and made completing the tasks an easier experience, although it did have a few errors and quirks that could do with fixing."

5	“In some case, the suggestion from FITS is not clearly. However, overall the FITS is great. It is a helpful tool to learn a language.”
6	“It provides you with a help system to see where and how to start designing the solution when you get stuck in the beginning and do not know what to do.”

6.4.3 Is the learning approach in FITS attractive?

The learning approach that is used in FITS is the engagement of students in flowchart development with the aim of improving their problem-solving skills as well as teaching them basic programming concept through interactive and personalized environment. The automatic text-to-flowchart conversion is used in this approach, as described earlier in section 4.3. In this approach, the student could enter their programming problem statement and FITS automatically generates its relevant sub-flowchart and engage the students to complete it. To get feedback on this approach, the students have been checked for their opinions on whether the use of an automatic text-to-flowchart conversion approach to develop flowchart is a good idea or not. The chart in Figure 6.14 shows that 83.33% student agreed or strongly agreed with the idea. None of the students disagreed with this idea. Therefore, it has been concluded that the majority of students like the idea and none dislike it.

In FITS, the students pursue the goal of learning all unknown concepts that are shown in green and red lights in the navigational menu. When students were asked about this idea, 90% agreed or strongly agreed with this approach, while none disagreed (see Figure 6.15). Because pursuing a goal is part of the method used in FITS, the students’ positive responses in this case indicate they liked the method.

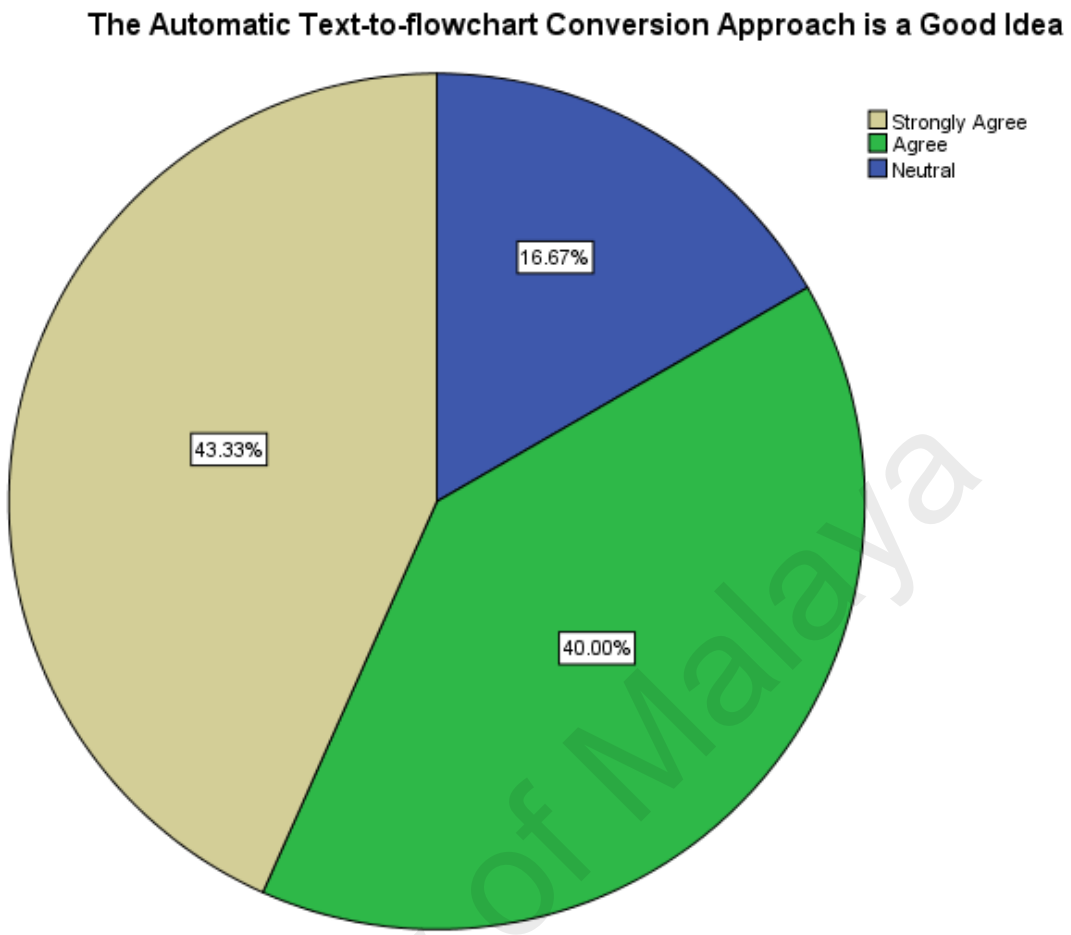

Figure 6.14: The flowchart-based approach is a good idea

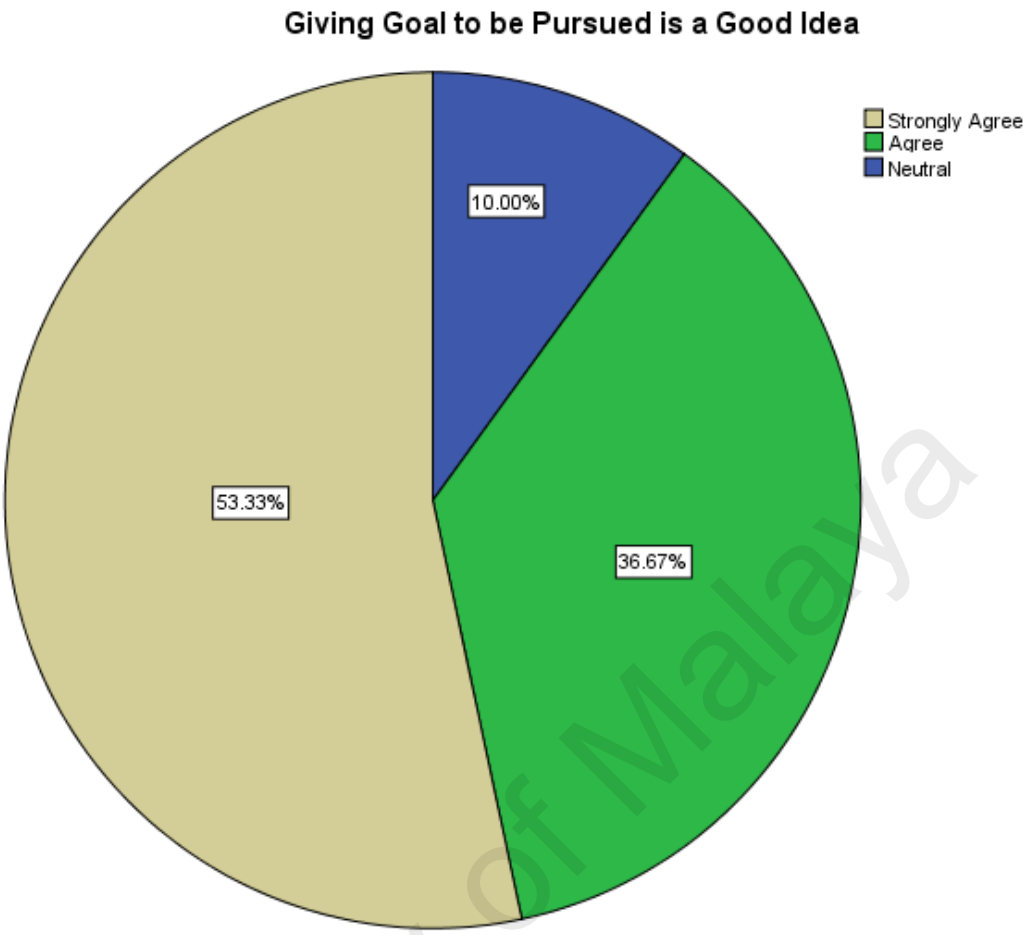

Figure 6.15: Giving goal to be pursued is a good idea

The next question is about the flowcharts development. When the students' opinions were asked on whether learning programming by engaging you in the process of flowchart development and teaching basic concepts of programming simultaneously is a good idea or not, 100% agreement were received (see Figure 6.16). Therefore, it seems that all the students consider that one possible way to understand programming is by developing flowchart. Providing an attractive tool to learn to program, such as FITS, can be a good approach to improve problem-solving skills of students.

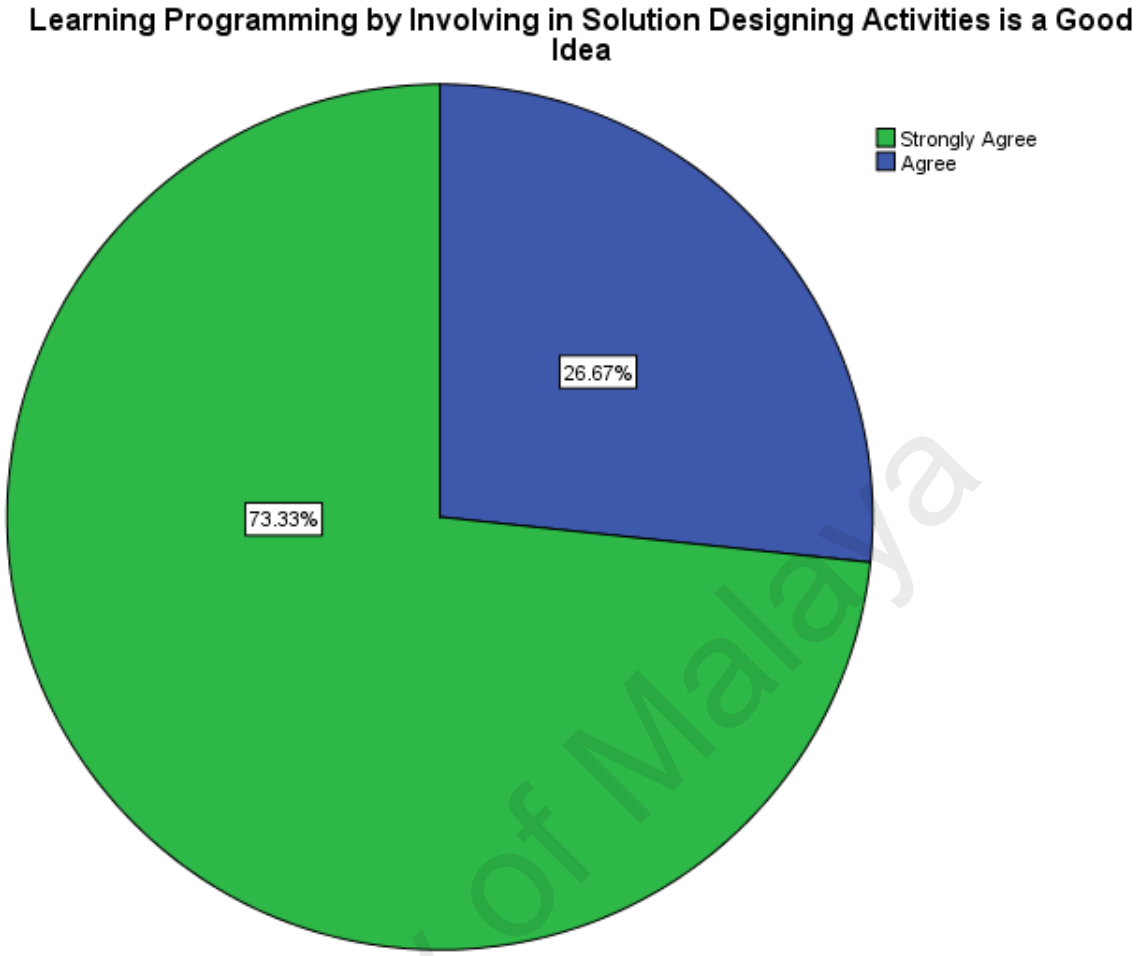

Figure 6.16: Learning programming by involving in flowchart development is a good idea

In the next question, the students were asked if providing information at the time when they encounter a problem is a good idea or not. As shown in Figure 6.17, 86.67% of the students agreed with this idea and none disagreed. This result shows that the students really expect to have an assistant that is ready to help them, whenever they have a problem in developing flowchart.

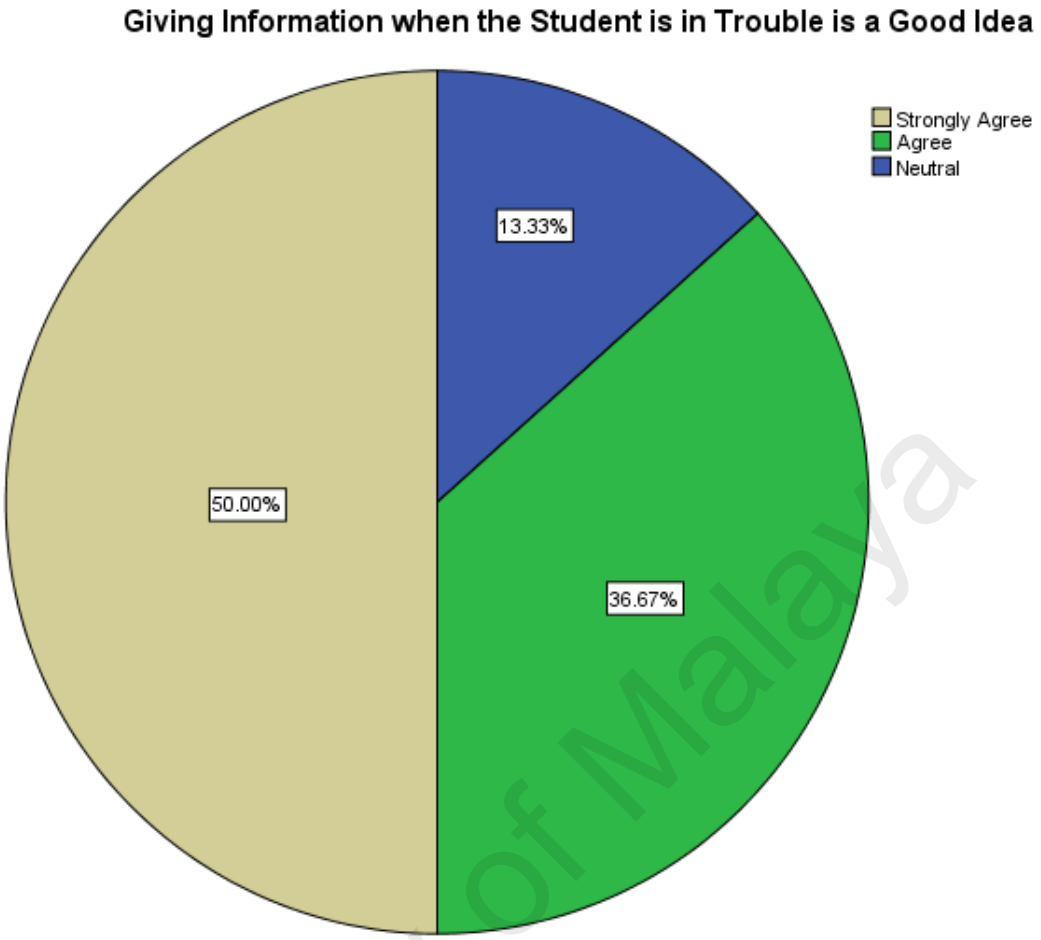

Figure 6.17: Give information when the student is in trouble is a good idea

The result of the students' opinions in Figure 6.14 to Figure 6.17 show that majority of students agree that the learning approach provided by FITS is good. Therefore, this agreement has been considered as one additional ingredient that shows the attractiveness of FITS. In the last question, the students were asked whether the tic-tac-toe game boosted their motivation and interaction during the learning process. This is related to improving student motivation and enthusiasm when timely guidance is lacking, as described in section 5.2.

This game incorporated in FITS gives opportunity of raising their motivation when timely guidance are lacking and students may get bored. From Figure 6.18, 93.33% students agree or strongly agree with this idea which indicates to the usefulness of

applying the game through FITS for encouraging students to get involved in programming activities more and more.

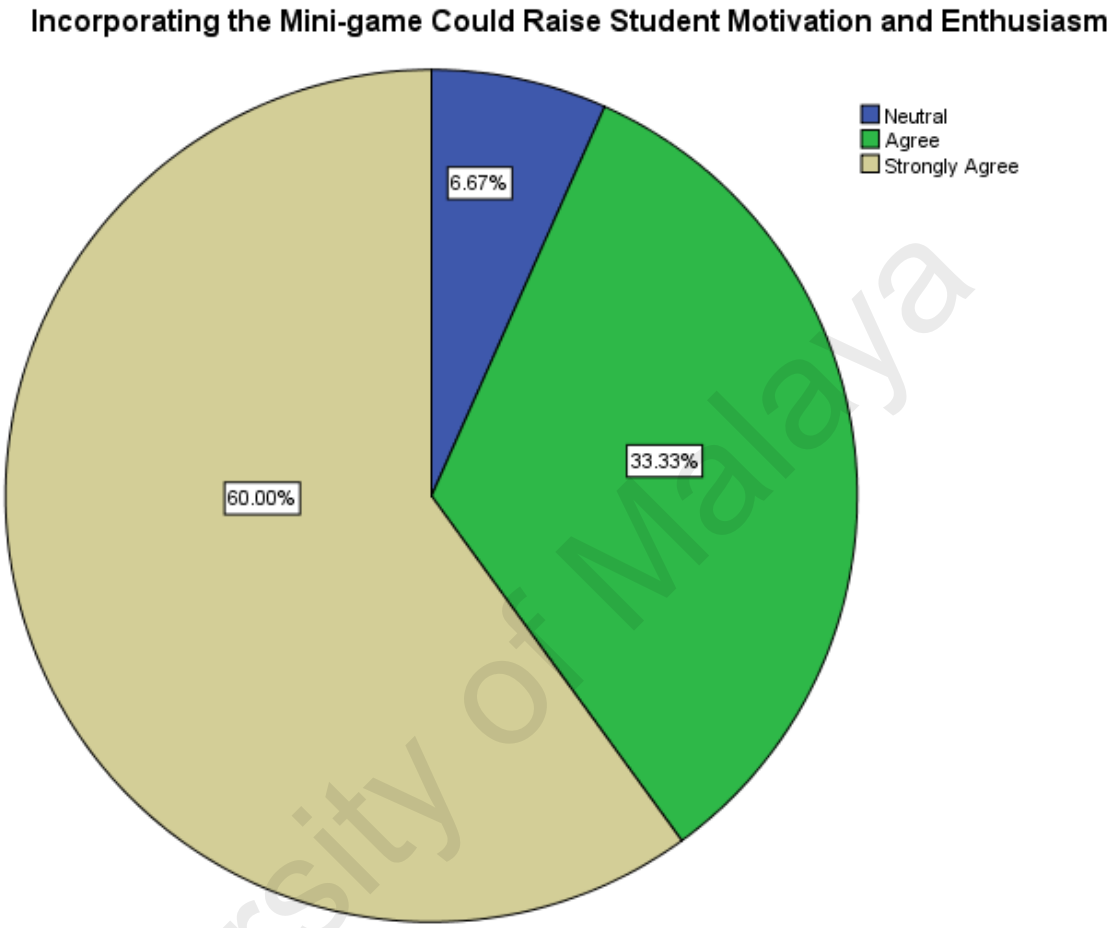

Figure 6.18: Incorporating the mini-game could raise students' motivation

Comments from the students indicate the students' satisfaction on this aspect, see Table 6.18.

Table 6.18: Students' comments about the mini-game incorporated in FITS

Number	Students' comments
1	"The feature that I like most is the opportunity of playing game while you are learning programming concepts."
2	"The feedbacks provided within the game were nicely and appropriately designed."
3	"Some of the questions within the game were difficult."

4	"It was the first learning environment I used having game and I really enjoyed it."
5	"FITS game playing was very encouraging."

6.5 Does FITS focus on problem-solving and assist the improvement of problem-solving skill?

There are three questions used to measure if using flowcharts are helpful in supporting the learning of programming. In these three questions, as the flowchart development could engage the students in meaningful problem-solving activity, the students were asked about the helpfulness of flowcharts in visualizing solutions and algorithms in their minds, whether the flowcharts are good problem-solving aids, and the usefulness of flowchart when designing problem solutions. The charts in Figure 6.19 shows that around 90% of students agree or strongly agree about the helpfulness of FITS's flowchart development in visualizing solutions and algorithms in their minds. Also, the charts 6.20 and 6.21 show that 83.33% and 86.67% of students agree or strongly agree that flowchart development of FITS is a good problem-solving aids and are useful when designing problem solutions. The responses to all three questions show almost all students believe the flowcharts: helped them visualize solutions in their mind, were good problem-solving aides and were useful in designing problem solutions. Clearly, these results show the majority of students believe FITS's flowcharts are having a positive impact on problem-solving ability and associated skills such as the development of mental models.

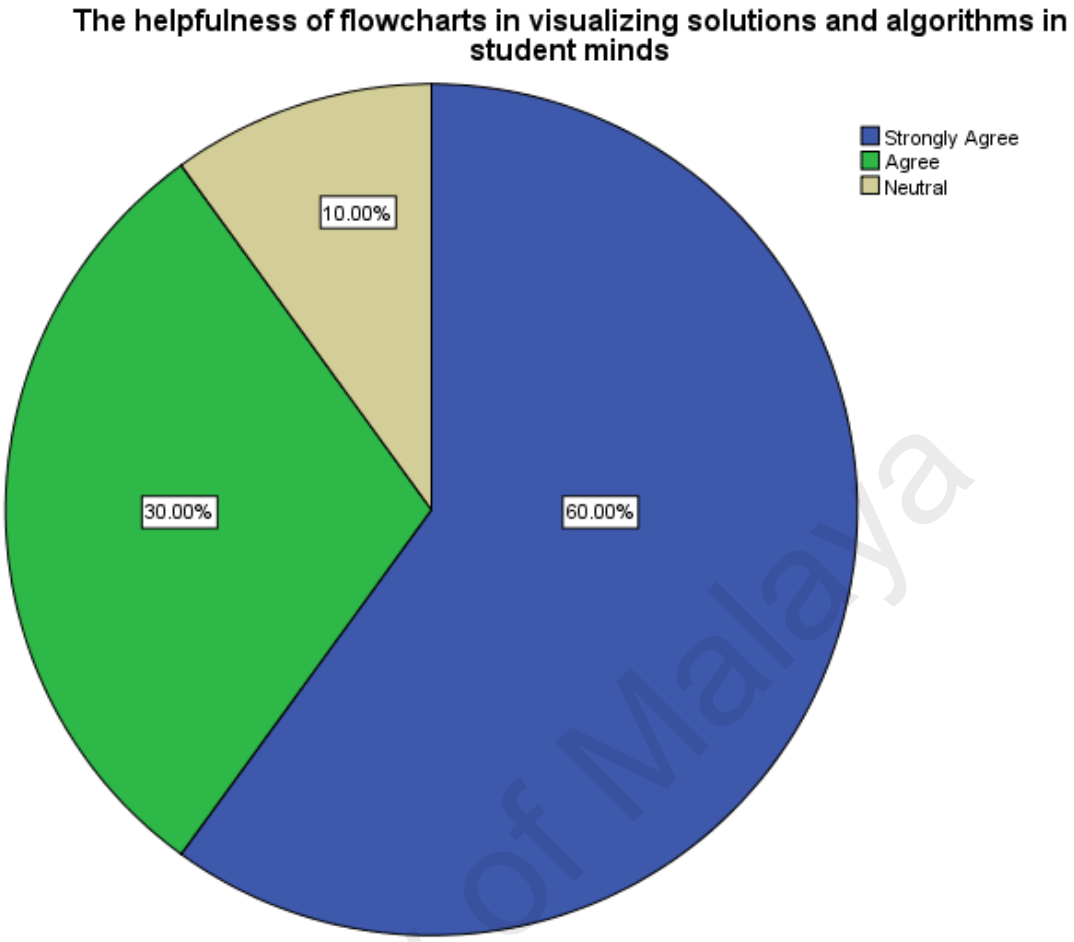

Figure 6.19: The helpfulness of flowchart in visualizing solutions in students mind

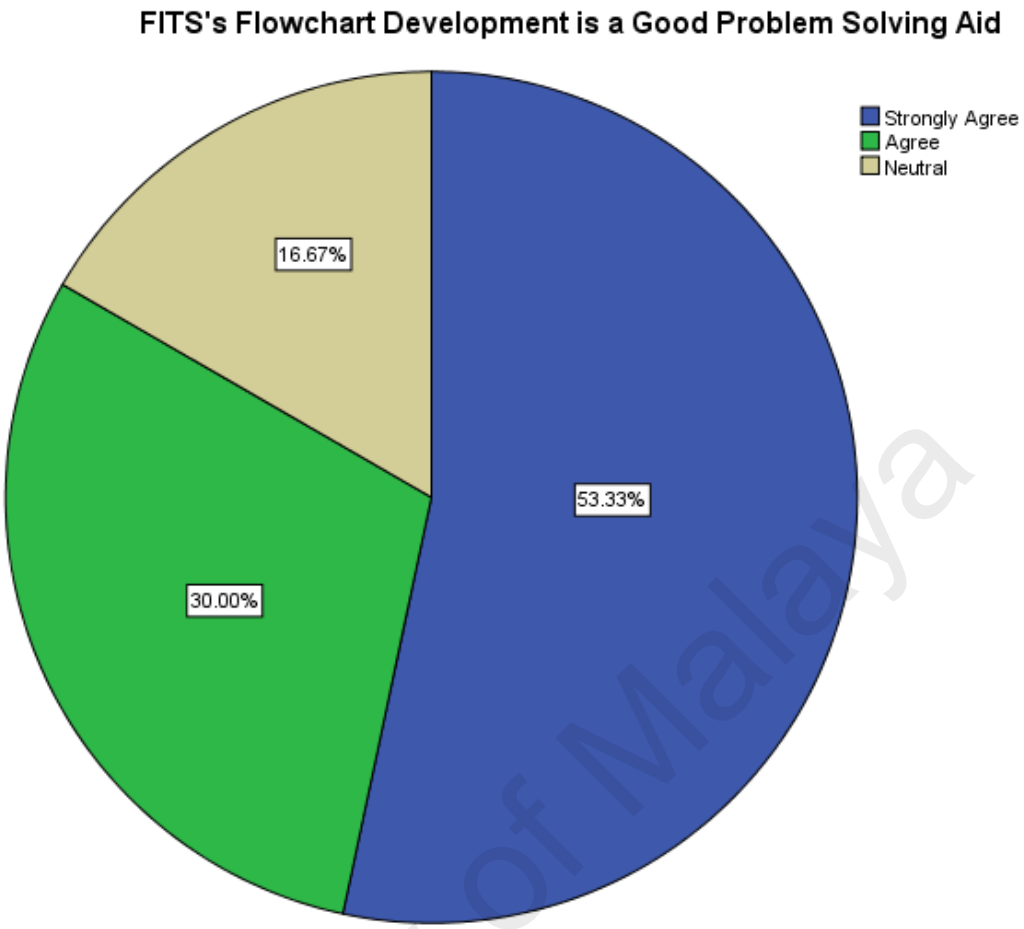

Figure 6.20: The flowchart development of FITS is a good problem-solving aid

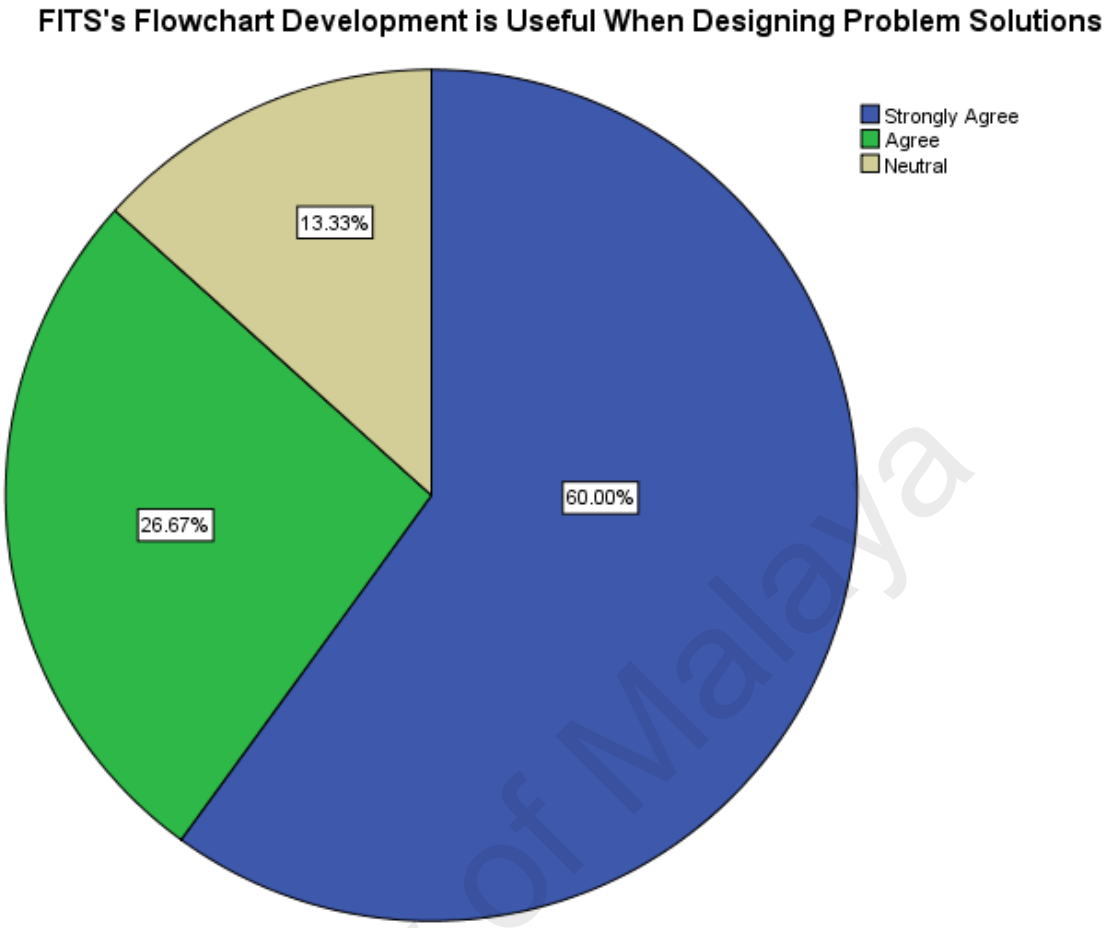

Figure 6.21: The usefulness of flowchart development when designing problem solutions

6.6 The Suggestion of using FITS for other Units

This section explored student satisfaction with FITS in greater detail. The hypothesis is that if students are very satisfied, they would not hesitate to recommend that FITS be used again in the same unit, or even for extended use in other programming units. For this, the students were asked if they would recommend FITS for use in other programming units.

It is very encouraging that none of the students disagreed with the idea of using FITS in the Programming I unit and in other programming units. The results in Figure 6.22 shows that 90% of the students agree or strongly agree that FITS should be used in Programming I course. Similarly, the results in Figure 6.23 shows that 86.66% of the

students agree or strongly agree that FITS should also be used for other programming units. These students' agreement would not be this high if the student were not satisfied with FITS. Based on these findings, it can be concluded that the students were really satisfied with FITS.

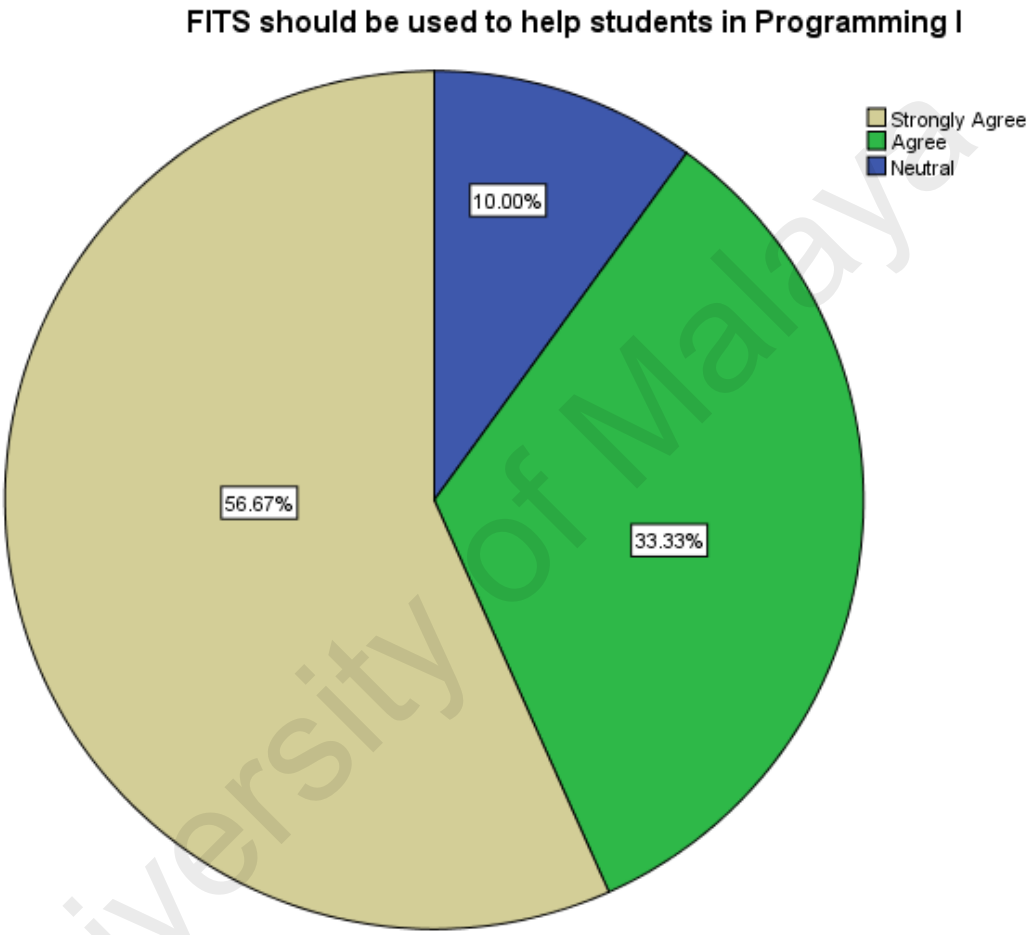

Figure 6.22: FITS should be used to help students in Programming I

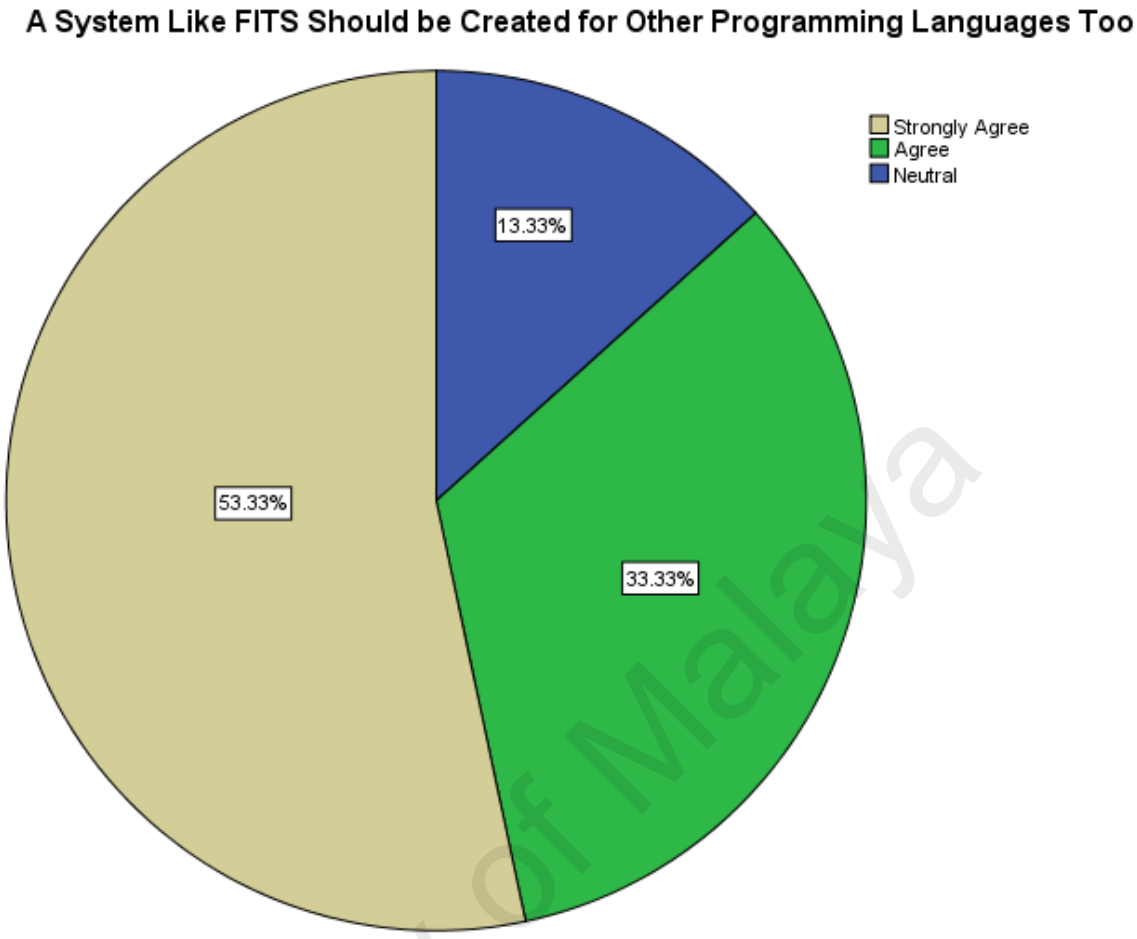

Figure 6.23: FITS should be used for other programming units

The following comments from the students, Table 6.18, indicate the students' overall satisfaction with FITS:

Table 6.19: Students' comments about FITS suggestion to other programming units

Number	Students' comments
1	"Great program; it should be mass-produced."
2	"This is one of the most entertaining assignments I did."
3	"FITS is really a helpful system. I would love to use it again and I recommend others to do so."
4	"I really like it and help me a lot and I recommend it to other students."
5	"Apart from a few little hiccups though, I'm happy with the experience."

6.7 Summary

This chapter described the process to evaluate FITS. The effectiveness of FITS was evaluated along with its attractiveness. Some evaluations were performed to justify the effectiveness and the attractiveness of FITS. Several instruments were used to in the evaluation process. Different data analysis methods were used to find out which one of the null hypothesizes are supported and which one is not, see Table 6.20.

Table 6.20: Different hypothesis used to evaluate different aspects

Hypothesis	The null hypothesis	Result
1	The student scores in post-test are not higher than the student scores in the pre-test.	The p-value of two-tailed paired t-test shows strong evidence against the null hypothesis. Therefore, it can be concluded that the scores in the post-test are significantly higher from the scores in the pre-test.
2	The average pre-test score of the students with a programming background is not better than the average pre-test score of the students without a programming background.	The p-value of the independent samples two-tailed t-test shows strong evidence against the null hypothesis. Therefore, it can be concluded that the average pretest score of the students' with programming background is significantly better than the average pre-test score of the students without a programming background.
3	The average post-test score of the students with a programming background is not better than the average post-test score of the students without a programming background.	According to the p-value of the independent samples two-tailed t-test, there is no strong evidence to reject the null hypothesis. Therefore the average post-test score of students with a programming background (i.e.: 60.31%) is not significantly better than the

		average post-test score for the students without a programming background (i.e.:58.57%). In other words, the students without a programming background can achieve post-test scores as good as the students with a programming background.
4	The increase in score of the students without a programming background is not higher than the increase in score of the students with a programming background.	The p-value of two-tailed paired t-test shows strong evidence to reject the null hypothesis. The average increase in score for both groups of students shows that the increase in score of students without a programming background is significantly higher than the increase in score of students with a programming background. Therefore, the students who get the highest benefit from FITS are the students who have not learnt programming before.

CHAPTER 7: CONCLUSIONS

This chapter starts with a summary of this research. The specific contributions of this thesis are described in section 7.2, addressing the research goal and research questions stated in Chapter 1. The final section, 7.3, offers ideas for enhancing FITS in the future.

7.1 Research Summary

This research was motivated by the known fact that learning programming is difficult for novices and by the need to help students learn programming through solution designing activities. One of the reasons why many students fail in programming courses is that they do not spend adequate time on flowchart development and rush to write the code once given a task. One way of encouraging students to improve their problem-solving skills is to practice designing a solution for a given programming problem through an adaptive flowchart development environment, which could provide assistance at any time when students encounter a programming problem. However, this solution is impractical if human tutors are to handle this. The solution to this problem – the focus of this research – is to create a Flowchart-based Intelligent Tutoring System model (FITS) that can assist students with programming tasks when necessary.

Such an ITS should not only be designed to help students but also to yield some improvements in students' problem-solving skills while engaging them in flowchart development. It should also be sufficiently appealing for students to use as a means to learn early programming stages. While the lack of problem-solving skills has been frequently reported as the main reason behind students' failure in computer programming courses, no ITS in the programming domain has addressed this issue by involving students in solution designing activities before the coding stage. The ITS described in this thesis is aimed to develop students' problem-solving skills by incorporating a flowchart-based approach as the learning approach. This ITS is named

FITS, and it is intended to be used by beginner programmers. It covers the following topics in the C++ programming language: variables, assignment and control structures. As an ITS, FITS allows students to learn at their own pace and learn material that is consistent with their current knowledge. A flowchart-based multi-agent system benefiting from an automatic text-to-flowchart conversion approach is applied to support flowchart development in FITS.

7.2 Research Contribution

As described in section 1.3, the main goal of this research is to investigate if a flowchart-based ITS can help students to learn basic concepts of computer programming effectively and to bring improvement in their problem-solving ability. In order to achieve this goal, a flowchart-based ITS model (named FITS) is designed and developed to help students learn basic concepts of computer programming effectively and to bring improvement in their problem-solving ability. FITS was used and evaluated by university students in programming course names "Programming I" at University of Malaya.

The evaluation of FITS was performed by analyzing the results of the students' work in FITS, and through a questionnaire. Details of the evaluation are presented in chapter 6. The main result of the evaluation (section 6.2) is that the students' scores increased significantly by using FITS. A more detailed analysis (section 6.3) shows that the students with no programming background get more benefit from FITS than the students with a programming background. This is shown by a higher increase in score that are obtained by the students with no programming background. The evaluation also showed that the more tasks completed by the students in FITS, the higher the improvement that was obtained. From those results, it can be concluded that FITS does help the students learn effectively.

The evaluation of whether the flowchart-based learning approach can assist the improvement of problem-solving skills is discussed in section 6.5. Because all of the questions on the flowchart-based learning components were responded positively by students, it has been concluded that the flowchart-based learning approach does bring some improvements in the students' problem-solving skills. Based on these findings and from the very positive students' comments on the various issues described in chapter 6, It can be confidently concluded that incorporating a flowchart-based learning approach in ITS can bring improvements in the students' problem-solving skills and can help them learn early stages of programming effectively. Therefore the main goal of this research is achieved.

Related to this research goal, the four more-specific research questions stated in section 1.3 are addressed here:

1) What is the best method of knowledge representation that can be used to model the subject matter necessary to effectively teach basic C++ programming concepts while achieving the following?

a. Involving students in an adaptive flowchart development process (solution designing activities)

One of the major challenges in FITS is to provide students with an adaptive and personalized environment for flowchart development with the aim of improving their problem-solving skills. This task is challenging because there has yet to be any personalized flowchart development environment in academic worlds aimed to bring improvements in student's problem-solving skills. As described in section 5.2.4, benefiting from Bayesian Network, FITS could provide students with an adaptive and personalized flowchart development environment so as to engage them in solution designing activities with the aim of bringing improvements in their problem-solving

skills before coding stage. As explained in section 4.4, FITS offers three different flowchart development options for the given programming problem based on the students' current knowledge and actively motivate them to complete the generated sub-flowcharts (some parts of a full flowchart) by FITS in order to improve their problem-solving skills.

b. Providing feedback based on the specific errors made by the student

One of the important challenges in FITS is to provide appropriate feedback and help that fits with each error made by the students. As described in section 4.4 and 5.2, FITS offers different types of feedbacks based on the error made. In addition, two agents are improvised in the architecture of the flowchart-based multi-agent system which incorporated in FITS in order to detect the error each student makes during the flowchart development and pass them to a web crawler to search the relevant learning materials and automatically add them to its database. Thus, the next user makes the same error, FITS could offer updated feedbacks for that specific error, see section 4.4.1.

2) How to provide adaptive guidance that fit with the students' current knowledge?

Providing adaptive guidance that fit with the students' current knowledge is important to make the students feel comfortable with the given tasks. In FITS, this is supported by providing three types of adaptive guidance to the students (i.e. navigational support, pre-requisite recommendations, and flowchart development) based on a student model, benefiting from Bayesian Network, which represents the students' current knowledge. Section 5.2 describes in detail the student model that is used in FITS.

3) How to attract the students to the learning materials and raise their motivation when they are demotivated?

As described in section 6.4, there are three supporting factors that can attract the students to develop flowcharts. These three factors are the existence of help when the students get trouble when developing a flowchart, the ease-of-use of the ITS, and the attractiveness of the learning approach provided by the ITS. The evaluation in section 6.4 showed that FITS does attract the students to develop flowcharts. Specifically, in the section 6.4.3, the result shows the game incorporated in FITS increases students' engagement in the leaning process and gives opportunity of raising their motivation when timely guidance are lacking.

7.3 Future Work

There are several enhancements that could be made to FITS, given additional time to develop and evaluate them. Some of the possible enhancements are:

1. Increasing the scope of the covered topics in FITS. In the current version, FITS covers topics such as variables, assignments, and control structures. For a more sophisticated system, it would be reasonable to include all basic and imperative programming concepts which should be taught before coding stage.
2. Enhancing the flowchart-based learning part. Several enhancements that could be made are:
    - Providing a better means of communication between the students and the ITS. In its current version, only FITS can actively send information to the students and not the other way around (apart from student flowcharts). Allowing the students to also send their questions or requests to the ITS will make the system look more alive. One possible way of realizing this idea without too much effort is by providing options that can be selected by students to send fixed questions or requests to the ITS (Wang, Wang, & Huang, 2008). Another possible way, requiring

more effort to realize it, is to provide a command line. Through this command line, the students could write answers to questions from the ITS, or make requests to the ITS (Graesser, Chipman, Haynes, & Olney, 2005). The flexibility of the language allowed to be used by the students would determine the effort required to realize it (Latham, Crockett, McLean, & Edmonds, 2012; Tegos, Demetriadis, & Tsiatsos, 2014).

- Providing pictures or animation to support the ITS. Providing the right pictures/animation at the right time could make the students become more absorbed with the system (Rickel & Johnson, 2000). To avoid distraction, a good balance must be found between the length of time that is used to show the pictures/animations and that spent by the students. The students should also have an option to skip these pictures/animations.

3. Extending FITS to provide students with the algorithm, flowchart, and its relevant code together while showing the relationship among different parts. Moreover, analyzing the written code by student while showing its relevant flowchart could be an effective feature to be added.
4. Developing the standalone application of FITS as well as its mobile application in order to be able to be used by a broader range of users. A new standalone application could be created as a framework for a new FITS. This framework could then be filled with most modules from the current FITS. The system should also be able to detect the existence of an internet connection automatically and work offline when the internet connection is not available. (Synchronization with the backend data is required when an internet connection becomes available.) Therefore the students would still be able to use FITS even though there was not always an internet connection.

In conclusion, as aforementioned, a system that allows the novice to construct a program visually via a flowchart-based representation will provide the novice with an effective mental model of the constructs and how they can be composed whilst reducing syntactic overheads. This will place a greater emphasis on the underlying abstractions of programming, problem-solving and program composition i.e. how the pieces fit together to form a solution to a problem or specification. The dynamic execution of a flowchart will allow the novice to evaluate the appropriateness of their solutions, gain a deeper understanding of the programs they have composed and foster a mental model of program execution.

REFERENCES

Aleven, & Ashley. (1997). *Teaching Case-Based Argumentation Through a Model and Examples: Empirical Evaluation of an Intelligent Learning Environment.* Paper presented at the Artificial Intelligence in Education, Proceedings of AI-ED 97 World Conference Amsterdam, The Netherlands.

Anderson, Corbett, Koedinger, & Pelletier. (1995). Cognitive Tutors: Lessons Learned. *The Journal ofthe Learning Sciences, 4*(2), 167-207.

Anderson, & Reiser, B. J. (1985). The LISP tutor: it approaches the effectiveness of a human tutor. *BYTE, 10*(4), 159-175.

Areias, C., & Mendes. (2007). *A tool to help students to develop programming skills*. Paper presented at the Proceedings of the 2007 international conference on Computer systems and technologies, Bulgaria.

Baker, R., Corbett, A., & Koedinger, K. (2004). Detecting Student Misuse of Intelligent Tutoring Systems. In J. Lester, R. Vicari, & F. Paraguaçu (Eds.), *Intelligent Tutoring Systems* (Vol. 3220, pp. 531-540): Springer Berlin Heidelberg.

Bassat Levy, R., Ben-Ari, M., & Pekka, U. (2001). *An Extended Experiment With Jelliot 2000.* Paper presented at the in Proceedings of the First International Program Visualization Workshop, Porvoo - Finland, University of Joensuu.

Beck, J. E., Chang, K.-M., Mostow, J., & Corbett, A. (2008). *Does Help Help? Introducing the Bayesian Evaluation and Assessment Methodology*. Paper presented at the Proceedings of the 9th international conference on Intelligent Tutoring Systems, Montreal, Canada.

Bell, C., & McNamara, D. S. (2007). *Integrating iSTART into a high school curriculum.* Paper presented at the Proceedings of the 29th Annual Meeting of the Cognitive Science Society, Austin, TX.

Borges, S. d. S., Durelli, V. H. S., Reis, H. M., & Isotani, S. (2014). *A systematic mapping on gamification applied to education*. Paper presented at the Proceedings of the 29th Annual ACM Symposium on Applied Computing, Gyeongju, Republic of Korea.

Bouras, A. S., & Ainarozidou, L. V. (2015). *C\# and Algorithmic Thinking for the Complete Beginner: Learn to Think Like a Programmer*: CreateSpace Independent Publishing Platform.

Buchanan, B. G., & Shortliffe, E. H. (1984). *Rule Based Expert Systems: The Mycin Experiments of the Stanford Heuristic Programming Project (The Addison-Wesley series in artificial intelligence)*: Addison-Wesley Longman Publishing Co., Inc.

Butz, C. J., Hua, S., & Maguire, R. B. (2006). A web-based bayesian intelligent tutoring system for computer programming. *Web Intelli. and Agent Sys., 4*(1), 77-97.

Butz, C. J., Hua, S., & Maguire, R. B. (2008). Web-Based Bayesian Intelligent Tutoring Systems. In R. Nayak, N. Ichalkaranje, & L. Jain (Eds.), *Evolution of the Web in Artificial Intelligence Environments* (Vol. 130, pp. 221-242): Springer Berlin Heidelberg.

Byrne, B. M. (2010). Structural equation modeling with AMOS: Basic concepts, applications, and programming. New York: Routledge Academic.

Cabada, R., Barrón Estrada, M., González Hernández, F., & Oramas Bustillos, R. (2014). Intelligent Tutoring System with Affective Learning for Mathematics. In A. Gelbukh, F. Espinoza, & S. Galicia-Haro (Eds.), *Human-Inspired Computing and Its Applications* (Vol. 8856, pp. 483-493): Springer International Publishing.

Carlisle, M. C., Wilson, T. A., Humphries, J. W., & Hadfield, S. M. (2004). RAPTOR: introducing programming to non-majors with flowcharts. *J. Comput. Sci. Coll., 19*(4), 52-60.

Carter, J., & Jenkins, T. (1999). *Gender and programming: what's going on?* Paper presented at the Proceedings of the 4th annual SIGCSE/SIGCUE ITiCSE conference on Innovation and technology in computer science education, Cracow, Poland.

Cohen, Jacob. (1988). Statistical Power Analysis for the Behavioral Sciences, 2nd ed. Hillsdale, NJ: Lawrence Erlbaum.

Conati. (2009). *Intelligent tutoring systems: new challenges and directions*. Paper presented at the Proceedings of the 21st international jont conference on Artifical intelligence, Pasadena, California, USA.

Conati, & Van Lehn. (1999). *Teaching Meta-Cognitive Skills: Implementation and Evaluation of a Tutoring System to Guide Self-Explanation while Learning from Examples*.

Conejo, R., Guzm, E., Mill, E., Trella, n., P, L., DeLa-Cruz, & R, A. (2004). SIETTE: A Web-Based Tool for Adaptive Testing. *Int. J. Artif. Intell. Ed., 14*(1), 29-61.

Corbett, A. T. (2000). *Cognitive Mastery Learning in the ACT Programming Tutor*. Retrieved from

Cordova, D. I., & Lepper, M. R. (1996). Intrinsic motivation and the process of learning: Beneficial effects of contextualization, personalization, and choice. *Journal of Educational Psychology,, 88*, 715-730.

Dale, N., & Weems, C. (2014). *Programming And Problem Solving With C++: Comprehensive Navigate 2 Advantage Access/Print Book Bundle*: Jones and Bartlett Publishers, Inc.

Deterding, S., Sicart, M., Nacke, L., O'Hara, K., & Dixon, D. (2011). *Gamification. using game-design elements in non-gaming contexts*. Paper presented at the CHI '11 Extended Abstracts on Human Factors in Computing Systems, Vancouver, BC, Canada.

Felder, R. M. (1988). Learning and Teaching Styles in Engineering Education. *Journal of Engineering Education, 78*(7), 674-681.

Fossati, D., Di Eugenio, B., Brown, C., & Ohlsson, S. (2008). Learning Linked Lists: Experiments with the iList System. In B. Woolf, E. Aïmeur, R. Nkambou, & S. Lajoie (Eds.), *Intelligent Tutoring Systems* (Vol. 5091, pp. 80-89): Springer Berlin Heidelberg.

Gálvez, J., Guzmán, E., & Conejo, R. (2009). A blended E-learning experience in a course of object oriented programming fundamentals. *Knowledge-Based Systems, 22*(4), 279-286. doi:http://dx.doi.org/10.1016/j.knosys.2009.01.004

Gertner, A. S., Conati, C., & VanLehn, K. (2006). *Procedural help in Andes: generating hints using a Bayesian network student model*. Paper presented at the Proceedings of the fifteenth national/tenth conference on Artificial intelligence/Innovative applications of artificial intelligence, Madison, Wisconsin, USA.

Gonz, C., Mora, A., & Toledo, P. (2014). *Gamification in intelligent tutoring systems*. Paper presented at the Proceedings of the Second International Conference on Technological Ecosystems for Enhancing Multiculturality, Salamanca, Spain.

Graesser, A. C., Chipman, P., Haynes, B. C., & Olney, A. (2005). AutoTutor: an intelligent tutoring system with mixed-initiative dialogue. *Education, IEEE Transactions on, 48*(4), 612-618. doi:10.1109/TE.2005.856149

Guzman, E., & Conejo, R. (2005). Self-assessment in a feasible, adaptive web-based testing system. *Education, IEEE Transactions on, 48*(4), 688-695. doi:10.1109/TE.2005.854571

Hafidi, M., & Bensebaa, T. (2014). Developing Adaptive and Intelligent Tutoring Systems (AITS): A General Framework and Its Implementations. *Int. J. Inf. Commun. Technol. Educ., 10*(4), 70-85. doi:10.4018/ijicte.2014100106

Hevner, A. R., March, S. T., Park, J., & Ram, S. (2004). Design science in information systems research. *MIS Q., 28*(1), 75-105.

Holland, J., Mitrovic, A., & Martin, B. (2009). *J-LATTE: a Constraint-based Tutor for Java.* Paper presented at the Proceedings of the 17th International Conference on Computers in Education [CDROM], Hong Kong.

Hooshyar, Ahmad, & Nasir. (2014). *A framework for automatic text-to-flowchart conversion: A novel teaching aid for novice programmers.* Paper presented at the Computer, Control, Informatics and Its Applications (IC3INA), 2014 International Conference on.

Hooshyar, D., Ahmad, R. B., Nasir, M. H. N. M., Shamshirband, S., & Horng, S.-J. (2015). Flowchart-based programming environments for improving comprehension and problem-solving skill of novice programmers: a survey. *Int. J. Adv. Intell. Paradigms, 7*(1), 24-56. doi:10.1504/ijaip.2015.070343

Howel, K. (2003). First Computer Languages. *Journal of Computing Sciences in Colleges, 18*(4), 317-330.

Hsiao, Brusilovsky, P., & Sosnovsky, S. (2008). *Web-based parameterized questions for object-oriented programming*. Paper presented at the In World Conference on ELearning in Corporate, Government, Healthcare, and Higher Education.

Hsiao, Sosnovsky, S., & Brusilovsky, P. (2010). Guiding students to the right questions: adaptive navigation support in an E-Learning system for Java programming. *Journal of Computer Assisted Learning, 26*(4), 270-283. doi:10.1111/j.1365-2729.2010.00365.x

Jackson, G. T., & McNamara, D. S. (2013). Motivation and Performance in a Game-Based Intelligent Tutoring System. *Journal of Educational Psychology,*. doi:doi: 10.1037/a0032580

Jacqui Chetty, & Barlow-Jones, G. (2015). Novice Students and Computer Programming: Toward Constructivist Pedagogy *Mediterranean Journal of Social Sciences, 5*(14), 240-251.

Jameson, A. (1995). Numerical uncertainty management in user and student modeling: An overview of systems and issues. *User Modeling and User-Adapted Interaction, 5*(3-4), 193-251. doi:10.1007/BF01126111

Johnson, W. L. (1990). Understanding and debugging novice programs. *Artif. Intell., 42*(1), 51-97. doi:10.1016/0004-3702(90)90094-g

Jonassen, D. (2000). Toward a design theory of problem solving. *Educational Technology Research and Development, 48*(4), 63-85. doi:10.1007/BF02300500

Kalelio, F. (2015). A new way of teaching programming skills to K-12 students. *Comput. Hum. Behav., 52*(C), 200-210. doi:10.1016/j.chb.2015.05.047

Kalelioğlu, F. (2015). A new way of teaching programming skills to K-12 students: Code.org. *Computers in Human Behavior, 52*, 200-210. doi:http://dx.doi.org/10.1016/j.chb.2015.05.047

Khalife, J. T. (2006). *Threshold for the introduction of programming: providing learners with a simple computer model.* Paper presented at the Information Technology Interfaces, 2006. 28th International Conference on.

Klir, G. J., & Yuan, B. (1995). *Fuzzy sets and fuzzy logic: theory and applications*: Prentice-Hall, Inc.

Kodaganallur, V., Weitz, R. R., & Rosenthal, D. (2005). A Comparison of Model-Tracing and Constraint-Based Intelligent Tutoring Paradigms. *Int. J. Artif. Intell. Ed., 15*(2), 117-144.

Koedinger, K. R. (2001). *Cognitive tutors as modeling tool and instructional model.* Paper presented at the Smart Machines in Education: The Coming Revolution in Educational Technology, Menlo Park, CA.

Laamarti, F., Eid, M., & Saddik, A. E. (2014). An overview of serious games. *Int. J. Comput. Games Technol., 2014*, 11-11. doi:10.1155/2014/358152

Lahtinen, E., Ala-Mutka, K., & Hannu-Matti. (2005). A study of the difficulties of novice programmers. *SIGCSE Bull., 37*(3), 14-18. doi:10.1145/1151954.1067453

Lakanen, A.-J., & Isom, V. (2015). *What Does It Take to Do Computer Programming?: Surveying the K-12 Students' Conceptions*. Paper presented at the Proceedings of the 46th ACM Technical Symposium on Computer Science Education, Kansas City, Missouri, USA.

Latham, A., Crockett, K., McLean, D., & Edmonds, B. (2012). A conversational intelligent tutoring system to automatically predict learning styles. *Computers & Education, 59*(1), 95-109. doi:http://dx.doi.org/10.1016/j.compedu.2011.11.001

Li, Y., Zhao, K., & Xu, W. (2015). Developing an Intelligent Tutoring System that has Automatically Generated Hints and Summarization for Algebra and Geometry. *Int. J. Inf. Commun. Technol. Educ., 11*(2), 14-31. doi:10.4018/ijicte.2015040102

M. McCracken, V., Almstrum, D., Diaz, M., Guzdial, D., Hagan, Y. B., Kolikant, C., . . . Wilusz. (2001). *A multinational, mult i-institutional study of assessment of programming skills of first -year CS students.* Paper presented at the The 6th Annual Conference on Innovation and Technology in Computer Science Education Kent, United Kngdm.

Ma, L., Ferguson, J., Roper, M., & Wood, M. (2011). Investigating and improving the models of programming concepts held by novice programmers. *Computer Science Education, 21*(1), 57-80. doi:10.1080/08993408.2011.554722

Mciver, L., & Conway, D. (2000). *The Effect of Programming Language on Error Rates of Novices*. Paper presented at the 12th Workshop of the Psychology of Programming Interest Group (ICCE99), Cozenza - Italy.

McNamara, D. S., Jackson, G. T., & Graesser, A. C. (2010). *Intelligent tutoring and games (ITaG).* Paper presented at the Gaming for classroom-based learning: Digital role-playing as a motivator of study, Hershey, PA.

Microsoft. (2009). Visio 2007. doi:10.1145/268809.268853

Microsoft. (2013). Microsoft bayesian network editor.

Mitrovic, A. (2003). An Intelligent SQL Tutor on the Web. *Int. J. Artif. Intell. Ed., 13*(2-4), 173-197.

Moser, R. (1997). A fantasy adventure game as a learning environment: why learning to program is so difficult and what can be done about it. *SIGCSE Bull., 29*(3), 114-116. doi:10.1145/268809.268853

Nilson, N. (1998). *Artificial Intelligence: A New Synthesis*. USA: Morgan Kaufmann.

Nwana, H. (1990). Intelligent tutoring systems: an overview. *Artificial Intelligence Review, 4*(4), 251-277. doi:10.1007/BF00168958

Ohlsson, S. (1992). Constraint-based student modelling. *Journal of Artificial intelligence in Education, 3*(4), 429-429.

Pearl, J. (1988). *Probabilistic reasoning in intelligent systems: networks of plausible inference*: Morgan Kaufmann Publishers Inc.

Peffers, K., Tuunanen, T., Rothenberger, M., & Chatterjee, S. (2007). A Design Science Research Methodology for Information Systems Research. *J. Manage. Inf. Syst., 24*(3), 45-77. doi:10.2753/mis0742-1222240302

Perkins, D., Hancock, C., Hobbs, R., Martin, F., & Simmonds, R. (1988). *Conditions of Learning In Novice Programmers*. Paper presented at the Studying the Novice Programmer, Hillsdale - NJ - USA.

Pertti. (2007). Action Research is Similar to Design Science. *Quality & Quantity, 41*(1), 37-54. doi:10.1007/s11135-005-5427-1

Pillay, N. (2003). Developing intelligent programming tutors for novice programmers. *SIGCSE Bull., 35*(2), 78-82. doi:10.1145/782941.782986

Pillay, N., & Jugoo, V. R. (2005). An investigation into student characteristics affecting novice programming performance. *SIGCSE Bull., 37*(4), 107-110. doi:10.1145/1113847.1113888

Rai, D., & Beck, J. E. (2012). *Math learning environment with game-like elements: an incremental approach for enhancing student engagement and learning effectiveness*. Paper presented at the Proceedings of the 11th international conference on Intelligent Tutoring Systems, Chania, Crete, Greece.

Ramalingam, V., LaBelle, D., & Wiedenbeck, S. (2004). *Self-efficacy and mental models in learning to program*. Paper presented at the Proceedings of the 9th annual SIGCSE conference on Innovation and technology in computer science education, Leeds, United Kingdom.

Rickel, J., & Johnson, W. L. (2000). Task-oriented collaboration with embodied agents in virtual worlds *Embodied conversational agents* (pp. 95-122): MIT Press.

Robins, A., Rountree, J., & Rountree, N. (2003). Learning and Teaching Programming: A Review and Discussion. *Computer Science Education, 13*(2), 137-172. doi:10.1076/csed.13.2.137.14200

Rowe, J. P., Shores, L. R., Mott, B. W., & Lester, J. C. (2011). Integrating learning, problem solving, and engagement in narrative-centered learning environments. *Int. J. Artif. Intell. Ed., 21*(1-2), 115-133. doi:10.3233/jai-2011-019

Rutherford, F. J. (1991). Science for all americans.

Sabourin, J., Rowe, J., Mott, B., & Lester, J. (2011). When Off-Task is On-Task: The Affective Role of Off-Task Behavior in Narrative-Centered Learning Environments. In G. Biswas, S. Bull, J. Kay, & A. Mitrovic (Eds.), *Artificial Intelligence in Education* (Vol. 6738, pp. 534-536): Springer Berlin Heidelberg.

Sack, W., Soloway, E., & Weingrad, P. (1992). *From PROUST to CHIRON: Its design as iterative engineering: Intermediate results are important*: Lawrence Erlbaum Associates, Hillsdale, NJ.

Schank, R. C., & Cleary, C. (1995). *Engines for education*. New Jersey: Lawrence Erlbaum.

Scott, A. (2010). *Using Flowcharts, Code and Animation for Improved Comprehension and Ability in Novice Programming.* (Doctor of Philosophy), University of Glamorgan, UK.

Snow, E. L., Jackson, G. T., Varner, L. K., & McNamara, D. S. (2013). *The impact of performance orientation on students' interactions and achievements in an ITS.* Paper presented at the Proceedings of the 26th annual Flordia artificial intelligence research society (FLAIRS) Menlo Park, CA.

Soloway, E., & Spohrer, J. C. (1988). *Studying the Novice Programmer*: L. Erlbaum Associates Inc.

Song, J. S., Hahn, S. H., Tak, K. Y., & Kim, J. H. (1997). An intelligent tutoring system for introductory C language course. *Comput. Educ., 28*(2), 93-102. doi:10.1016/s0360-1315(97)00003-1

Stanford. (2014). Stanford online parser.

Stroustrup, B. (2000). *The C++ Programming Language*: Addison-Wesley Longman Publishing Co., Inc.

Swain, N., Moses, W., Anderson, J. A., & Davis, C. T. (2013). *RAPTOR - A Vehicle to Enhance Logical Thinking*. Paper presented at the 120th ASEE Annual Conference & Exposition Atlanta, Georgia.

Sykes, E. R. (2007). Developmental Process Model for the Java Intelligent Tutoring System. *Journal of Interactive Learning Research, 18*(3), 399-410.

Sykes, E. R., & Franek, F. (2003). *A prototype for an intelligent tutoring system for students learning to program in JavaTM*. Paper presented at the In Advanced Learning Technologies, Athens, Greece.

Teague, D. M., & Roe, P. (2009). *learning to program, Pair programming, collaborative learning*. Paper presented at the Proceedings of the First International Conference on Computer Supported Education, Lisboa, Portugal.

Tegos, S., Demetriadis, S., & Tsiatsos, T. (2014). A Configurable Conversational Agent to Trigger Students' Productive Dialogue: A Pilot Study in the CALL Domain.

International Journal of Artificial Intelligence in Education, 24(1), 62-91. doi:10.1007/s40593-013-0007-3

Vygotsky, L. S. (1978). *Mind in society: The development of higher psychological process*. Cambridge, MA:: Harvard University Press.

Wang, T.-I., Wang, K.-T., & Huang, Y.-M. (2008). Using a style-based ant colony system for adaptive learning. *Expert Syst. Appl., 34*(4), 2449-2464. doi:10.1016/j.eswa.2007.04.014

Weragama, D., & Reye, J. (2013). The PHP Intelligent Tutoring System. In H. C. Lane, K. Yacef, J. Mostow, & P. Pavlik (Eds.), *Artificial Intelligence in Education* (Vol. 7926, pp. 583-586): Springer Berlin Heidelberg.

Westphal, B. T., Harris, J., F.C. , & Fadali, M. S. (2003). *"Graphical Programming: A Vehicle for Teaching Computer Problem Solving*. Paper presented at the In Proceedings of the 33rd ASEE/IEEE Frontiers in Education Conference, Boulder, CO, USA.

Winslow, L. E. (1996). Programming pedagogy & mdash;a psychological overview. *SIGCSE Bull., 28*(3), 17-22. doi:10.1145/234867.234872

Wong, S. K. M., & Butz, C. J. (2001). Constructing the dependency structure of a multiagent probabilistic network. *Knowledge and Data Engineering, IEEE Transactions on, 13*(3), 395-415. doi:10.1109/69.929898

Wong, S. K. M., Butz, C. J., & Wu, D. (2000). On the implication problem for probabilistic conditional independency. *Systems, Man and Cybernetics, Part A: Systems and Humans, IEEE Transactions on, 30*(6), 785-805. doi:10.1109/3468.895901

Woolf. (2008). *Building intelligent interactive tutors: Student-centered strategies for revolutionizing e-learning*. USA: Morgan Kaufmann.: Burlington, MA.

Woolf, Burleson, W., Arroyo, I., Dragon, T., Cooper, D., & Picard, R. (2009). Affect aware tutors: recognising and responding to student affect. *Int. J. Learn. Technol., 4*(3/4), 129-164. doi:10.1504/ijlt.2009.028804

Woolf, B. P., & Hall, W. (1995). Multimedia pedagogues: interactive systems for teaching and learning. *Computer, 28*(5), 74-80. doi:10.1109/2.384121

Zadeh, L. A. (1986). A simple view of the Dempster-Shafer theory of evidence and its implication for the rule of combination. *AI Mag., 7*(2), 85-90.

LIST OF PUBLICATIONS AND PAPERS PRESENTED

Doctoral Symposium

1. 23rd Australasian Software Engineering Conference (ASWEC) 7 - 10 April 2014, Sydney, Australia
2. University of Malaya, postgraduate research excellence symposium, Concorde Hotel Shah Alam, Kuala Lumpur, Malaysia, 2014

Conference Presentations

1. Flowchart-based Approach to Aid Novice Programmers: A Novel Framework, the Second International Conference on Computer and Information Science, Kuala Lumpur, Malaysia, 3-5 June 2014 (IEEE-indexed)
2. A Framework for Automatic Text-to-Flowchart Conversion: A Novel Teaching Aid for Novice Programmers, The International Conference on Computer, Control, Informatics and its Applications, Bandung, Indonesia, 21-23 October 2014 (IEEE-indexed)
3. Flowchart-based Bayesian Intelligent Tutoring System for Computer Programming, International Conference on Smart Sensors and Application (ICSSA), Kuala Lumpur, Malaysia, 26-27 May 2015 (IEEE-indexed)
4. Improving web-based problem solving skills of novice programmers with a novel game-based intelligent tutoring system, The 2015 International Conference on Science in Information Technology (ICSITech 2015) , Yogyakarta, Indonesia, 27-8 October 2015 (IEEE-indexed)

Journal Publications (Scopus Indexed)

1. Flowchart-based programming environments for improving comprehension and problem-solving skill of novice programmers: a survey. Int. J. Advanced Intelligence Paradigms, Vol. 7(1), pp: 24-56 (2015).

Journal Publications (ISI Indexed)

1. Applying an Online Game-based Formative Assessment in a Flowchart-based Intelligent Tutoring System for Improving Problem-Solving Skills. Computers & Education, 2015, In Press.
2. A flowchart-based intelligent tutoring system for improving problem-solving skills of novice programmers. Journal of Computer Assisted Learning, Vol 31(4), pp: 345-361, 2015.
3. A Flowchart-based Multi-Agent System for Assisting Novice Programmers with Problem Solving Activities. Malaysian Journal of Computer Science. Vol 28(2), pp: 132-151, 2015
4. A Flowchart-based Programming Environment for Improving Problem Solving Skills of Cs minors in computer programming. Asia Life Science. Vol 24(2), pp: 629-646, 2015