

DEVELOPMENT OF PROGRAM FLOWCHART DRAWING TOOL

WONG KHAI MENG

**A project report submitted in partial fulfilment of the
requirements for the award of Bachelor of Engineering
(Hons.) Software Engineering**

**Faculty of Engineering and Science
Universiti Tunku Abdul Rahman**

May 2016

DECLARATION

I hereby declare that this project report is based on my original work except for citations and quotations which have been duly acknowledged. I also declare that it has not been previously and concurrently submitted for any other degree or award at UTAR or other institutions.

Signature : _____

Name : _____

ID No. : _____

Date : _____

APPROVAL FOR SUBMISSION

I certify that this project report entitled “**DEVELOPMENT OF PROGRAM FLOWCHART DRAWING TOOL**” was prepared by **WONG KHAI MENG** has met the required standard for submission in partial fulfilment of the requirements for the award of Bachelor of Science (Hons.) Software Engineering at Universiti Tunku Abdul Rahman.

Approved by,

Signature : _____

Supervisor : _____

Date : _____

The copyright of this report belongs to the author under the terms of the copyright Act 1987 as qualified by Intellectual Property Policy of Universiti Tunku Abdul Rahman. Due acknowledgement shall always be made of the use of any material contained in, or derived from, this report.

© 2016, Wong Khai Meng. All rights reserved.

DEVELOPMENT OF PROGRAM FLOWCHART DRAWING TOOL

ABSTRACT

Novices faced extreme difficulties when trying to learn programming. Despite having numerous advanced Integrated Development Environments (IDEs) available, the learning process was still slow and difficult for them. This project was conducted to produce a learning tool targeted at programming novices in order to minimize the difficulties of the learning process. As flowchart drawing was part of the learning process, this project aimed to introduce the basic concepts of programming through the construction of flowcharts using a flowchart drawing tool. As a result of this project, a flowchart drawing tool was produced. The developed tool featured several core components, namely flowchart project loading and saving, flowchart printing, flowchart construction, variable declaration, flowchart execution, source code generation, as well as undo and redo performed actions. Then, evaluation was performed on the developed tool by targeted users to solve the prepared questions before feedback was elicited from them. In conclusion, the developed tool was able to allow the participants to create flowcharts and simplify the learning process of preliminary programming skills.

TABLE OF CONTENTS

DECLARATION	ii
APPROVAL FOR SUBMISSION	iii
ABSTRACT	v
TABLE OF CONTENTS	vi
LIST OF TABLES	x
LIST OF FIGURES	xi
LIST OF SYMBOLS / ABBREVIATIONS	xiii
LIST OF APPENDICES	xiv

CHAPTER

1	INTRODUCTION	1
	1.1 Background	1
	1.1.1 Problem Statement	2
	1.1.2 Proposed Approach/Solution	4
	1.2 Aims and Objectives	4
	1.2.1 Goals	4
	1.2.2 Objectives	5
	1.2.3 Project Scope	5
	1.3 Conclusion	6
2	LITERATURE REVIEW	7
	2.1 Introduction	7
	2.2 Existing Flowchart Drawing Tools	7
	2.2.1 Progranimate	8

2.2.2	RAPTOR	11
2.2.3	ProGuide	14
2.2.4	Iconic Programmer	17
2.2.5	FLINT	19
2.2.6	SICAS	22
2.2.7	B#	24
2.2.8	Comparison	27
2.3	Proposed Features	28
2.4	Existing Development Methodologies	28
2.4.1	Agile Development (Feature Driven Development)	29
2.4.2	Iterative and Incremental Development	32
2.4.3	Rapid Application Development (RAD)	33
2.4.4	Spiral Development	33
2.5	Existing Software Platforms	34
2.5.1	Windows Forms (WinForms)	34
2.5.2	Windows Presentation Foundation (WPF)	37
2.5.3	JavaFX	39
2.6	Conclusion	43
3	PROJECT METHODOLOGY AND PLAN	44
3.1	Implementation of Chosen Development Methodology	44
3.1.1	Develop an Overall Model	44
3.1.2	Build a Features List	44
3.1.3	Plan by Feature	45
3.1.4	Design by Feature	45
3.1.5	Build by Feature	45
3.2	Implementation of Chosen Software Platform	45
3.3	Project Plan	46
3.4	Conclusion	47
4	PROJECT SPECIFICATIONS	48
4.1	Fact Finding	48

4.1.1	Interview	48
4.1.2	Survey	49
4.2	User Requirements	50
4.2.1	Functional Requirements	50
4.2.2	Non-Functional Requirements	50
4.3	User Interface Design	51
4.3.1	Menu Bar	52
4.3.2	Toolbox	52
4.3.3	Variable Inspector	53
4.3.4	Console	53
4.3.5	Flowchart Designer	54
4.4	Storyboard	54
4.4.1	Scenario 1: Draw a Flowchart	54
4.4.2	Scenario 2: Create, Edit and Display Variable	55
4.4.3	Scenario 3: Execute Flowchart	55
4.4.4	Scenario 4: Generate Source Code	55
4.5	Use Case Modelling	55
4.5.1	Use Case Diagram	55
4.5.2	Use Case Descriptions	56
4.6	FDD Deliverables	57
4.6.1	Develop an Overall Model	57
4.6.2	Build a Features List	57
4.6.3	Plan by Feature	57
4.6.4	Design by Feature	58
4.7	Conclusion	61
5	PROJECT IMPLEMENTATION AND TESTING	62
5.1	Class Diagrams	62
5.2	Version Control	62
5.3	Testing	63
5.3.1	Unit Testing	63
5.3.2	User Testing	63
5.4	Conclusion	65

6	CONCLUSION AND RECOMMENDATIONS	66
6.1	Contribution	66
6.2	Limitations	67
6.3	Future Enhancements	68
6.4	Conclusion	68
	REFERENCES	69
	APPENDICES	72

LIST OF TABLES

TABLE	TITLE	PAGE
1.1	Existing Flowchart Drawing Tools and Their Weaknesses	3
2.1	Comparison between Flowchart Drawing Tools	27

LIST OF FIGURES

FIGURE	TITLE	PAGE
1.1	Sample Flowchart	1
2.1	Progranimate Screenshot	9
2.2	RAPTOR Main Window Screenshot	12
2.3	RAPTOR Console Window Screenshot	12
2.4	ProGuide Screenshot	15
2.5	Iconic Programmer Screenshot	17
2.6	Top-Down Chart Design Screenshot	20
2.7	FLINT Screenshot	20
2.8	SICAS Screenshot	23
2.9	B# Screenshot	26
2.10	FDD Project Lifecycle	29
2.11	Design Area Screenshot	35
2.12	Code-behind Screenshot	36
2.13	Designer Screenshot	36
2.14	XAML Design Area Screenshot	38
2.15	Code-behind Screenshot	38
2.16	FXML Document Screenshot	40
2.17	FXML Document Controller Screenshot	41
2.18	JavaFX Application Screenshot	41

2.19	JavaFX Scene Builder Screenshot	42
3.1	Project Plan (a)	46
3.2	Project Plan (b)	46
4.1	User Interface Design	51
4.2	Menu Bar	52
4.3	Toolbox	52
4.4	Variable Inspector	53
4.5	Console	53
4.6	Flowchart Designer	54
4.7	Use Case Diagram	56
4.8	Features List	57
4.9	Features List Development Plan	58
4.10	Flowchart Printing Sequence Diagram	58
4.11	Project Management Sequence Diagram	59
4.12	Flowchart Designing Sequence Diagram	59
4.13	Variable Management Sequence Diagram	60
4.14	Flowchart Execution Sequence Diagram	60
4.15	Source Code Generation Sequence Diagram	61
4.16	Action Reversal Sequence Diagram	61
5.1	Number of Commits per Week	63

LIST OF SYMBOLS / ABBREVIATIONS

API	Application Programming Interface
CPU	Central Processing Unit
CSS	Cascading Style Sheets
FLINT	Flowchart Interpreter
GDI/GDI+	Graphics Device Interface
GUI	Graphical User Interface
HTML	HyperText Markup Language
JRE	Java Runtime Environment
IDE	Integrated Development Environment
IT	Information Technology
Mac OS	Macintosh Operating System
MVC	Model-View-Controller
OS	Operating System
RAPTOR	Rapid Algorithmic Prototyping Tool for Ordered Reasoning
RUP	Rational Unified Process
SDLC	Software Development Life Cycle
URL	Uniform Resource Locator
VB6.0	Visual Basic 6.0
VB.NET	Visual Basic .NET
WPF	Windows Presentation Foundation
XAML	eXtensible Application Markup Language

LIST OF APPENDICES

APPENDIX	TITLE	PAGE
A	Interview Questions	72
B	Interview Replies	76
C	Pre-Evaluation Survey Questions	81
D	Pre-Evaluation Survey Results	84
E	Evaluation Questions	90
F	Post-Evaluation Feedback Questions	91
G	Post-Evaluation Survey Results	93
H	Storyboards	96
I	Use Case Descriptions	101
J	Class Diagrams	109
K	Unit Tests	120

CHAPTER 1

INTRODUCTION

1.1 Background

A flowchart is a basic and fundamental diagram representing a procedural workflow to exhibit and to convey information in a clear and logical manner. It was first introduced in the 1940s when computers were only starting to be unveiled to the world (Shneiderman, et al., 1977, pp.373-381). It is now widely used by people across many fields, including education, business and sales. Figure 1.1 shows a sample flowchart that displays the current value of the “count” variable for 5 times.

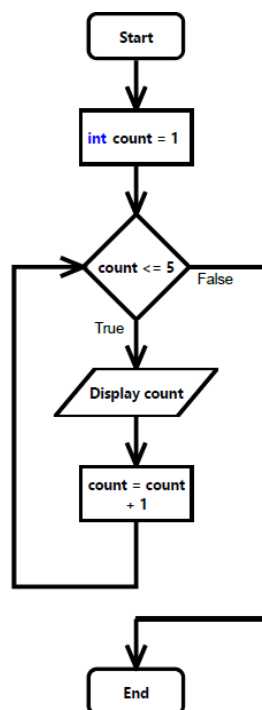


Figure 1.1: Sample Flowchart

In the education industry, particularly during the pursuit of IT knowledge, students will definitely come across the usage of flowcharts to graphically display the logical order for the flow of programs for ease of comprehension, and there were studies conducted indicating that they are quite effective for visual learners in both writing and comprehending algorithms (Hall, 2007, pp.110-111). Moreover, some flowchart drawing tools may be utilised to let students obtain hands-on experience whilst learning and they were also proven to be able to improve learning and recalling more effectively (Xinogalos, 2013, pp.1313-1322).

With the advancement of technologies and the ever-expanding libraries of numerous programming languages that are potential targets for developing this product on, it has become increasingly easier to develop software with complex functionalities. Currently, there are quite a number of software applications as well as websites available online that provide flowchart drawing services, either free or paid. The services provided range from basic shape drawings, available at websites such as www.draw.io and www.glimpy.com, to educational flowcharts from applications like Proanimate, Raptor, Flint, B#, BACCII, SFC Editor, SICAS, ProGuide and Iconic Programmer (Xinogalos, 2013, pp. 1313-1322), to high-level flowcharts used in businesses, such as swimlane flowcharts, event-driven process chain diagrams, workflow diagrams, process map and many others (SmartDraw, n.d.).

This project proposes a new flowchart drawing tool designed specifically for students as they learn the programming fundamentals. The tool will feature an interactive and user-friendly interface. Students could be able to draw flowcharts, animate the flowchart, generate source code, create variables and declare data types using the tool.

1.1.1 Problem Statement

Students with no prior knowledge of programming as well as having a weak logical thinking would face a hard time to understand the fundamentals of programming. Additionally, students are also vulnerable in the algorithmic problem solving aspects

of programming involving the basics of sequence selection and iteration (Scott, n.d.). Although the existing education system incorporated the usage of flowcharts, they are written on paper and are static, which does not provide any assistance in improving comprehension in neither the dynamic nature of program execution nor the control structures (Xinogalos and Satratzemi 2004, pp. 60-65). This can be related to the time when I was just starting to learn the fundamentals of programming, whereby the usage of flowcharts in my studies is limited.

When flowcharts were introduced to the learning process, I was taught to write and to memorize the symbols with their usage and was required to mentally walk through the processes of the flowcharts to validate their correctness. This was hard to conduct as when there are many variables that require mental tracking, I tended to lose track of the current values that they contain. Moreover, as my understanding of data types and variables was still weak back then, I may have created syntactically incorrect statements as I may have accidentally combined variables of different data types together when I should not have did it.

Even though there are currently quite a number of existing tools available for drawing flowcharts, there are some flaws in them that disrupt the entire learning process for the programming fundamentals. Table 1.1 shows a list of the existing flowchart drawing tools along with their weaknesses.

Table 1.1: Existing Flowchart Drawing Tools and Their Weaknesses

Weakness	Flowchart Drawing Tools
No longer supported	FLINT
Provides only basic flowchart drawing	www.draw.io, www.gliffy.com, Microsoft Visio
Not available for free	BACCII, BACII++, B#, FLINT, ProGuide, SICAS, SICAS-COL, H-SICAS, Microsoft Visio
Not available in English language	Flowchart, Portugol IDE, Visual Flowchart
Automatic source code generator unavailable	FLINT, ProGuide
Flowchart symbols are not common symbols	BACCII, BACII++, B#, Iconic Programmer
Program animation unavailable	BACCII, BACII++, SFC Editor
Universal data type	RAPTOR, Iconic Programmer

Thus, this project is to create a flowchart drawing tool targeted at students that unifies the strengths and eliminates the weaknesses of the existing tools.

1.1.2 Proposed Approach/Solution

The proposed solution is the development of a desktop tool that provides students the ability to swiftly draw flowcharts in an interactive GUI. Besides that, it will feature flowchart execution and display the current values and outputs of the flowchart. Then, the tool could provide instant feedback whenever any syntax error or data type mismatch occurred while designing the flowchart. Next, it will feature the ability to generate the flowchart's equivalent in pseudocode or source codes, such as C or C++ languages.

1.2 Aims and Objectives

1.2.1 Goals

This project aims to produce a flowchart drawing tool that could be used by students to draw flowcharts and to simplify as well as to improve the process of learning preliminary programming skills.

1.2.2 Objectives

This project aims to produce a flowchart drawing tool with the following objectives:

- To display the process flow of the program created by students.
- To evaluate the usefulness and effectiveness of the tool produced.
- To validate the accuracy of algorithms devised by students to solve their problems
- To improve effectiveness of drawing flowcharts, by reducing error rates while drawing flowcharts
- To improve efficiency of drawing flowcharts, by saving time in the drawing of flowcharts
- To incorporate core programming principles, such as variable declarations, data types, control structures, and correct programming syntax
- To provide the ability to execute visual programs, similar to coded programs
- To provide the ability to generate the designed flowchart's equivalent in program code.

1.2.3 Project Scope

This project is targeted at students with no programming background who are currently learning the basics of programming. This project aims to create a flowchart drawing tool that could assist students during flowchart training sessions. The tool will feature an interactive GUI, where students could perform drag-and-drop operations to create flowchart symbols in flowcharts.

The tool would also feature flowchart execution in one go in order to display the values of variables and outputs produced by the designed flowchart. This allows students to easily determine whether the flowchart produced behaved exactly as they expected, or worked incorrectly due to inaccurate flowchart symbols being inserted or mistakes in the statements produced. Additionally, the tool will also provide

instant visible feedback on any syntax errors produced during the flowchart setup. Whenever there are any data type mismatches or invalid operators being used in any flowcharts' statements, the tool would notify the user about the issues produced and prevent the statements from being updated until they are resolved. Moreover, incorrect variable naming conventions will also prevent the students from creating the variables until they conform to the required syntax.

The tool would also be able to generate and display the designed flowcharts' equivalent in source code. This enables students to view and learn the syntax of writing the source code of specific programming languages easily. The source code that would be supported is C++. Additionally, the source codes produced would be syntactically correct and are able to be copied and executed in an IDE to view as well as to verify the outputs.

The scope of this project will not include the implementation of advanced flowchart symbols like functions, and nested decisions, nested loops and loops in decision statements due to time constraints and complexity. Moreover, the tool will also not include the generation of other source codes from the flowcharts produced, such as pseudocode, Java and C#. Besides that, variables with array data types will not be included as well, as it is slightly too advanced for novices.

1.3 Conclusion

There are many difficulties faced by novice programmers when they start to learn the programming fundamentals. Although there are existing tools that were designed to tackle the problems, most of these tools still lack features to cover all the basic knowledge required by students to advance to the next stage in programming. Thus, this project is designed to produce a new flowchart drawing tool that fulfils the requirements for learning basic programming.

CHAPTER 2

LITERATURE REVIEW

2.1 Introduction

In this chapter, only existing flowchart drawing tools that have the program animation ability, a feature that executes the flowchart like a normal program, were studied and evaluated extensively. Literature review was done by including well-known and well-cited papers, as well as conducting fact finding with lecturers and students. On top of that, a table of comparison was produced to compare and contrast between the existing flowchart drawing tools. Additionally, suitable development methodologies and development platforms were compared and contrasted.

2.2 Existing Flowchart Drawing Tools

Currently, there are several flowchart drawing tools that are developed for educating novices on the fundamentals of programming. They are aimed at making the initial stages of programming easier for novices by removing the complexities linked to professional development environments and syntax writing, allowing them to focus solely on the algorithmic problem solving of programming (Scott, n.d.). Then, these tools enable users to create a simple yet complete program using just flowchart representations alone. For novice programmers, this is vital as most of them could comprehend program algorithms more easily using visual representations rather than looking through program code (Hall, 2007, pp.110-111).

2.2.1 Progranimate

Progranimate is an interactive and dynamic visualisation programming tool targeted towards novice programmers (Scott, n.d.). It emphasizes on enabling novices to focus solely on their key weakness, which is the algorithmic problem solving aspects of programming involving the basics of sequence selection and iteration (Scott, n.d.). Progranimate is developed by Dr Andrew Scott, an assistant professor from the University of Western Carolina.

Development Environment

Progranimate is developed in the Java programming language. It is free and can either be installed as a stand-alone application or deployed over the Internet via either Java web start or Java applet, removing the hassles of requiring installation of applications into the local computers of users, but requiring an installation of the JRE to continue.

Progranimate can be run on multiple operating systems, including Windows and Mac OS, as it is platform independent. This makes it available for use anywhere regardless of the operating systems of the computers that the novice programmers possess.

Progranimate employs the imperative-procedural programming technique to teach novices on the fundamentals. Additionally, Progranimate uses both drag-and-drop functionality and code modification to produce flowcharts.

Progranimate is separated into 3 major sections, with the flowchart and its symbols positioned at the left, the code generator in the middle and the variable and array inspectors at the right. The flowchart acts as the visual metaphor to visualize the programming and execution process while the code generator will display the syntactically correct code for the related programming language. Then, the variable inspector is used to keep track on the variables declared and their respective values in

the project while the array inspector is used to record all the values of the arrays. Figure 2.1 shows a screenshot of Progranimate.

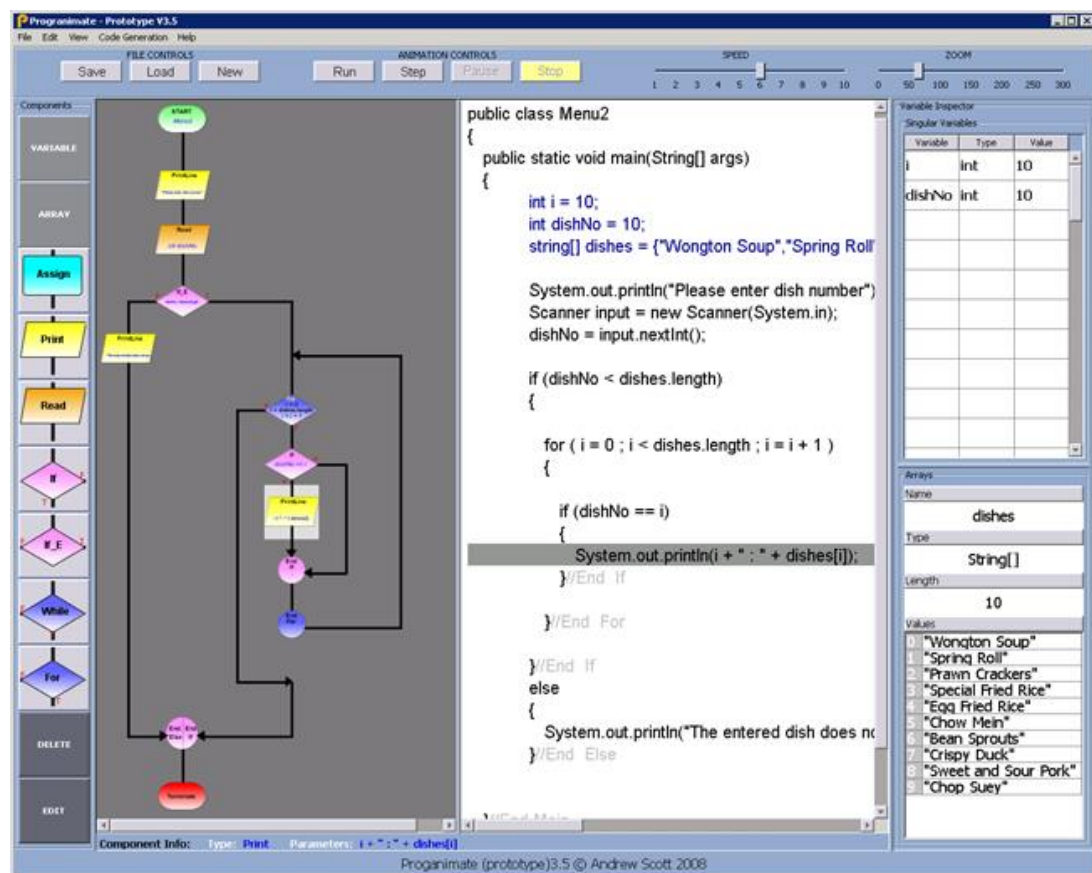


Figure 2.1: Progranimate Screenshot (Scott, n.d.)

Features

Progranimate uses common flowchart symbols when designing flowcharts. Besides that, different flowchart symbols are displayed using different colors. This allows for easier distinction between flowchart symbols, thus reducing confusion and making it easier to locate the key functionalities of the program and the algorithm modelled in Progranimate (Scott, Watkins and McPhee, 2008b, pp.1-6).

Progranimate supports the ability to automatically generate source code simultaneously with flowcharts when symbols are being added and removed. This provides novices with the chance to learn about the syntax for the produced symbols'

equivalent in source codes. Progranimate provides support on a range of languages, including Java-like pseudocode, Java, VB.NET, VB6.0, Pascal and JavaScript.

Progranimate offers the uniqueness of being able to perform synchronization between the flowchart and the source code generated by highlighting both the currently focused flowchart symbol with its related row in the source code. This feature enables novices to learn and to recognize easily the source code and the syntax required to be written in order to perform the expected results as produced through creating the flowchart symbol.

Progranimate provides the ability to animate program flow by enabling flowchart execution. When the program is executed, the currently executing line of code and its relative flowchart representation are both highlighted simultaneously to notify that they will be the ones being executed next. Besides that, the execution of the program could be paused to allow manual execution. During the execution of the program, the variable inspector will be keeping tabs on the variables that have been declared along with their current values up until that point. In this way, novices could observe the changes that would occur on the variables affected when the next statement of the program is executed.

Progranimate also supports the declaration of variables and arrays in specific data types, such as integer, double, character, string and boolean. For arrays, the size and possible initial values that they may contain could also be set.

Limitations

Amongst the list of codes that could be generated, Progranimate fails to feature the ability to generate codes in C or C++ language, which are also popular choices for teaching the fundamentals of programming. Additionally, as it is developed in the Java language, the execution speed of the tool will be slower than other tools developed in C++ and C# language (Scott, Watkins and McPhee, 2008a, pp.498-508).

Evaluation

There were initial studies conducted on 185 students ranging from high school to undergraduate studies to assess the usefulness of Progranimate. Based on the preliminary findings, Progranimate was shown to be effective in all of the three categories being evaluated on, namely usability, efficacy and problem solving exercises (Scott, Watkins and McPhee, 2008b, pp.1-6).

2.2.2 RAPTOR

RAPTOR is a free flowchart-based programming environment designed specifically to assist novices in algorithm visualization (Carlisle, et al., 2005, pp.176-180). It allows novices to create algorithms just by utilising and compositing basic flowcharts symbols. RAPTOR is developed through collaboration efforts between Martin Carlisle, Terry Wilson, Jeffrey Humphries and Steven Hadfield from the Department of Computer Science at the United States Air Force Academy, and is currently maintained by Professor Martin Carlisle.

Development Environment

RAPTOR employs both the imperative-procedural and object-oriented programming techniques, whereby the former is targeted at novice and intermediate programmers while the latter is targeted at experienced programmers. It uses drag-and-drop functionality to modify the symbols of the flowchart. It is developed in a combination of programming languages, including Ada and C#, and runs as part of the .NET Framework. It can be run only on the Windows operating system, but it could also be executed in the Ubuntu operating system, albeit with some features removed due to compatibility issues.

RAPTOR is separated into two windows, with one being the main window and the other being the console window. In the main window, the flowchart symbols and variable inspector are visible on the left column while the rest of the space is occupied by the flowchart being designed. The console window is used to display the outputs produced from the executing flowchart. Figures 2.2 and 2.3 show the screenshots of the main window and console window of RAPTOR respectively.

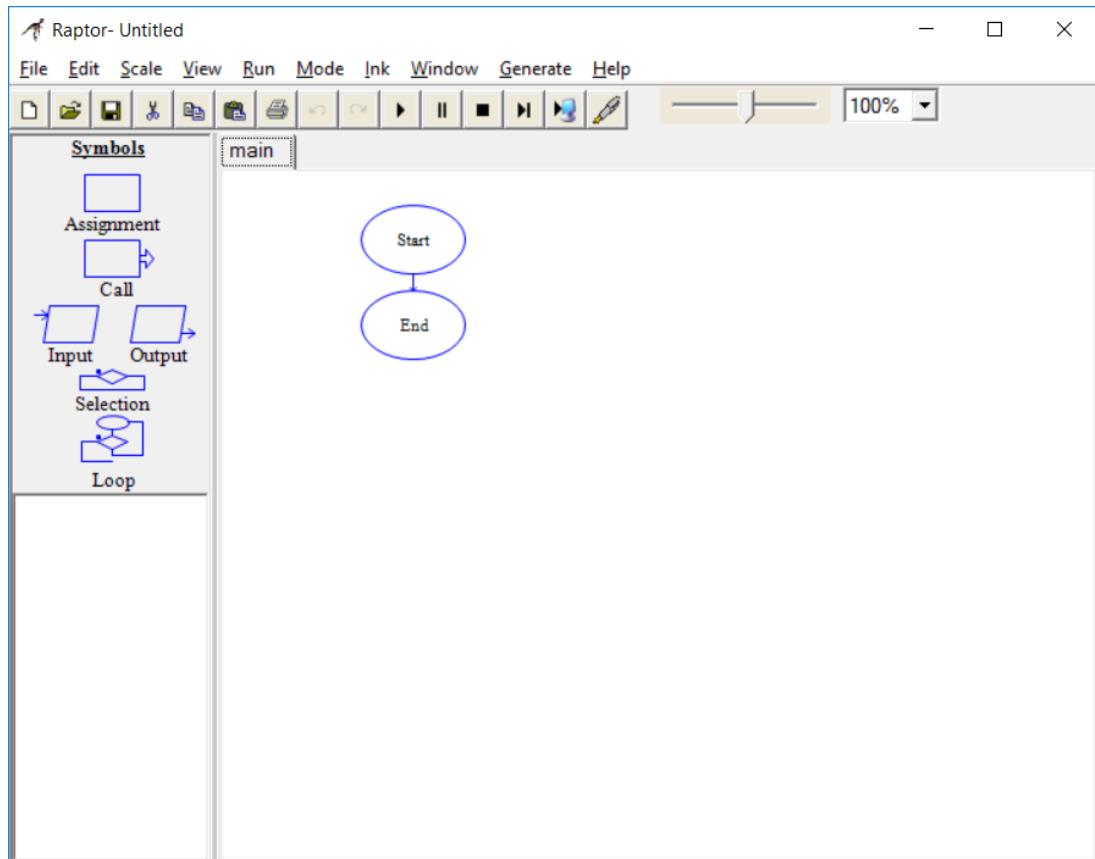


Figure 2.2: RAPTOR Main Window Screenshot

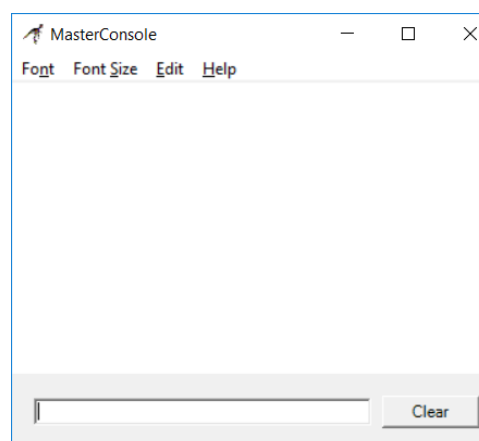


Figure 2.3: RAPTOR Console Window Screenshot

Features

RAPTOR has several built-in functions and features that generates random numbers, perform trigonometric calculations, manipulating time functions, drawing graphics and interfacing with pointing devices (Xinogalos, 2013, pp. 1313-1322). Additionally, it supports an auto-completion feature in creating procedure calls and provides the ability to develop additional procedures that could be called similar to built-in procedures (Xinogalos, 2013, pp. 1313-1322). Besides that, RAPTOR enforces syntax checking during the construction of flowcharts, thus preventing the creation and execution of a syntactically incorrect program.

RAPTOR supports the ability to generate syntactically correct source code for the flowchart produced. The range of languages it could generate includes Ada, C#, C++ and Java. Moreover, additional languages could be supported by developing a C# class for the other languages.

RAPTOR also offers the uniqueness of being able to create comments on the flowchart directly. When comments are created, they appear as “talking bubbles” next to the flowchart symbols that they were created for.

RAPTOR allows program animation through flowchart execution, similar to Proanimate. It highlights the currently executing flowchart with a different colour, enables manual walkthrough of the program and supports tracking of value for the variables declared in the program.

Limitations

RAPTOR has the concept of creating weakly typed variables, whereby the same variable used to store a value in one specific data type, could also be used to store a value in another data type. This will prevent novices from learning the concept of declaring strongly typed variables, which is also part of the core in understanding programming.

Evaluation

An initial assessment on RAPTOR was done on three consecutive semesters of a course in the US Air Force Academy during the years 2003 to 2004. The result of the assessment showed that RAPTOR was more effective in developing problem solving capabilities and understanding flow of control in students (Xinogalos, 2013, pp. 1313-1322). However, when one of the exams taken by the students required the usage of arrays, students' performance was bad because arrays in RAPTOR were declared implicitly (Carlisle, Wilson, Humphries and Hadfield, 2005, pp. 176-180).

2.2.3 ProGuide

ProGuide is an educational environment designed to support students in problem solving activities (Areias and Mendes, 2007, p. 89) and it uses a tutoring system model (Xinogalos, 2013, pp.1313-1322). It is a dialogue based tool to support weaker students to create basic algorithms by interacting with students during program development (Areias, Mendes and Gomes, 2007). ProGuide is a product from the joint efforts of Cristiana Areias, Antonio Mendes and Anabela Gomes. It is developed partly based on inspiration obtained during the previous development of a flowchart tool, SICAS by their research group (Areias, Mendes and Gomes, 2007).

Development Environment

ProGuide adapts its environment to the imperative-procedural approach in its flowchart design, similar to Progranimate and RAPTOR. ProGuide creates the flowchart by selecting the desired flowchart symbols and clicking on the round grey circle found between flowchart symbols to add them.

ProGuide is divided into three main sections, namely the problem statement, the editor and simulator, and text based communication. The problem statement on

the top left corner of the window shows the problem for the exercise that the students are trying to solve after they selected an exercise from a range of customized questions in the tool. The editor and simulator on the right half of the window is a user-friendly iconic space that supports the design of algorithms using flowcharts (Areias, Mendes and Gomes, 2007). The text based communication on the left bottom corner of the window provides help, encouragement and guidance to students when they are designing algorithms by way of hints, examples and questions to trigger their reasoning skills (Areias, Mendes and Gomes, 2007). Figure 2.4 shows a screenshot of ProGuide.

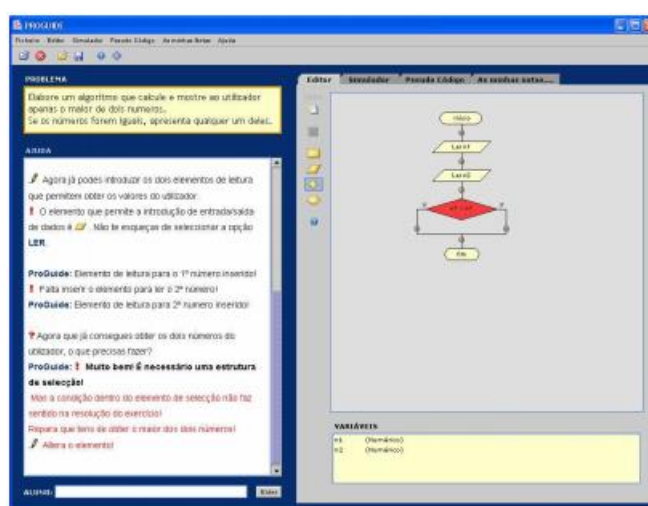


Figure 2.4: ProGuide Screenshot (Areias, Mendes and Gomes, 2007)

Features

ProGuide is preloaded with example problems and solutions that could be solved by students as a means of learning programming algorithms. During the entire phase of completing the examples chosen, students are guided by the communication integrated in ProGuide that could slowly develop problem solving habits in students. Whenever the editor and simulator detect any wrong design in the flowchart, the communication will notify students by providing warnings and hints.

Similar to Proanimate and RAPTOR, ProGuide offers the ability to animate flowchart execution. It highlights the currently executing flowchart by changing its background colour and supports tracking of value for the variables declared.

ProGuide also enables students to read information regarding similar problems or programming concepts by suggesting them. The built-in information includes texts and examples in flowcharts that could be simulated, allowing better understanding of the concept being explained (Areias, Mendes and Gomes, 2007).

Limitations

ProGuide does not emphasize on the concept of variable declaration. Variables are declared on the fly and data types for them are implicitly declared, similar to RAPTOR.

ProGuide could not generate source codes. This is a major setback as students could not easily relate between the flowchart symbols with sections of codes if they are looking at the flowchart's equivalent in codes for the same example. Next, ProGuide is not available for use freely.

Evaluation

There was no assessment or evaluation conducted on the effectiveness of ProGuide as a flowchart drawing tool.

2.2.4 Iconic Programmer

Iconic Programmer is an interactive tool that allows programs to be developed in the form of flowcharts through a graphical and menu-based interface (Chen and Morris, 2005, pp. 104-107).

Development Environment

Iconic Programmer employs the imperative-procedural programming techniques for its environment as it is targeted at novices. Iconic Programmer presents a single window with an initial flowchart having a start and end symbols connected by a line with a yellow square box in the middle between them. New flowchart symbols are added by clicking on the box and selecting the type of statement to be inserted, namely sequence, selection or repetition. After selection, users are presented with several menus to decide on the actions that the flowchart symbol will do. When flowchart symbols are created, they are not the common symbols used in flowcharts, and the texts that appear on the symbols are only the actions that they will do, not the entire statement itself. Figure 2.5 shows a screenshot of Iconic Programmer.

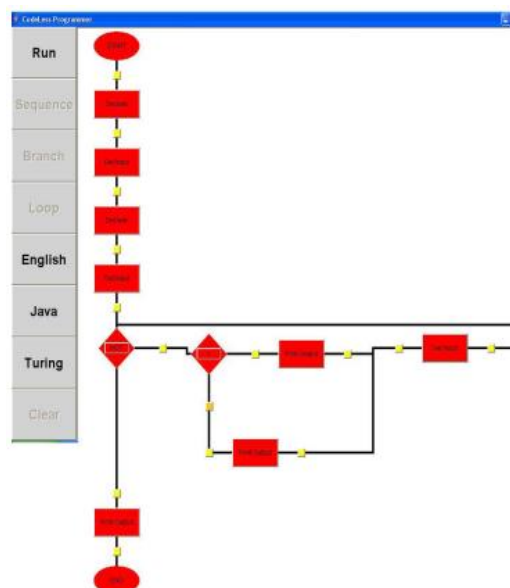


Figure 2.5: Iconic Programmer Screenshot (Chen and Morris, 2005, pp. 104-107)

Features

Iconic Programmer supports the ability to generate syntactically correct source code for the flowchart produced. The languages it supports include pseudocode, C/C++, Java and Turing.

Iconic Programmer offers the uniqueness of creating statements through the use of menus instead of requiring users to create whole statements themselves.

Iconic Programmer also allows program animation through flowchart execution in a step by step manner, similar to other tools, by highlighting the currently executing flowchart symbol. During the program animation, explanations in natural language are provided to explain on the variables' values.

Limitations

As Iconic Programmer is designed to allow students to focus on the design and development of algorithms, it does not provide syntax checking. Thus, whenever students produced syntax errors but could execute the statements, they may automatically assume that they actually created correct statements.

Flowcharts developed in Iconic Programmer use text instead of common symbols to indicate the actions of the symbols. This will cause complication in students when they are drawing flowcharts during exams or some other situations that use common symbols as they may be accustomed to the format of Iconic Programmer.

Only one data type for variables could be declared, which is the integer data type. Even if they are not explicitly declared, they are implicitly declared by having needed to store values that are of integer type.

Evaluation

Although Iconic Programmer was used by students in several courses in the past, there was no proper assessment or evaluation conducted on it.

2.2.5 FLINT

FLINT is a visual development environment that facilitates the construction of top-down design charts and the implementation and simulation of algorithms as flowcharts (Crews and Ziegler, 1998, pp. 307-312). It presents programming to students as activities surrounding design, implementation, testing and debugging (Crews and Ziegler, 1998, pp. 307-312). It was developed to address problems regarding syntax, problem solving and support for program execution in a unified manner (Crews and Ziegler, 1998, pp. 307-312).

Development Environment

FLINT employs the imperative-procedural approach and uses the concept of creating icons to develop flowcharts. This enables students to concentrate on pondering for ways to solve the problem by hiding low-level details from them (Crews and Ziegler, 1998, pp. 307-312).

FLINT is separated into two windows, a structure chart and the flowchart interface itself. The structure chart is a division of a problem into major steps, where each step will have its own flowchart that will be designed in the following flowchart interface. In order to be able to develop a flowchart for a process, FLINT requires the students to produce a structure chart, which will include the process. Then, in the flowchart interface, the flowchart for the process is designed. Both the structured chart and flowchart designs are created and modified by clicking on the desired

buttons that perform the different functions needed. Figures 2.6 and 2.7 show the screenshots of the structure chart and the flowchart interface of FLINT respectively.

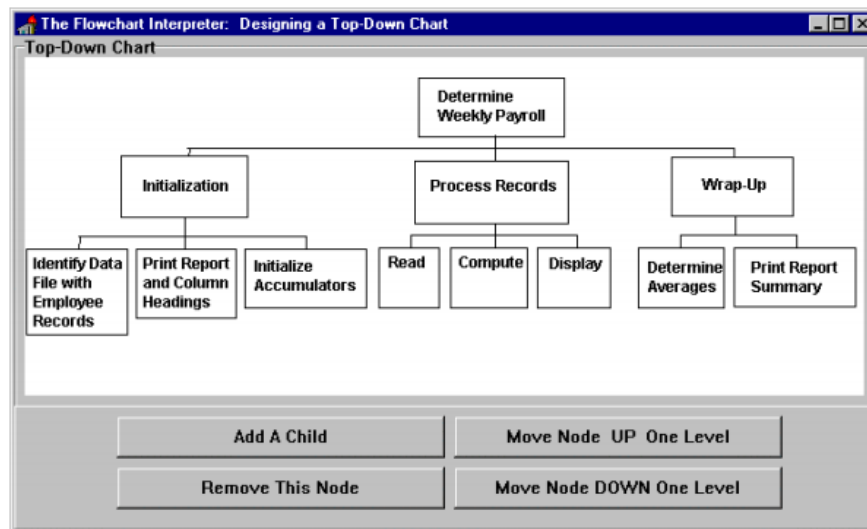


Figure 2.6: Top-Down Chart Design Screenshot (Crews and Ziegler, 1998, pp. 307-312)

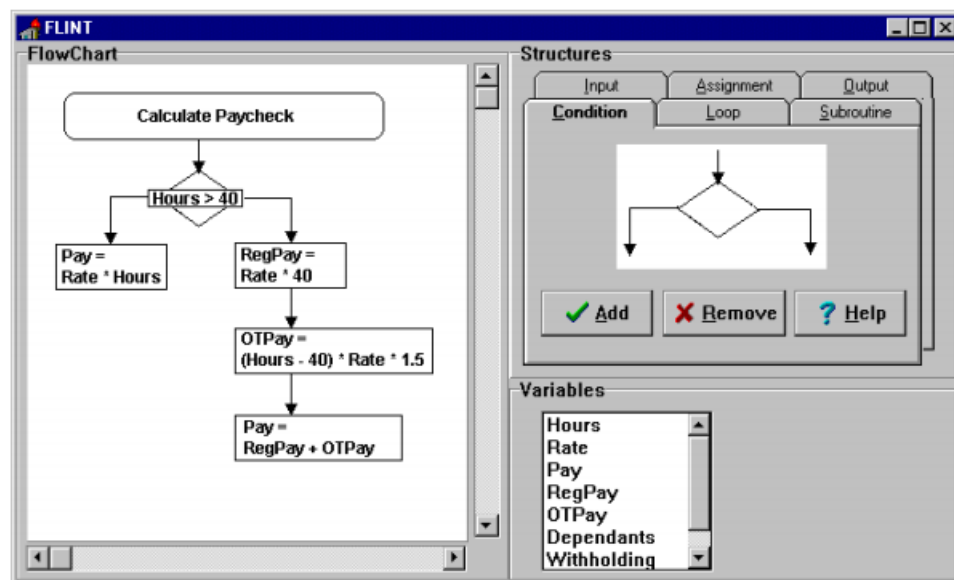


Figure 2.7: FLINT Screenshot (Crews and Ziegler, 1998, pp. 307-312)

Features

FLINT provides the ability to develop structure charts that provide the complete picture of the whole system before the development of flowcharts. This is useful as it emphasizes to students the importance of problem solving and design activities that they should thought of first before they initiate the implementation of the program (Crews and Ziegler, 1998, pp. 307-312).

FLINT allows program animation through manual flowchart execution, similar to the other more advanced tools. It highlights the currently executing flowchart and supports tracking of value for the variables declared in the program.

Limitations

FLINT is not freely available to use. Besides that, it does not provide the ability to generate syntactically correct source codes. Additionally, it does not support data type declaration for the variables that are created.

Evaluation

An experiment was carried out on the usefulness of structured flowcharts in programming. The results produced showed that they were beneficial to students in terms of reducing time needed to comprehend structured flowcharts, fewer errors, greater confidence and less time required to answer questions and preference of flowcharts as complexity increases (Xinogalos, 2013, pp.1313-1322).

2.2.6 SICAS

SICAS is an educational tool developed to assist in the learning of the basic concepts of programming (Mendes, et al., 2005, pp.193-197). It allows designing of algorithms through flowchart and execution of the produced algorithms by students. This tool is developed by Antonio Mendes, Anabela Gomes, Micaela Esteves, and Maria Marcelino from Portugal, as well as Crescencio Bravo and Miguel Redondo from Spain.

Development Environment

SICAS employs the imperative-procedural approach for its flowchart algorithm. It has two modes, a teacher mode and a student mode. In the former mode, teachers could introduce problems to be solved by students and possibly providing solutions to those problems, while in the latter mode, students will create their own algorithms to solve the questions given, and compare their answers with the solutions provided, if any (Marcelino, Mihaylov and Mendes, 2008, pp.T4A-7).

SICAS is separated into four major sections, namely the toolbar dock, the main content area, the variables and functions tabs, and the problem and console tabs. The toolbar dock located near the top of the window contains the buttons of structures utilised in the creation of the flowchart and the buttons for execution purposes. The main content area located in the middle displays the flowchart to be modified. The variables and functions tabs found on the middle right of the window displays the variables and functions declared that could be used by the algorithm. The problem and console tabs are located on the bottom, with the former showing the current problem required to be solved, and the latter used to receive inputs and display outputs. Figure 2.8 shows the screenshot of SICAS.

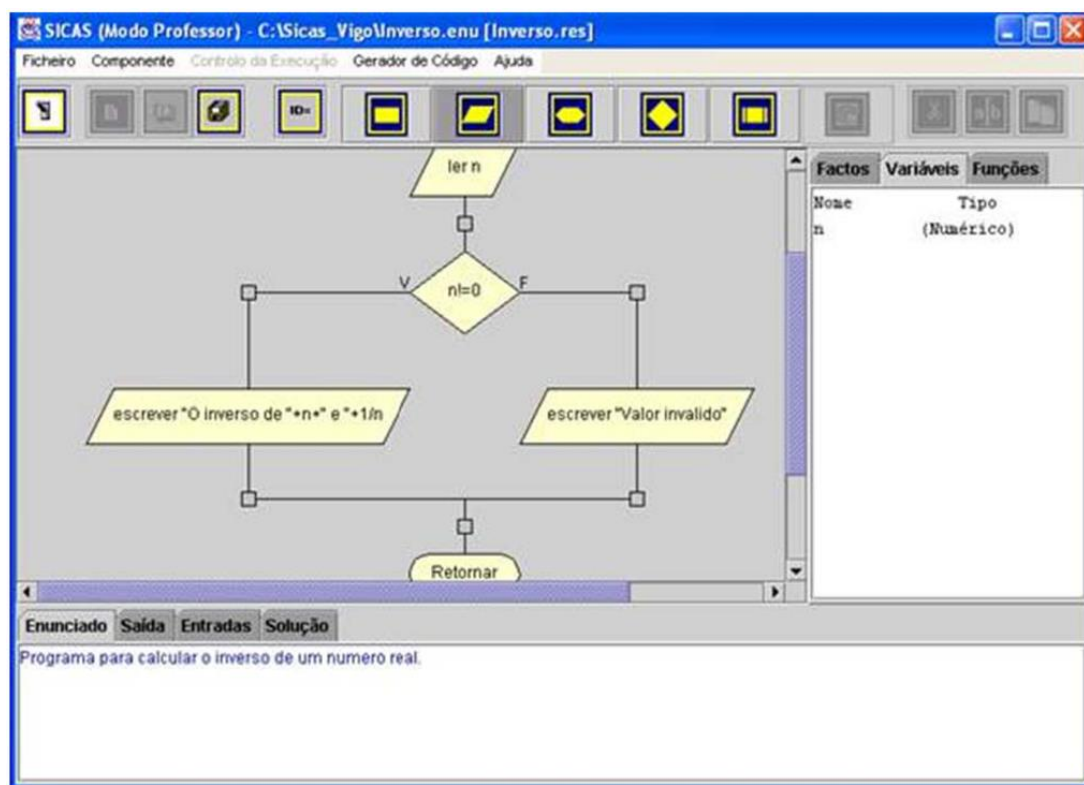


Figure 2.8: SICAS Screenshot (Gomes and Mendes, n.d.)

Features

SICAS supports algorithms for simple instructions, such as assignments, input or output, repetition and selection instructions (Marcelino, Mihaylov and Mendes, 2008, pp.T4A-7). Then, variables of numeric, string and array data types could be declared and are tracked under the Variables pane.

To construct the flowchart, flowchart elements could be added by clicking one of the elements in the toolbar dock and pointing at the flowchart (Marcelino, Mihaylov and Mendes, 2008, pp.T4A-7). Besides that, they could be removed, copied or modified at any point in time.

SICAS also provides the uniqueness of allowing its users to produce self-defined functions in order to introduce them to the concept of modularization (Marcelino, Anabela, Dimitrov and Mendes, 2004, pp8-6). Functions are also constructed using the same algorithms and there are also pre-defined functions for

number and string manipulation (Marcelino, Anabela, Dimitrov and Mendes, 2004, pp8-6).

SICAS automatically translates the produced flowchart algorithm into its equivalent in pseudocode, C or Java code. Besides that, simulation of the algorithm could be performed, with its users being able to control the simulation speed, to pause the simulation, and to return and repeat certain simulation events (Marcelino, Anabela, Dimitrov and Mendes, 2004, pp8-6).

Limitations

SICAS is not available to be used. Then, the symbol containing the decision condition in SICAS does not conform to the common flowchart symbols syntax as it uses a hexagon instead of a diamond.

Evaluation

SICAS was evaluated by both the programming lecturers as well as students, and the overall response is quite positive. Lecturers liked the graphical interface, the availability of functions, data type specifications for variable declarations and on top of these, the algorithm simulation is highly welcomed (Marcelino, Anabela, Dimitrov and Mendes, 2004, pp8-6).

2.2.7 B#

B# is an experimental iconic programming notation designed to provide initial technological support in the learning of introductory programming (Cilliers, Calitz and Greyling, 2005, pp.543-576). It provides an integrated visual environment in

order to enhance the learning experience of students in introductory programming courses (Cilliers, Calitz and Greyling, 2005, pp.543-576). It was developed by the Department of Computer Science and Information Systems at the Nelson Mandela Metropolitan University in South Africa as a response to the challenge of increasing throughput in introductory programming courses (Cilliers, Calitz and Greyling, 2005, pp.543-576).

Development Environment

B# employs the imperative-procedural technique in the construction of its flowchart. The flowchart being constructed uses a top-down single-sequence structure of icons connected by lines (Cilliers, Calitz and Greyling, 2005, pp.543-576).

B# consists of six sections, namely the development environment toolbox, debugging toolbox, icon palette, flowchart editing area, variable declaration and display area, and source code area. The development environment toolbox contains the actions needed to manage the application, the debugging toolbox stores the controls needed to animate the program, the icon palette contains the flowchart icons, the flowchart editing area displays the flowchart being designed, the variable declaration and display area keeps track of declared variables as well as displays the program animation outputs, and the source code area displays the corresponding source code of the constructed flowchart. Figure 2.9 shows the interface for B#.

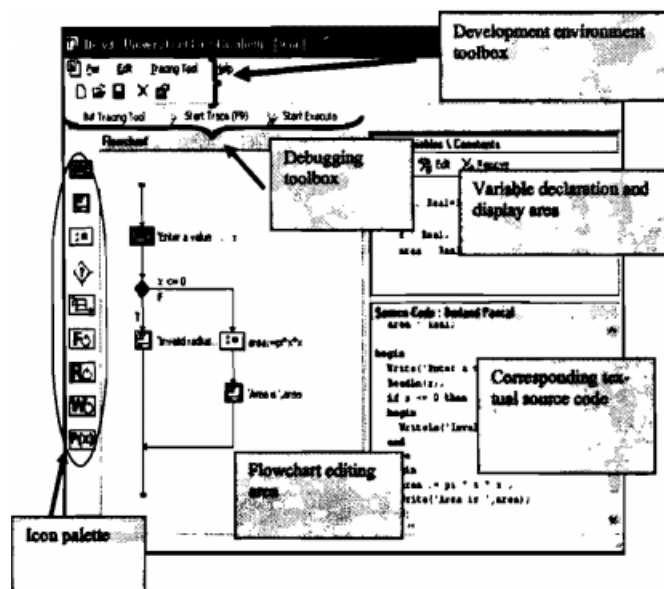


Figure 2.9: B# Screenshot (Cilliers, Calitz and Greyling, 2005, pp.543-576)

Features

B# uses uncommon flowchart icons for its flowchart. Then, it supports the basic fundamental algorithmic constructs of sequence, selection and iteration (Xinogalos, 2013, pp.1313-1322).

To construct the flowchart, flowchart icons are added by selecting one of the icons from the icon palette and pointing at the desired position in the flowchart. When the icons are added into the flowchart, a dialog box pops up for the configuration of the properties required by the added icon (Cilliers, Calitz and Greyling, 2005, pp.543-576).

B# provides the ability to animate the flowchart and also allows tracing through the execution. Then, syntactically correct Borland Pascal source code is generated alongside the constructed flowchart in the source code area.

Limitations

An empirical study was conducted in 2003 at Nelson Mandela Metropolitan University on 59 first year students, and it was discovered that the performance of the students is significantly improved with the inclusion of B# (Xinogalos, 2013, pp.1313-1322).

Evaluation

SICAS was evaluated by both the programming lecturers as well as students, and the overall response is quite positive. Lecturers liked the graphical interface, the availability of functions, data type specifications for variable declarations and on top of these, the algorithm simulation is highly welcomed (Marcelino, Anabela, Dimitrov and Mendes, 2004, pp8-6).

2.2.8 Comparison

Table 2.1 shows a comparison between the existing flowchart drawing tools analysed based on their available features.

Table 2.1: Comparison between Flowchart Drawing Tools

Tool	Progranimate	RAPTOR	ProGuide	Iconic Programmer	FLINT	SICAS	B#
Available freely	√	√		√			
Imperative-procedural technique	√	√	√	√	√	√	√
Object-Oriented technique		√					
Flowchart Design Method	Drag-and-drop flowchart, modify code	Drag-and-drop flowchart	Select icon and add	Context menu	Buttons and textboxes	Select icon and add	Select icon and add
Common flowchart symbols	√	√	√		√	√ (Except Decision)	
Program animation	√	√	√	√	√	√	√

Source code generation	√	√		√		√	√
Target OS Platform	Any (With Java support)	Windows	Windows	Windows	Windows	Windows	Windows
Evaluation	√	√			√	√	√
Special features	Synchronized flowchart and source code execution	Allows development of more source code generators, flowchart commenting	Uses a tutoring system model	Natural language explanation of flowcharts	Requires step-wise refinement of structured flowchart in problem solving	Has lecturer mode and student mode, and supports backward stepping	-

2.3 Proposed Features

In the proposed tool, it was conceptualized to contain the following significant features:

- Project Management (create, load and save flowchart projects)
- Flowchart Printing (print flowchart)
- Flowchart Designing (add and remove flowchart symbols)
- Variable Management (create, edit and delete variables with data types)
- Flowchart Execution (execute flowchart to animate program)
- Source Code Generation (automatically generate syntactically correct source code from designed flowchart)
- Action Reversing (undo and redo added and removed flowchart symbols)

2.4 Existing Development Methodologies

During the development of software products, one or more suitable software methodologies are implemented in order to maximize development efficiency. The development methodologies to be discussed are considered better than the traditional waterfall methodology.

2.4.1 Agile Development (Feature Driven Development)

Agile development is a methodology that emphasizes on flexibility, proper planning, early prototyping and continuous improvement of the prototypes produced. With this methodology, software is developed in small, incremental cycles, whereby after each cycle, prototypes with extra functionality are added until a complete product is produced. As there are several processes under agile development, the specific process that will be focused on is Feature Driven Development (FDD).

FDD is a process that emphasizes on the progression of project development through the features of the system to be developed. Features are the source of requirements for FDD, similar to the way use cases and storyboards are to RUP and Scrum respectively (Ambler, 2014). Figure 2.8 shows the five main activities of FDD that are performed iteratively.

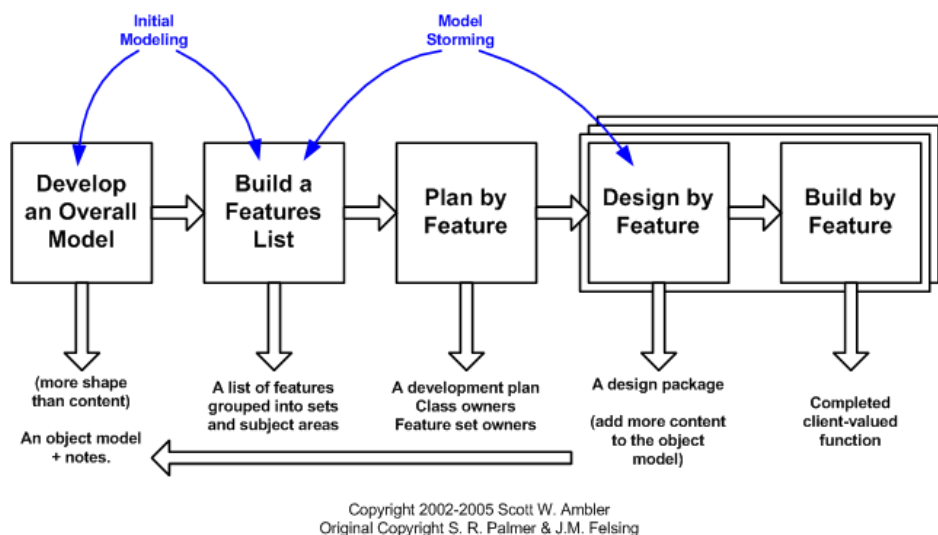


Figure 2.10: FDD Project Lifecycle (Ambler, 2014)

2.4.1.1 Develop an Overall Model

This process involves constructing an object model that highlights the domain problem. Small groups are used to create detailed domain models, which are then presented for peer reviews in order to obtain either one or a combination of proposed

models for use in each area of the domain (Lawton, 2015). The overall model is then produced when these models merged over time.

2.4.1.2 Build a Features List

Using knowledge learnt during the modelling process, domains are sliced into subject areas that contain information on business activities to establish a features list (Lawton, 2015). A categorised list of features is represented by the steps used for each business activity. Then, all the features are expressed in the form of “<action> <result> <object>”. The features are expected to be completed within two weeks, but in the event that any feature took longer than that duration, it is broken down in smaller sections (Lawton, 2015).

2.4.1.3 Plan by Feature

Once the features list is properly established, programmers are assigned to handle various feature sets (Lawton, 2015). This is done by way of creating the development plan for the features with the assigned programmers.

2.4.1.4 Design by Feature

A design package for each feature is then produced and a chief programmer will then select a group of features that should be developed within the available timeframe (Lawton, 2015). Additionally, detailed diagrams for each feature are also created by the chief programmer when the model refinement is underway (Lawton, 2015). After that, prologues are produced and design inspection is conducted (Lawton, 2015).

2.4.1.5 Build by Feature

When design inspections are over, designers plan the activity for each feature and develop the code for each respective class (Lawton, 2015). Finally, the completed feature is combined with the main build after a successful code inspection and unit test (Lawton, 2015).

Advantages

Agile development promotes higher customer satisfaction due to more frequent and continuous delivery of functional software prototype with incremental additions of functionality. Then, high interaction between customers and developers improves customer confidence, as customers are aware of the progress of the software.

Minor changes in the requirements during development are acceptable as it is easy to adapt the prototype to the new requirements. Additionally, testing is conducted during every iteration of the project life cycle, minimizing the likelihood of bugs in the software.

Disadvantages

For some software deliverables, especially large-scale deliverables, it is difficult to evaluate the amount of effort needed at the start of the SDLC (ISTQB Exam Certification, n.d.). Then, there is a lack of design and documentation during the commencement of the project.

The deliverables for the project may not be accurate as there is a possibility that the customer representative is unsure about the actual outcome expected from the software (ISTQB Exam Certification, n.d.a). Additionally, Agile is unsuitable to

be executed by junior programmers as they lack the skill and experience for the decision-making process required during development.

2.4.2 Iterative and Incremental Development

Iterative and incremental development is a methodology that is focused on the gradual increment of software features along with a repetitive release and improvement cycle. It is involved in the development and integration of various software features at various times as well as the revision of the features in order to improve on them.

Advantages

Firstly, the end products developed are highly representative of the expected products required by clients as demonstration and adjustment of the functionalities of the products are conducted during every iteration of the project. Next, the complexities and risks of the projects are contained to enable the development team to continuously review and adapt the projects to the changing requirements. The risk of any delays in the project is minimized as core tasks of the project are completed first.

Disadvantages

The implementation of this methodology requires developers with excellent technical expertise and discipline in order to be able to conduct it effectively. Then, there will be a high maintenance cost as errors unknown since the beginning of the project life cycle will be improved through maintenance (UK Essays, 2013).

2.4.3 Rapid Application Development (RAD)

RAD is a methodology where components of software are developed in parallel as if they are mini-projects (ISTQB, n.d.b). Working prototypes are swiftly produced in order to gain feedback regarding the accuracy of the features in the prototypes with the requirements.

Advantages

Firstly, RAD reduces project risk as there is a high involvement of client and feedback during every iteration of the project cycle, increasing customer confidence and product acceptance. Then, better quality software is produced, whereby it is more usable and focuses more on the business issues faced by customers instead of technical issues faced by developers.

Disadvantages

For low-cost and small-scale projects, implementing RAD is inefficient as the cost of modelling and automated code generation is very high (ISTQB, n.d.b). Next, RAD requires highly skilled developers and designers. Without them, the design and quality of the software produced will be very poor.

2.4.4 Spiral Development

Spiral development combines the concept of iterative development and the emphasis on risk analysis. It is mainly used in large-scale and expensive projects as these projects are of high risk.

Advantages

Spiral development helps in providing better risk management as risky components could be developed earlier in the project cycle, thus minimizing project risk. Apart from that, new functionalities could be added later during the project cycle without conflicting with the original requirements.

Disadvantages

The risk analysis conducted in this methodology requires extremely expert individuals. Additionally, the projects' successes are highly dependent on the risk analysis phase, which puts more emphasis on the need to have experts conducting the risk analysis. Then, it is possible that the spiral may continue indefinitely due to the lack of good change discipline.

2.5 Existing Software Platforms

There are currently several potential platforms that the flowchart drawing tool could be developed with and deployed on. Each platform has its own advantages and disadvantages, and it is imperative that the most suitable platform is selected in order to ensure the success in the completion and delivering of this project.

2.5.1 Windows Forms (WinForms)

WinForms is a GUI class library in the Microsoft .NET framework. It enables developers to create rich client-based desktop applications in the Windows OS. WinForms is event-driven, whereby it is idle during most of its lifetime while waiting for user input.

Development Environment

Development using WinForms primarily involves the following areas, the design area of a form class, the code-behind of the design area, and the designer of the form class. The design area is the place where user interface controls from the toolbox are added directly into the form shown, allowing developers to view the layout of the controls on the form during design time instead of during runtime. The attributes and events of the each control could be configured through the “Properties” window when the control is currently being selected in the design area. The code-behind of the design area handles the logic for all the controls in the form and is primarily the location for developers to create development codes. The methods for handling events of controls will appear here, whereby developers will need to write their own codes to handle the event raised. The designer is the place that stores the configuration codes for the controls in a form and links the events of controls to their respective methods. The codes here are automatically generated every time a configuration is made in the “Properties” window for any controls. Figures 2.11, 2.12 and 2.13 show the design area, code-behind and designer screenshots respectively.

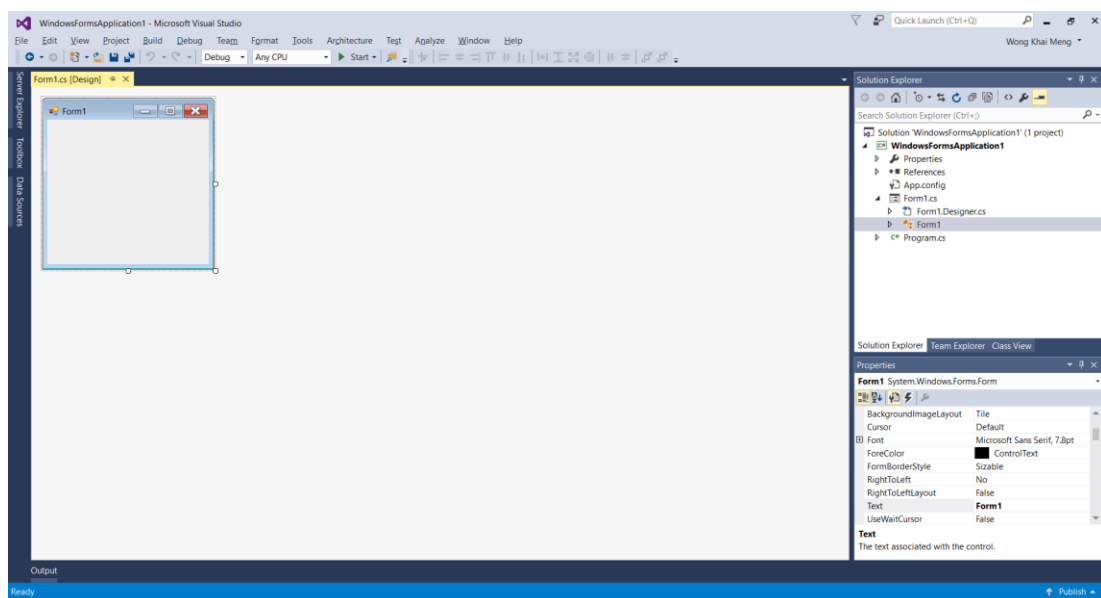


Figure 2.11: Design Area Screenshot

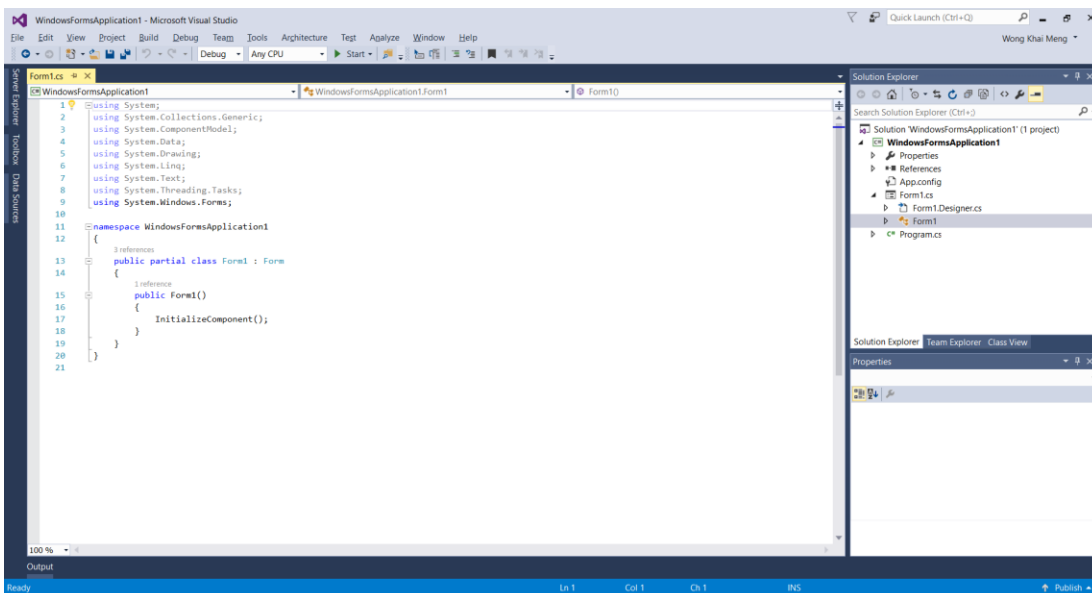


Figure 2.12: Code-behind Screenshot

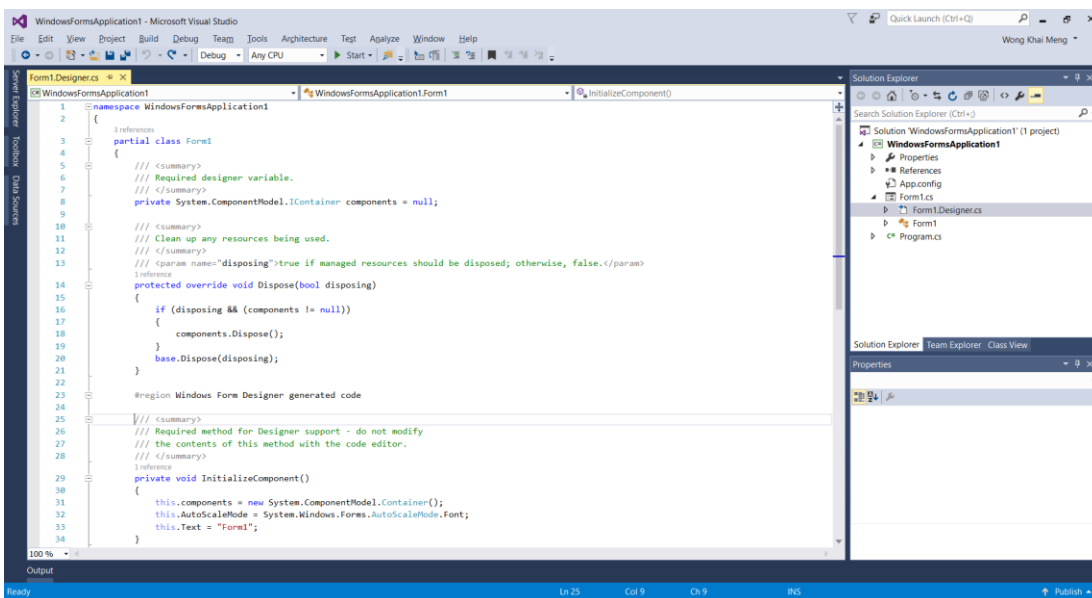


Figure 2.13: Designer Screenshot

Advantages

WinForms provides minimal functionality of user interface elements and events. Additionally, as WinForms has existed for a long period of time, it contains many libraries that support a diverse amount of functionalities that may be required by

developers. Then, docking support is also available that allows controls to automatically rearrange when placed in a parent. Next, impaired users could utilise WinForms more easily, thanks to the support of Microsoft Active Accessibility.

Limitations

WinForms is currently under maintenance mode, whereby it will receive no extra features, but will continue to obtain bug fixes from Microsoft. Then, WinForms controls are drawn using GDI+, which may cause software to behave strangely or become unresponsive when the maximum limit of GDI objects is achieved.

2.5.2 Windows Presentation Foundation (WPF)

WPF is a flexible subsystem designed for rendering Windows-based desktop applications with rich user experience by acting as a GUI framework. WPF utilises Microsoft DirectX instead of GDI, which is an older graphics rendering API.

Development Environment

Development in WPF primarily involves the eXtensible Application Markup Language, XAML design area and the code-behind for it. XAML is a declarative markup language based on XML used to develop application user interfaces, and the XAML design area is a place where controls could be added to the user interface either through writing XAML codes or by dragging and dropping controls from the toolbox onto the user interface display available as part of the design area. The attributes for the controls could be configured either through the writing XAML codes, by setting the attributes in the “Properties” window similar to WinForms, or through the user interface display, which could only set certain attributes. However,

configuring attributes through the “Properties” window or the user interface display will always automatically generate related XAML codes into the XAML design area. Any modifications made in any one of these 3 areas will immediately update the user interface display.

The code-behind for XAML will contain the code logic for the events of all controls created as well as any other development logic produced by the developer. Figures 2.14 and 2.15 display the XAML design area and code-behind screenshots respectively.

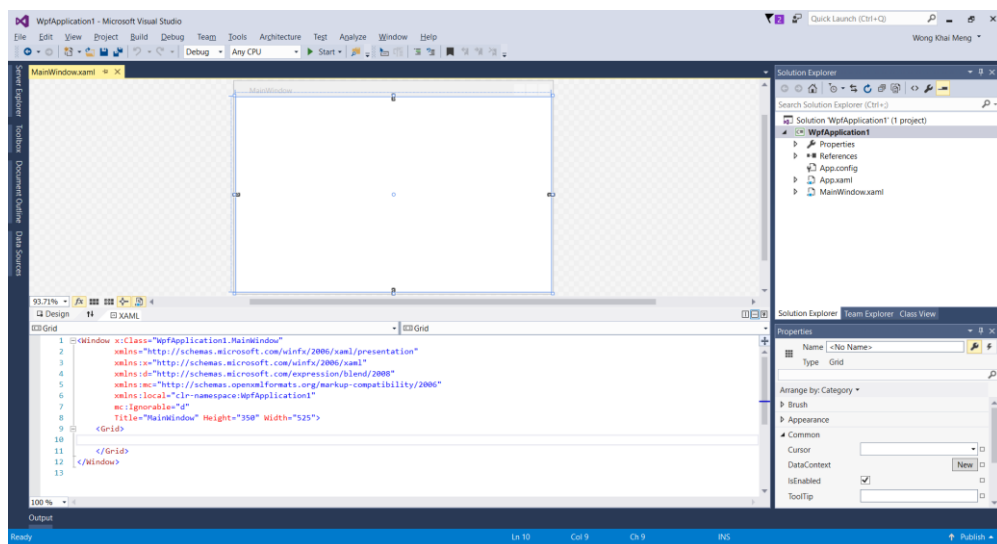


Figure 2.14: XAML Design Area Screenshot

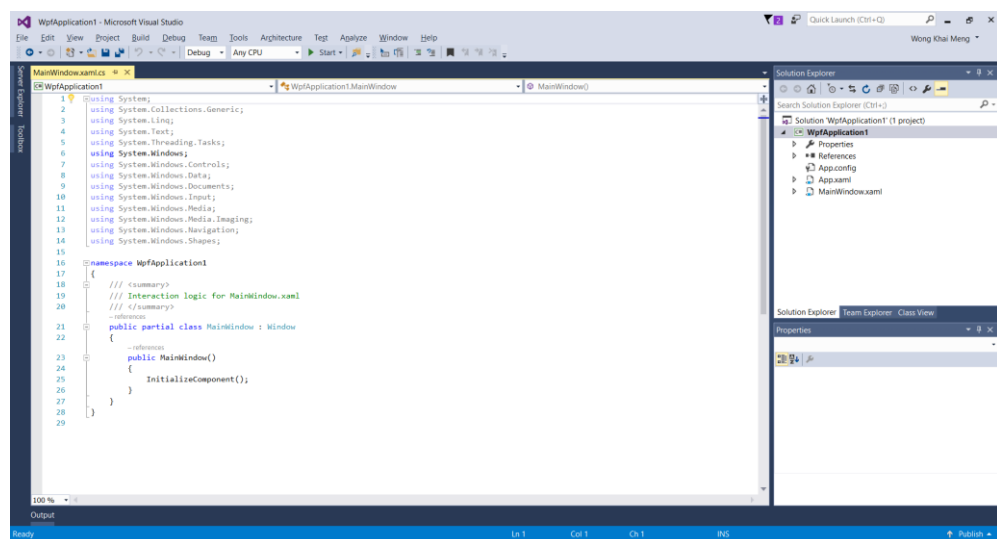


Figure 2.15: Code-behind Screenshot

Advantages

The usage of GPU in graphics rendering reduces CPU workload and speeds up screen refreshes. Next, the introduction of XAML allows separation of GUI from business logic, which resembles the good practices of a model-view-controller (MVC) pattern.

Type conversion in XAML, a method that converts a line of string to a target type, allows values of controls' properties to accept a line of string in a specific format unique to each property instead of creating the same values for the properties using multiple lines of XAML codes, making the XAML codes more compact and streamlined.

Customized controls called "User Controls" could be created from scratch using XAML for reuse in the application without the need to purchase new controls. Besides that, the availability to create templates for controls and reuse them across multiple controls also promotes reusability and reduces duplicated codes.

Limitations

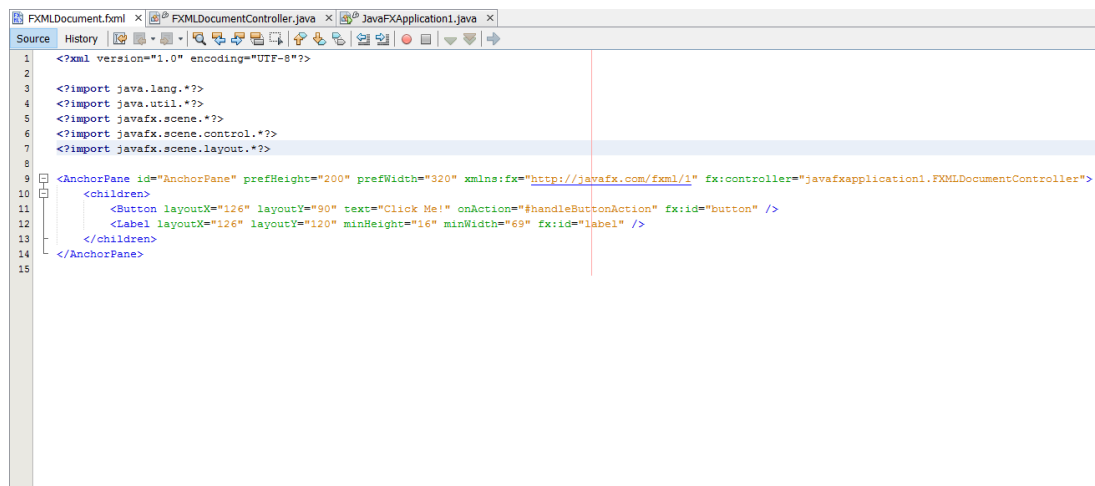
WPF requires huge amount of design work done manually by developers. Besides that, any additional dependency properties required for "User Controls" need to be manually registered and this may lead to a bloated amount of codes when many additional dependency properties are added.

2.5.3 JavaFX

JavaFX is a software platform for developing desktop applications and web applications. It is designed to replace Swing, a GUI widget toolkit for Java, in providing interactive ways for developers to design GUIs.

Development Environment

Development in JavaFX involves typically 3 types of files, the FXML Document, the FXML Document Controller and the JavaFX Application. The FXML Document is the place where the user interface of the application is created. Controls are created and added to the application by typing in the related control's codes with its attributes. The FXML Document Controller is the code-behind for the FXML Document and it handles the implementation logic for the controls. The JavaFX Application acts as the starting point for the execution of the application, whereby it creates and displays the user interface for the application. Figures 2.16, 2.17 and 2.18 show the FXML Document, FXML Document Controller and JavaFX Application Screenshot respectively.



```
FXMLDocument.fxml x FXMLDocumentController.java x JavaFXApplication1.java x
Source History
1 <?xml version="1.0" encoding="UTF-8"?>
2
3 <?import java.lang.*?>
4 <?import java.util.*?>
5 <?import javaFX.scene.*?>
6 <?import javaFX.scene.control.*?>
7 <?import javaFX.scene.layout.*?>
8
9 <AnchorPane id="AnchorPane" prefHeight="200" prefWidth="320" xmlns:fx="http://javafx.com/fxml/1" fx:controller="javafxapplication1.FXMLDocumentController">
10   <children>
11     <Button layoutX="126" layoutY="90" text="Click Me!" onAction="#handleButtonAction" fx:id="button" />
12     <Label layoutX="126" layoutY="120" minHeight="16" minWidth="69" fx:id="label" />
13   </children>
14 </AnchorPane>
15
```

Figure 2.16: FXML Document Screenshot

```

11 import javafx.fxml.FXML;
12 import javafx.fxml.Initializable;
13 import javafx.scene.control.Label;
14
15 /**
16  *
17  * @author KhaiMeng
18  */
19 public class FXMLDocumentController implements Initializable {
20
21     @FXML
22     private Label label;
23
24     @FXML
25     private void handleButtonAction(ActionEvent event) {
26         System.out.println("You clicked me!");
27         label.setText("Hello World!");
28     }
29
30     @Override
31     public void initialize(URL url, ResourceBundle rb) {
32         // TODO
33     }
34
35 }
36

```

Figure 2.17: FXML Document Controller Screenshot

```

18 public class JavaFXApplication1 extends Application {
19
20     @Override
21     public void start(Stage stage) throws Exception {
22         Parent root = FXMLLoader.load(getClass().getResource("FXMLDocument.fxml"));
23
24         Scene scene = new Scene(root);
25
26         stage.setScene(scene);
27         stage.show();
28     }
29
30     /**
31     * @param args the command line arguments
32     */
33     public static void main(String[] args) {
34         launch(args);
35     }
36
37 }
38

```

Figure 2.18: JavaFX Application Screenshot

Advantages

Applications developed using JavaFX is platform-independent as the JavaFX uses native code. Thus, any platform that supports Java could run applications developed in JavaFX. Next, JavaFX has CSS support, and HTML as well as JavaScript could be integrated into JavaFX applications.

Limitations

The client computers running JavaFX applications require the installation of JRE. Next, JavaFX is still a maturing technology, as it lacks a lot of features available in other platforms and frameworks. Then, the designing of the user interface only displays the structure of the interface in codes. In order to design the user interface visually, another application called JavaFX Scene Builder needs to be installed. This application will automatically generate FXML codes for the controls created visually in an FXML file, which could be used with a JavaFX project. Figure 2.19 shows the screenshot for the JavaFX Scene Builder

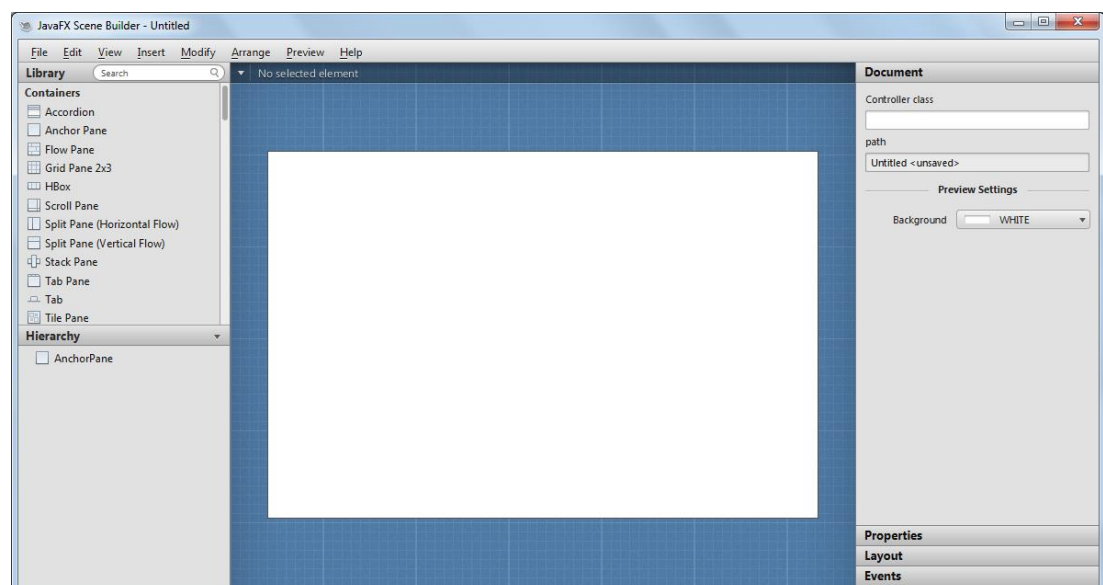


Figure 2.19: JavaFX Scene Builder Screenshot

2.6 Conclusion

Many existing tools that were produced were due to the realization that the current methodology of learning for novice programmers is difficult, time-consuming and contains irrelevant information that is inappropriate at the current level of learning. Based on the analysis conducted, the best candidates currently available to be utilised to teach basic programming to novices are RAPTOR and Proanimate. Thus, the features to be included flowchart drawing tool to be developed will mainly reference the features available in these two programming environments.

CHAPTER 3

PROJECT METHODOLOGY AND PLAN

3.1 Implementation of Chosen Development Methodology

For this project, the chosen development methodology was Agile, and the process selected to be utilised was Feature Driven Development (FDD).

3.1.1 Develop an Overall Model

Class diagrams of the tool to be developed were produced to represent the models and the relationships between the models of the system. Moreover, attributes and operations were also included along with the class diagrams.

3.1.2 Build a Features List

A list of features for the tool was created to provide a general view of all the available actions that could be performed by users when they are using the tool.

3.1.3 Plan by Feature

A development plan based on the features listed in the features list was produced. The features were planned for development in a sequential manner, but some features may overlap and start before the development on the current feature is completed.

3.1.4 Design by Feature

When every feature becomes active, a sequence diagram will be produced for each of them. Besides that, the design for the features and the object model may be refined and updated over time due to small changes in the tool.

3.1.5 Build by Feature

Each feature is coded and implemented into the tool. By the end of this phase, a tool with the newly working feature is completed.

3.2 Implementation of Chosen Software Platform

For this project, the selected software platform for the development of the flowchart tool is WPF, and the development environment to be utilised is Microsoft Visual Studio.

Throughout the entire development, the MVC architectural pattern will be implemented to segregate the model classes, business logics and user interfaces that will be created. Then, the availability of user controls in WPF enables flowchart symbols to be created as custom controls that could be dynamically created easily.

3.3 Project Plan

Figures 3.1 and 3.2 display the entire project plan.

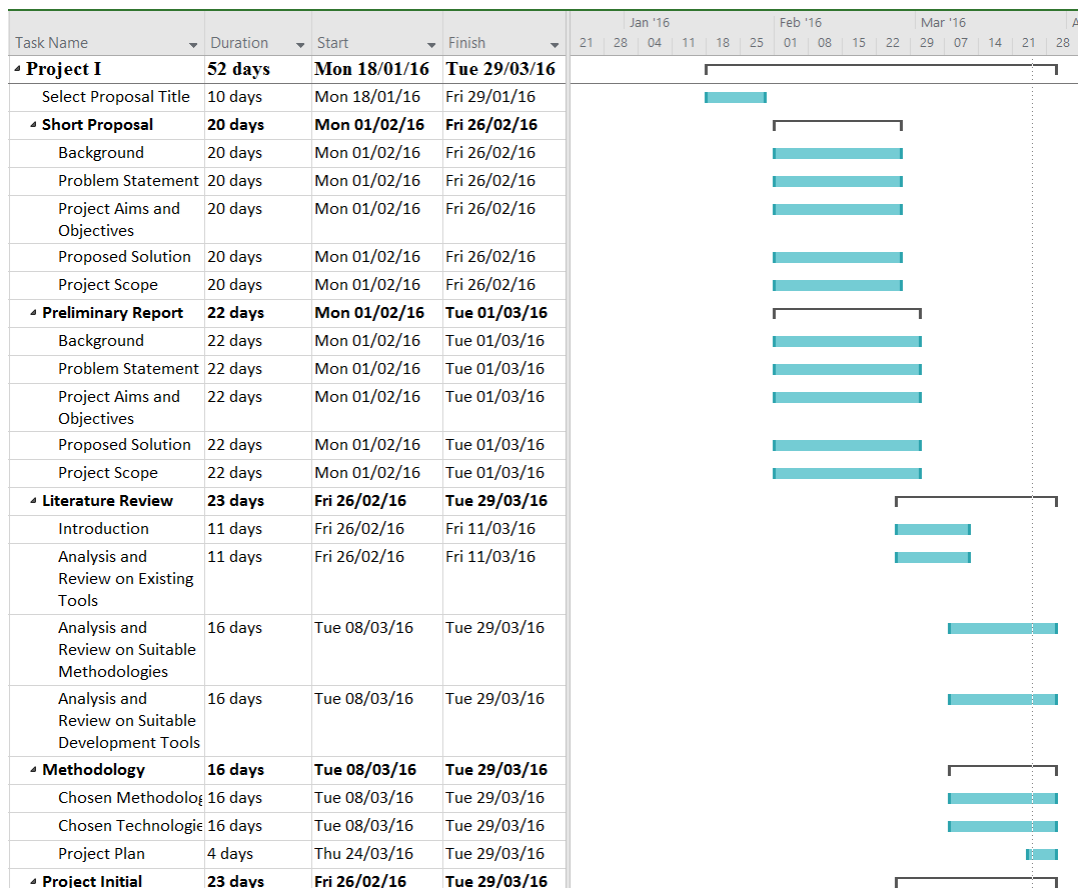


Figure 3.1: Project Plan (a)

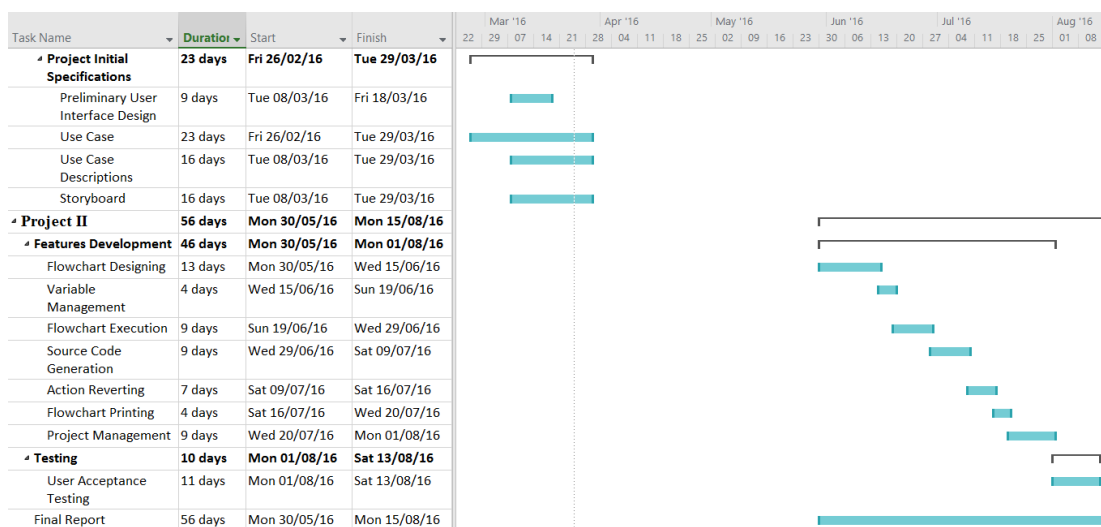


Figure 3.2: Project Plan (b)

In Project 1 shown in Figure 3.1 and partly shown in Figure 3.2, the requirements, analysis and specifications of the project were planned. Then, in Project 2, the project plan included the development of the proposed solution in phases based on features, the types of testing involved and the production of the final project report.

3.4 Conclusion

FDD is an appropriate methodology for the progression of the development of this project. The separation of features enables incremental additions be made to a functional prototype and reduces the scope when detecting errors.

WPF is the suitable platform for development mainly because of its flexibility over other platforms, as it allows easy designing of user interfaces and creations of “User Controls” by letting developers have more control during development.

CHAPTER 4

PROJECT SPECIFICATIONS

4.1 Fact Finding

4.1.1 Interview

Email interview was conducted with 3 lecturers who taught students learning the fundamentals of programming. The lecturers were provided questions to gauge their knowledge about the current state of learning of students and the existence of flowchart drawing tools, as well as their opinions on the usefulness of the proposed flowchart drawing tool based on its user interface. The interview questions are shown in Appendix A, and the interview results are shown in Appendix B.

Based on the interview conducted, several conclusions could be drawn. Firstly, lecturers expected students to create flowcharts by hand, and they do face some difficulties when drawing flowcharts and writing codes. Then, among several basic programming concepts, variables and data types were considered easy to students, while control structures and functions were deemed difficult to be comprehended. In the context of flowchart and its symbols, they were perceived to be neither hard nor easy for understanding.

2 of the 3 interviewees did know of the existence of flowchart drawing tools that could be suitable for use by students during their studies, yet they feel that those tools could be improved on. Then, for the visual evaluation of the proposed tool's user interface, it was suggested to be informative and directional as new users of the

tool did not know the location to begin to learn to use the tool. However, all interviewees held hopes that the proposed tool may be able to improve the quality of learning among students.

4.1.2 Survey

The survey was utilised to gather information from the students about their learning process on the fundamentals of programming. A total of 9 students completed the survey. The survey questions are found in Appendix C, and the replies are found in Appendix D.

Based on the analysis conducted on the survey replies, several conclusions could be drawn. Firstly, 55.6% or 5 students found it to be easy to learn programming, while 3 students found it to be difficult and the rest neutral. Then, in contrast with the perspective of lecturers, more than half, or 67% of students did use applications to create flowcharts, with only the rest of them drew by hand. When asked about the difficulty of constructing flowcharts using the selected methods, they averaged a neutral rating. The problems commonly faced when using those methods were mainly them being too time-consuming and for tools, they also involved confusing interfaces.

When asked about the improvements that they hoped to see in the tools chosen, the main priority was on having a more user-friendly interface, with the rest going to automatic source code generation. Additionally, among the programming basics that the participants learnt, the decision control structure was surprisingly rated the easiest followed by data types, with the rest revolving around the neutral mark. Lastly, when asked about the importance of having certain features in a possible future flowchart drawing tool, the participants rated high importance on almost all of them, with the exception of variables and data type declaration, whereby it obtained a more neutral score than the others.

4.2 User Requirements

The user requirements of the proposed tool are separated into functional and non-functional requirements. The term “user” refers to the user of the proposed tool, and the term “system” refers to the proposed tool.

4.2.1 Functional Requirements

1. User shall be able to create project.
2. User shall be able to load project.
3. User shall be able to save project.
4. User shall be able to execute flowchart.
5. User shall be able to print flowchart.
6. User shall be able to add symbol.
7. User shall be able to edit symbol.
8. User shall be able to remove symbol.
9. User shall be able to generate source code.
10. User shall be able to add variable.
11. User shall be able to edit variable.
12. User shall be able to delete variable.
13. User shall be able to undo action.
14. User shall be able to redo action.

4.2.2 Non-Functional Requirements

1. The tool shall be able to execute in Windows XP and above with no compatibility issue.
2. The tool shall be able to execute with .NET 4.5.2 and above with no compatibility issue.
3. The tool shall validate user input for expressions.

4. The tool shall validate user input for statements.
5. The tool shall validate user input for variables.
6. The tool shall prevent user from undoing removed Decision and Loop symbols.
7. The tool shall prevent user from executing flowchart with insufficient inputs.
8. The tool shall prevent user from loading incorrect project file.

4.3 User Interface Design

The user interface design of the proposed tool is separated into several sections, namely the menu bar, toolbox, variable inspector, console and the flowchart designer. Figure 4.1 shows the user interface design of the tool.

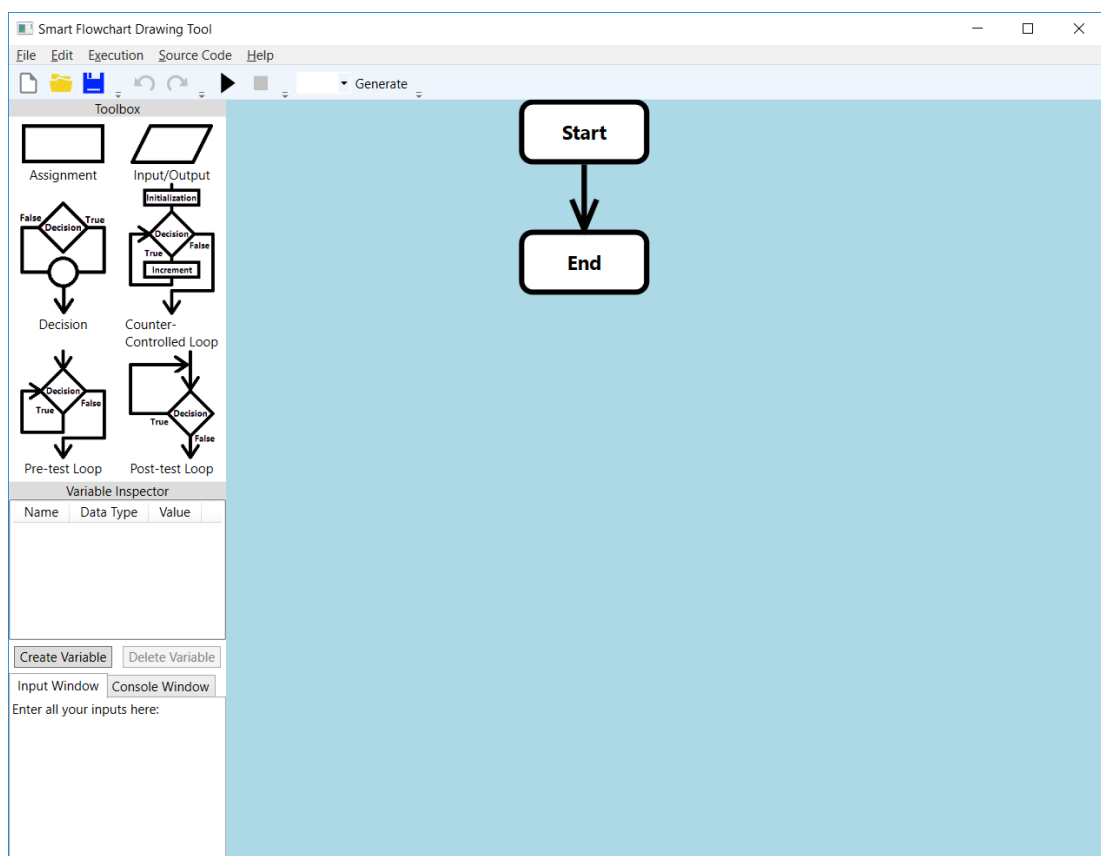


Figure 4.1: User Interface Design

4.3.1 Menu Bar

The menu bar consists mainly of actions and shortcuts that are important to the utility of the flowchart drawing tool. It supports actions such as the standard create, load and save project, print flowchart, undo and redo actions, flowchart execution, source code generation and the user guide. Figure 4.2 shows the menu bar.

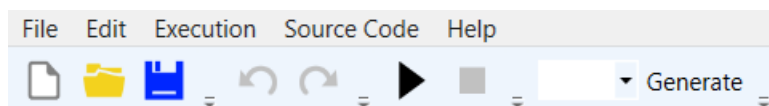


Figure 4.2: Menu Bar

4.3.2 Toolbox

The toolbox stores the various types of flowchart symbols required during the design of flowcharts. The symbols available include assignment, input or output, decision, counter-controlled, pre-test and post-test loops. Figure 4.3 displays the toolbox.

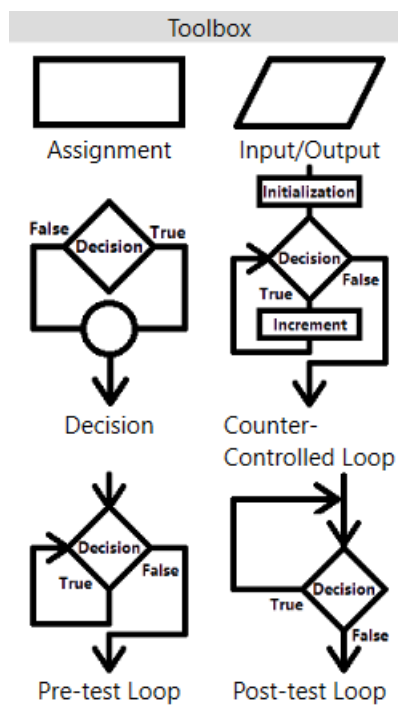


Figure 4.3: Toolbox

4.3.3 Variable Inspector

The variable inspector keeps track on the variables declared and used in the designed flowchart. Each value of the declared variables will only show the latest assigned value from the executed flowchart. Most importantly, the variable inspector supports the creation of variables when needed and the removal of variables when unused. Figure 4.4 shows the variable inspector.

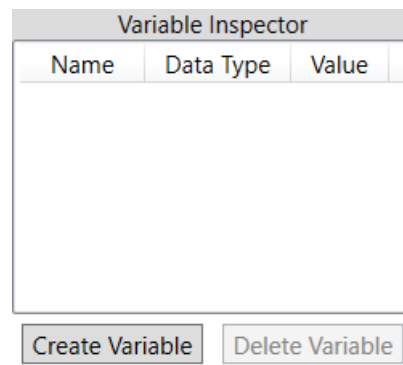


Figure 4.4: Variable Inspector

4.3.4 Console

The console contains two tabs, the Input Window tab and the Console Window tab. The former is used to receive all the inputs required for the execution of flowcharts, while the latter displays the outputs from the execution of flowcharts. Figure 4.5 displays the console.

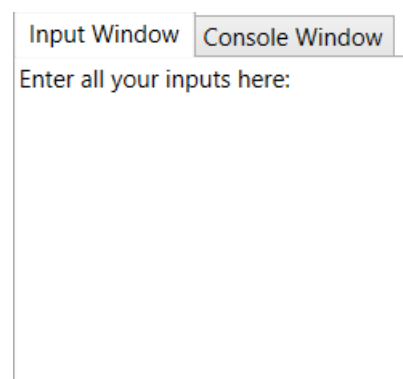


Figure 4.5: Console

4.3.5 Flowchart Designer

The flowchart designer contains the flowchart that is currently being constructed. Initially, the flowchart will consist of a start symbol and an end symbol connected with an arrow. Figure 4.6 shows the flowchart designer.

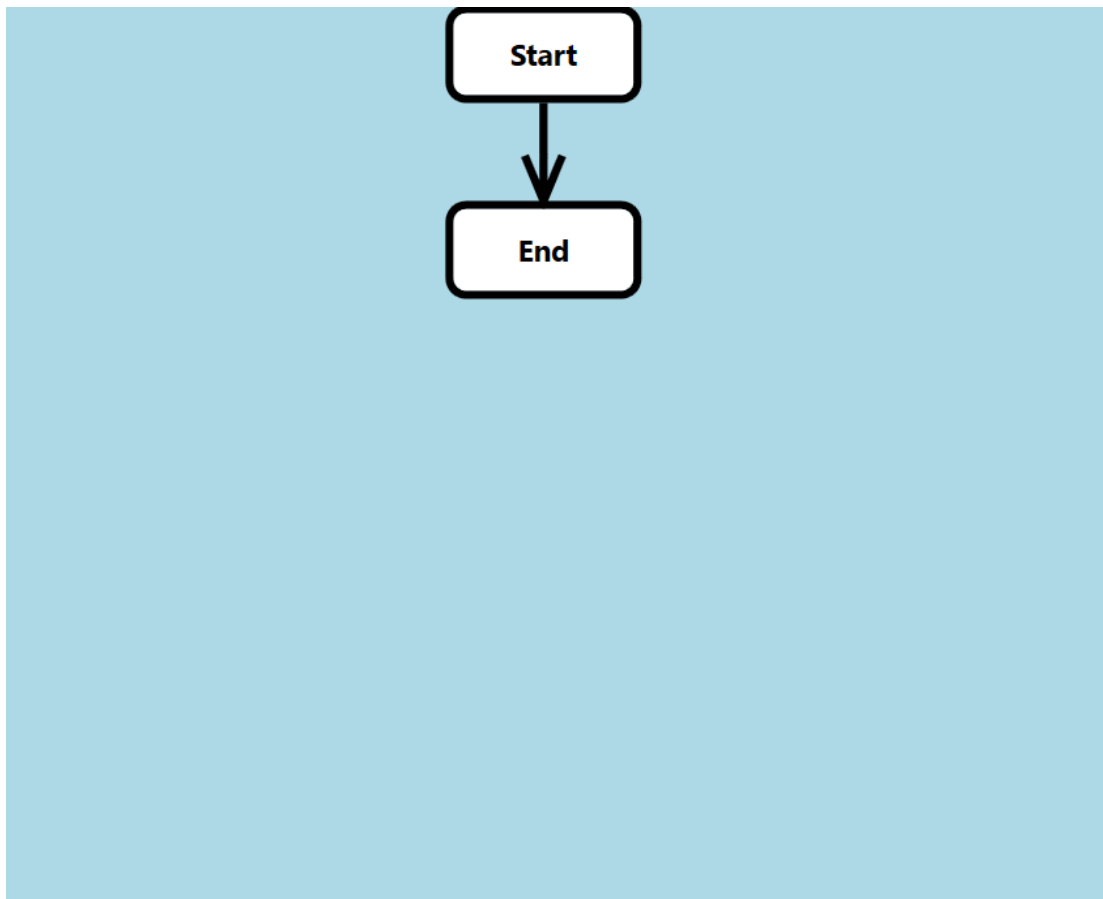


Figure 4.6: Flowchart Designer

4.4 Storyboard

4.4.1 Scenario 1: Draw a Flowchart

This scenario was triggered when a user wants to draw a flowchart. For example, a user wants to design a flowchart to display the value of $1 + 2$. The storyboard for this scenario could be found at Appendix H.1.

4.4.2 Scenario 2: Create, Edit and Display Variable

This scenario was triggered when a user wants to create, edit and display a variable in the flowchart. For example, a user wants to design a flowchart to create a variable named “a”, to store the result of $1 + 2$ into “a” and to display the value of “a”. The storyboard for this scenario could be found at Appendix H.2.

4.4.3 Scenario 3: Execute Flowchart

This scenario was triggered when a user wants to execute a flowchart. For example, the user wants to execute the flowchart created from Scenario 2 to view the results. The storyboard for this scenario could be found at Appendix H.3.

4.4.4 Scenario 4: Generate Source Code

This scenario was triggered when a user wants to generate the C++ source code of a flowchart. For example, the user wants to generate the source code of the flowchart created from Scenario 2. The storyboard for this scenario could be found at Appendix H.4.

4.5 Use Case Modelling

4.5.1 Use Case Diagram

A use case diagram for the flowchart drawing tool was created. Figure 4.7 depicts the use case diagram for the proposed tool.

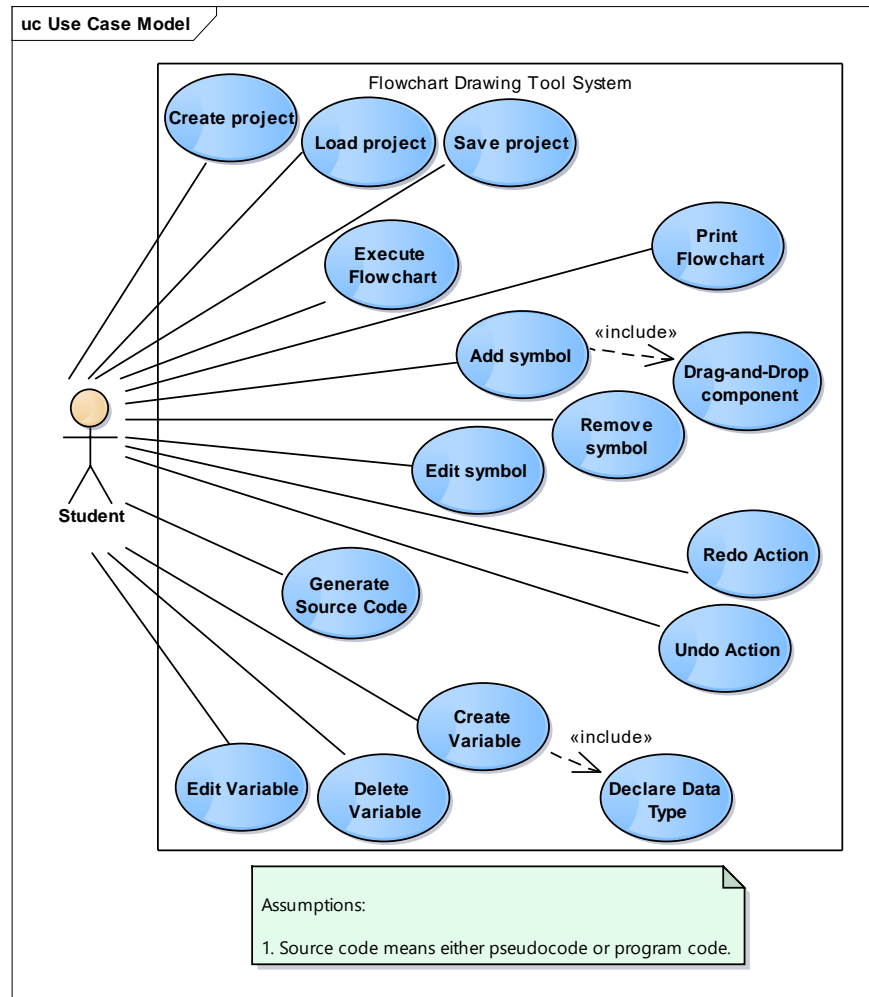


Figure 4.7: Use Case Diagram

4.5.2 Use Case Descriptions

Each of the use case shown in the use case diagram was elaborated in more detail in each of their respective use case descriptions. The use case descriptions could be found at Appendix I.

4.6 FDD Deliverables

4.6.1 Develop an Overall Model

The overall domain model of the system is separated into 3 parts due to the high number of classes involved. They could be found in Appendix J, Figure 1 to Figure 3.

4.6.2 Build a Features List

There were a total of seven features in the features list. They were project management, flowchart printing, flowchart designing, variable management, flowchart execution, source code generation and action reversing. Figure 4.8 shows the features list for the proposed flowchart drawing tool.

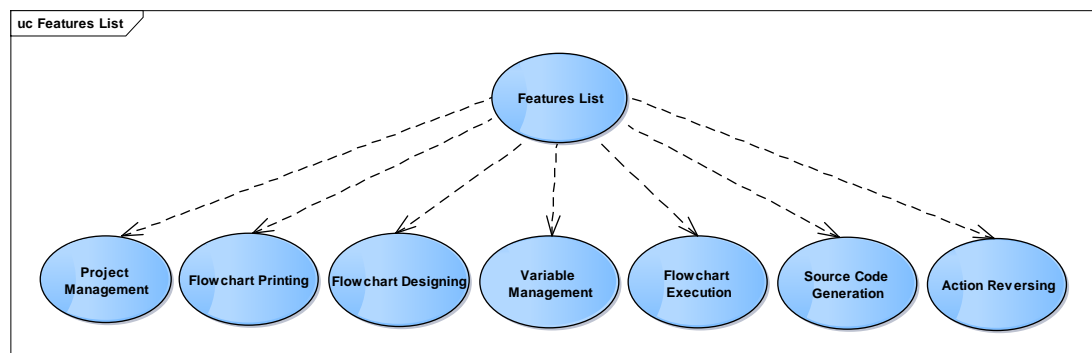


Figure 4.8: Features List

4.6.3 Plan by Feature

When producing the development plan for phases based on features, core functionalities of the system, such as flowchart drawing, were prioritized and developed first compared to usability functionalities like flowchart printing. Figure 4.9 displays the development plan for all the features listed in the features list.

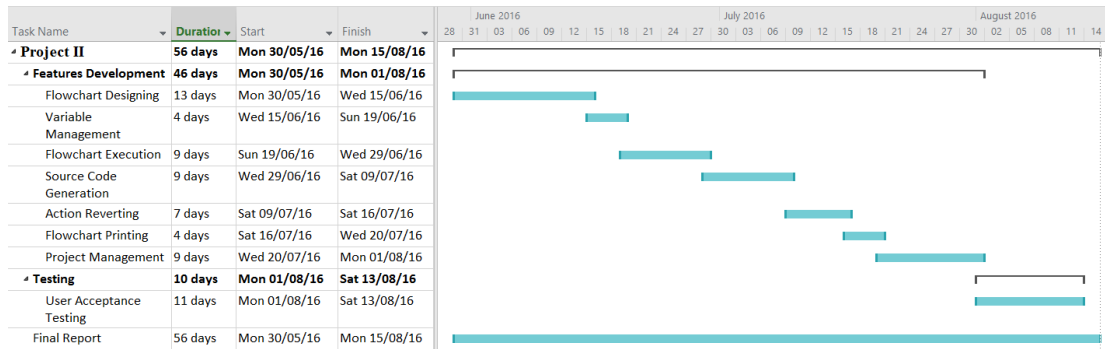


Figure 4.9: Features List Development Plan

4.6.4 Design by Feature

The sequence diagrams for all the features in the features list were produced. Figures 4.10 to 4.16 display all the sequence diagrams for the features.

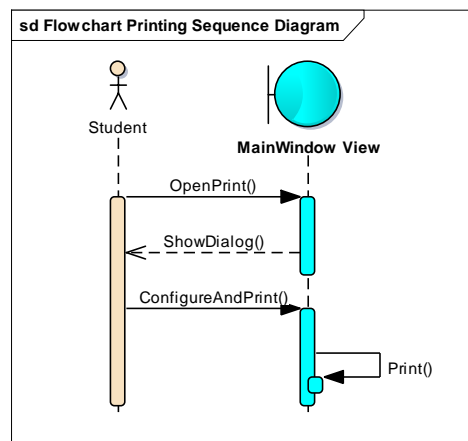


Figure 4.10: Flowchart Printing Sequence Diagram

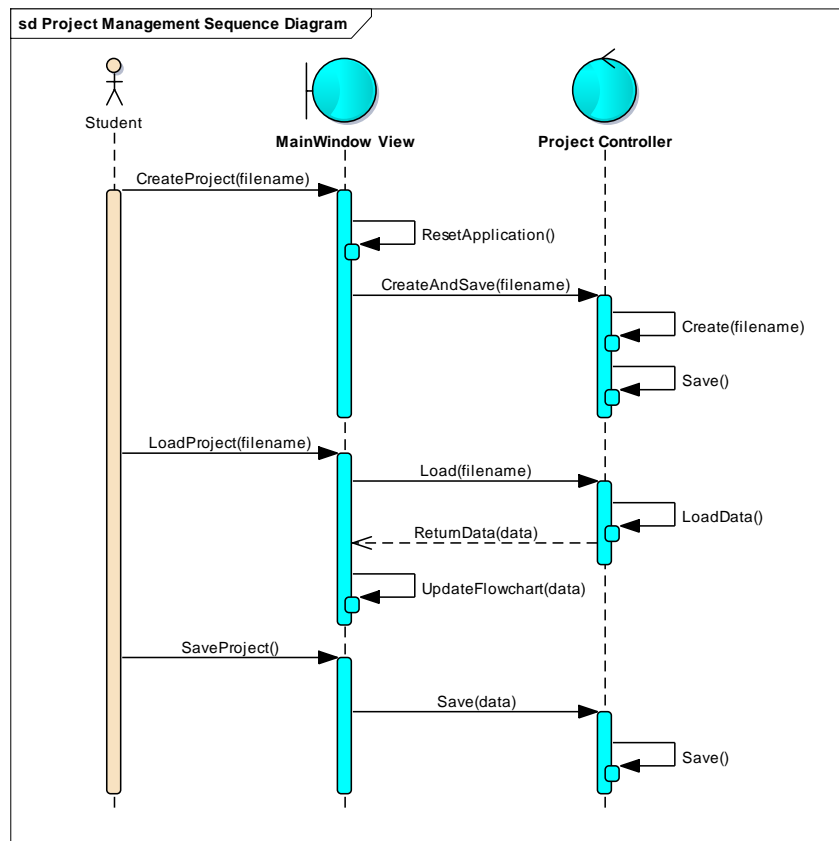


Figure 4.11: Project Management Sequence Diagram

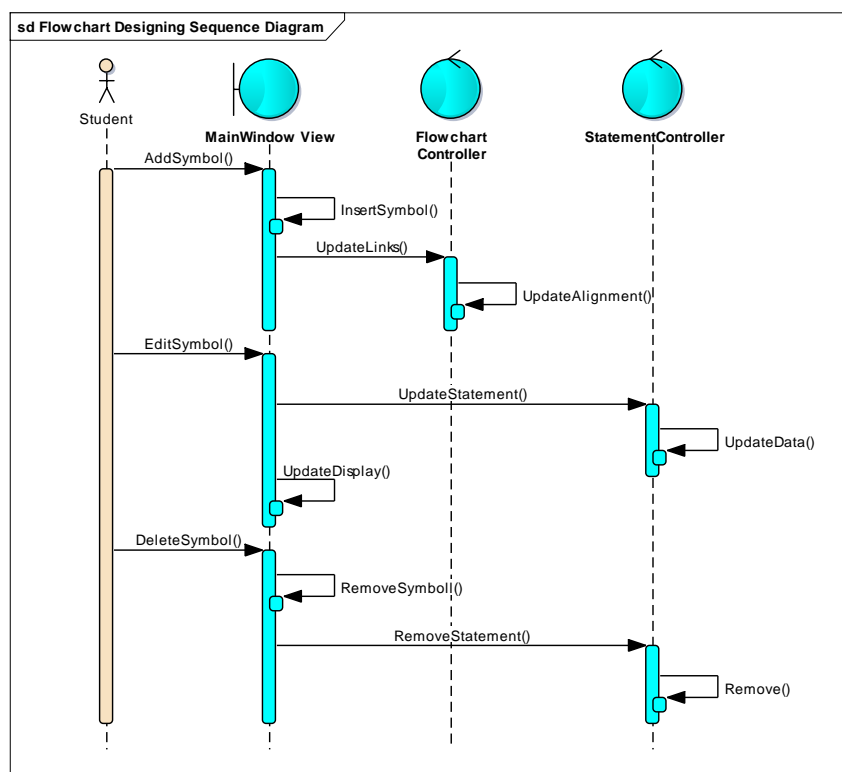


Figure 4.12: Flowchart Designing Sequence Diagram

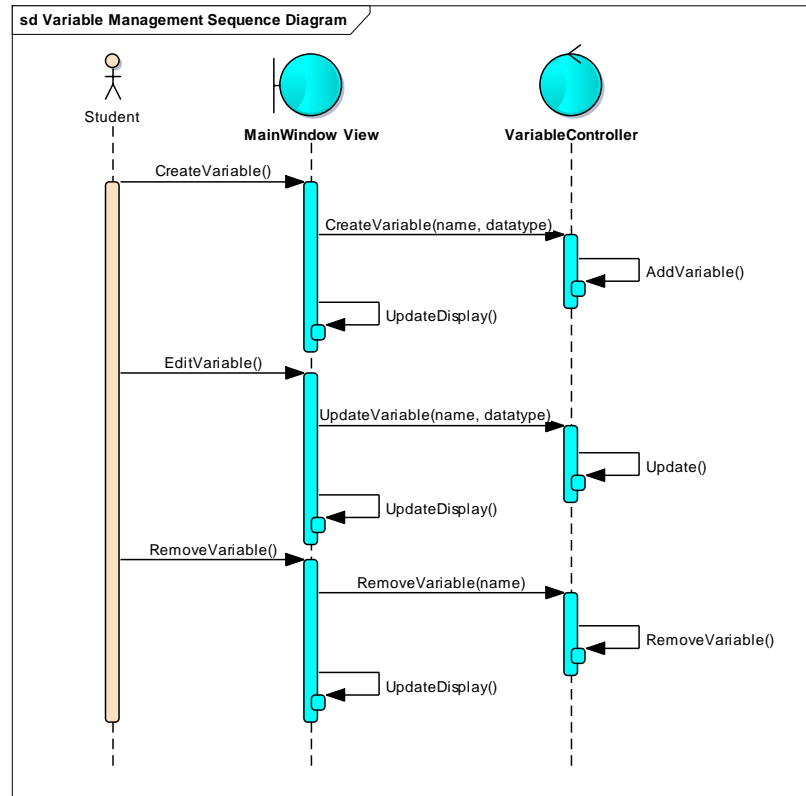


Figure 4.13: Variable Management Sequence Diagram

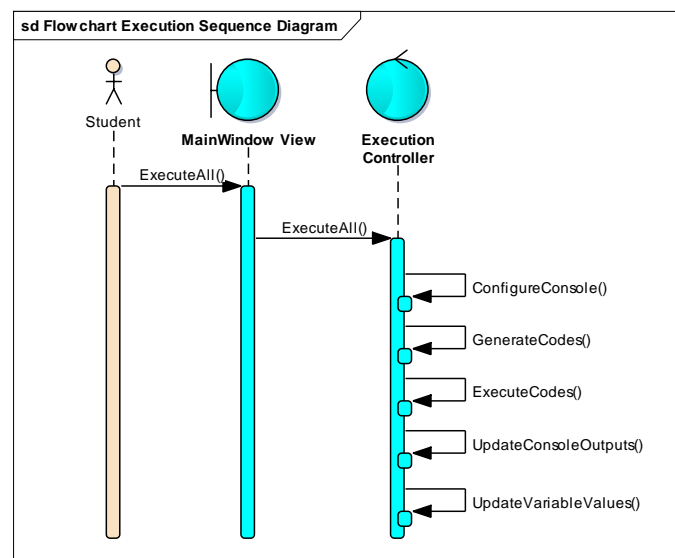


Figure 4.14: Flowchart Execution Sequence Diagram

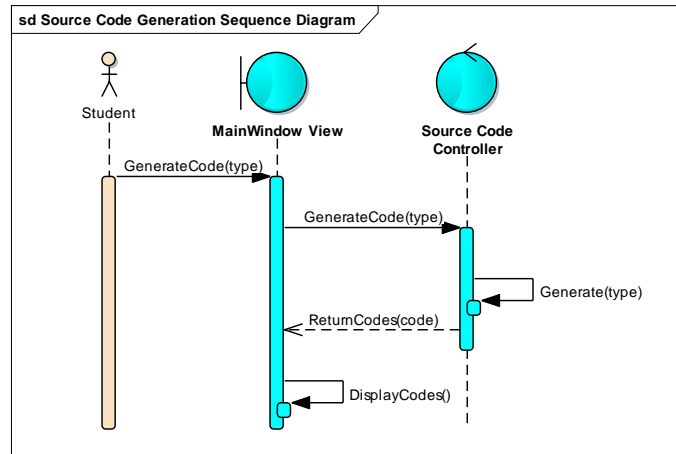


Figure 4.15: Source Code Generation Sequence Diagram

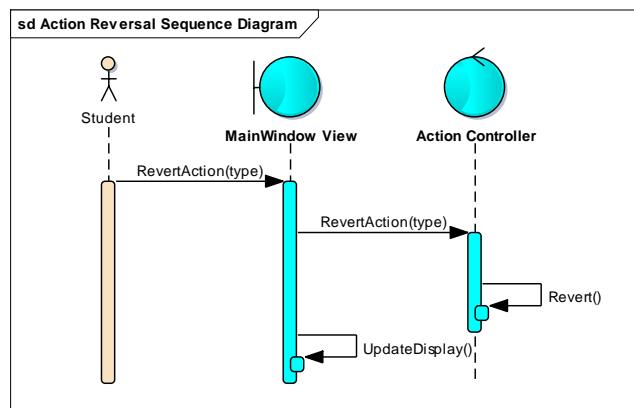


Figure 4.16: Action Reversal Sequence Diagram

4.7 Conclusion

A user interface design, storyboard and use cases for the system to be developed were produced. Then, the results for the first 4 non-development stages of the FDD process were also shown.

CHAPTER 5

PROJECT IMPLEMENTATION AND TESTING

5.1 Class Diagrams

The overall class diagram for the developed tool was created in three diagrams with only the classes' names due to the high number of class involved. Then, all the classes with attributes and operations were created separately. The class diagrams could be found at Appendix J.

5.2 Version Control

During development, a source control management system was utilised to keep track on changes made to the source code over time up until completion. The source control management system used was Git. To ensure the changes made to Git were more secure, a private repository was created in addition to the local repository in the computer developing the tool. This was to ensure that there was still a backup repository that could be accessed on other computers should any issues occur on the current computer being used to develop the tool. On top of that, the local repository could be reverted to the original state stored in the private repository if any local changes that occurred and were committed were to be discarded. Figure 5.1 shows the number of commits for each week, starting from the week of 30 May until the week of 14 August.

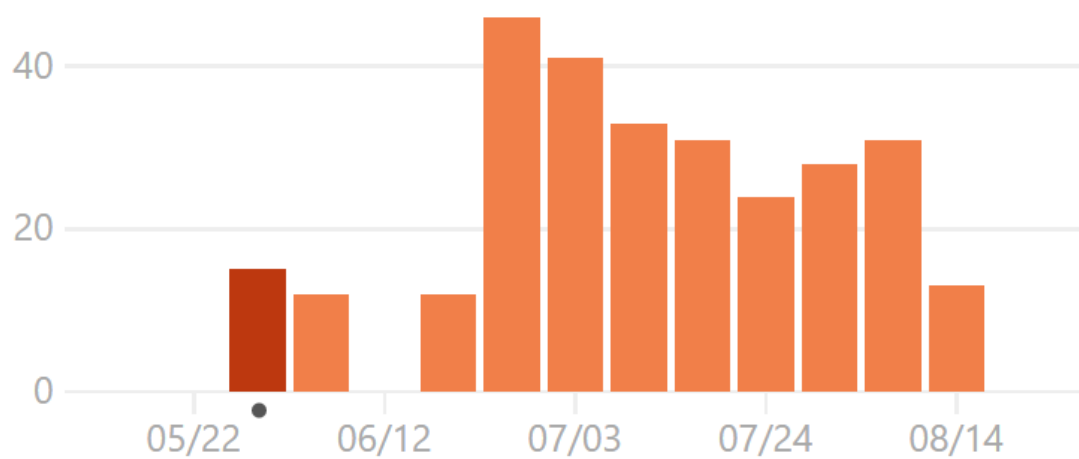


Figure 5.1: Number of Commits per Week

5.3 Testing

5.3.1 Unit Testing

Unit testing was conducted on the developed tool to reduce the number of bugs in the system and to ensure vital areas of the system work as normal. MSTest was the unit testing framework utilised to create and execute the unit tests for the system. There were a total of 92 unit tests created, and they were grouped into 6 distinct categories. The unit tests could be found at Appendix K.

5.3.2 User Testing

A total of 6 students who just learnt the fundamentals of programming conducted user testing on the developed flowchart drawing tool. The students were mainly first year students pursuing a degree course in Software Engineering in UTAR. They were given a survey to be filled up before they started evaluating the tool. Evaluation of the tool involved familiarizing themselves with the tool and attempting to solve the evaluation questions provided. After they completed the evaluation, they were asked to provide feedback about the developed tool on an evaluation form.

5.3.2.1 Evaluation Process

During the evaluation process, participants were provided two questions in which they had to attempt to solve using the flowchart drawing tool developed. The first question involved warming up the participants on utilising the tool to perform basic actions on the application, while the second question involved introducing to them other functionalities that the tool could perform. The questions for the evaluation are found in Appendix E.

5.3.2.2 Evaluation Feedback

Feedback was gathered from participants who evaluated the developed tool in order to obtain insights about the tool. The feedback questions asked involved retrieving the degree in which the developed tool fulfilled the proposed features listed, the strengths and shortcomings, and any additional feedback about the tool. The feedback questions could be found at Appendix F.

Based on the analysis conducted on the feedback obtained, the user interface, flowchart design process, variables and data type declaration, and source code generation were fulfilled by the developed tool, while flowchart execution, flowchart project management as well as flowchart undo and redo actions were more neutral as these were not very clear and obvious in the tool. However, when asked about the likeable features available in the tool, source code generation topped the responses, followed by undo and redo actions, and flowchart creation. When asked about the least adored features in the tool, undo and redo actions topped the responses, slightly contradicting the previous result indicating it being the next best feature adored by participants. The user interface of the tool also topped this question, suggesting that there was a lack of information and instruction to users to guide them through the features of the tool.

5.4 Conclusion

The final class diagram for the developed tool was produced. The version control system used throughout the development was also included along with screenshots of all the commits created. Additionally, user testing was carried out to verify and validate the tool created to ensure the features developed were the required specifications of its target end-users.

CHAPTER 6

CONCLUSION AND RECOMMENDATIONS

6.1 Contribution

At the end of this project, a flowchart drawing tool was produced. The flowchart drawing tool was able to fulfil all of the objectives stated earlier in the project, as shown in the following:

1. To display the process flow of the program created by students.
(A flowchart designed by the students could be displayed in the tool to show the program flow.)
2. To evaluate the usefulness and effectiveness of the tool produced.
(The tool was evaluated by students who were newly exposed to programming for maximum feedback on the usefulness and effectiveness of the tool.)
3. To validate the accuracy of algorithms devised by students to solve their problems.
(The tool could be executed by students to determine whether the algorithms devised by them produced the correct outputs.)
4. To improve effectiveness of drawing flowcharts, by reducing error rates while drawing flowcharts.
(Students only require to drag-and-drop their target flowchart symbols on the arrows provided, and the tool will automatically draw and insert the symbol and any arrows into the appropriate location in the flowchart.)

5. To improve efficiency of drawing flowcharts, by saving time in the drawing of flowcharts.
(Students only require to drag-and-drop their target flowchart symbols on the arrows provided, and the tool will automatically insert the symbol and any additional arrows into the flowchart.)
6. To incorporate core programming principles, such as variable declarations, data types, control structures, and correct programming syntax.
(Variables could be created with multiple data type choices, decision and loops were included as flowchart symbols, and correct programming syntax in writing source codes could be seen from the source codes generated.)
7. To provide the ability to execute visual programs, similar to coded programs.
(The flowchart could be executed by the tool to produce outputs.)
8. To provide the ability to generate the designed flowchart's equivalent in program code.
(Syntactically correct source code in C or C++ could be generated by the tool.)

6.2 Limitations

Although the flowchart drawing tool developed had achieved its intended objectives, there were still several limitations on the tool due to lack of time and scope. The following was a list of the limitations of the tool:

1. Nested decisions, nested loops, loops in decision statements, and pseudocode generation were not included in the tool due to time constraints.
2. Functions and arrays were deemed slightly too advanced and thus were excluded as the primary focus of this project was to introduce the programming fundamentals to students first.

6.3 Future Enhancements

In the future, the flowchart drawing tool produced could be further enhanced to overcome its limitations as well as increasing its arsenal of features. The following list of features would be useful additions:

1. Nested decisions, nested loops and loops in decision statements
2. Pseudocode generation
3. Function declaration and function call
4. Variables with array data types
5. Execution step-by-step
6. Synchronized source code and flowchart displayed together in the same window

6.4 Conclusion

This project successfully produced a flowchart drawing tool that fulfilled the project objectives. Despite of that milestone, the tool still contained limitations that prevented it from achieving its full potential. Thus, future enhancements on the tool were encouraged in order to enable it to be commercialized.

REFERENCES

- Ambler, S., 2014. *Feature Driven Development (FDD) and Agile Modelling*. [online] Available at: <<http://www.agilemodeling.com/essays/fdd.htm>> [Accessed 19 March 2016].
- Areias, C. and Mendes, A., 2007, June. A tool to help students to develop programming skills. In *Proceedings of the 2007 international conference on Computer systems and technologies* (p. 89). ACM.
- Areias, C.M., Mendes, A.J. and Gomes, A.J., 2007. Learning to program with ProGuide. In *Proc. of International Conference on Engineering Education–ICEE*.
- Carlisle, M.C., Wilson, T.A., Humphries, J.W. and Hadfield, S.M., 2005, February. RAPTOR: a visual programming environment for teaching algorithmic problem solving. In *ACM SIGCSE Bulletin* (Vol. 37, No. 1, pp. 176-180). ACM.
- Chen, S. and Morris, S., 2005. Iconic programming for flowcharts, java, turing, etc. *ACM SIGCSE Bulletin*, 37(3), pp.104-107.
- Crews, T. and Ziegler, U., 1998, November. The flowchart interpreter for introductory programming courses. In *Frontiers in Education Conference, 1998. FIE'98. 28th Annual* (Vol. 1, pp. 307-312). IEEE.
- Gomes, A. and Mendes, A.J., Interactive system for algorithm development and simulation.
- Hall, M.S., 2007. Raptor: nifty tools. *Journal of Computing Sciences in Colleges*, 23(1), pp.110-111.
- ISTQB Exam Certification, n.d.a. *What is Agile model – advantages, disadvantages and when to use it?* [online] Available at: <<http://istqbexamcertification.com/what-is-agile-model-advantages-disadvantages-and-when-to-use-it/>> [Accessed 13 March 2016].
- ISTQB Exam Certification, n.d.b. *What is RAD model – advantages, disadvantages and when to use it?* [online] Available at: <<http://istqbexamcertification.com/what-is-rad-model-advantages-disadvantages-and-when-to-use-it/>> [Accessed 13 March 2016].

- Cilliers, C., Calitz, A. and Greyling, J., 2005. The Application of The Cognitive Dimension Framework for Notations as an Instrument for the Usability analysis of an Introductory Programming tool. *Alternation Journal*, 12, pp.543-576.
- Lawton, R., 2015. *Feature Driven Development: A Guide*. [online] Available at: <<http://www.arrkgroup.com/thought-leadership/feature-driven-development-a-guide/>> [Accessed 21 March 2016].
- Marcelino, M., Mihaylov, T. and Mendes, A., 2008, October. H-SICAS, a handheld algorithm animation and simulation tool to support initial programming learning. In *2008 38th Annual Frontiers in Education Conference* (pp. T4A-7). IEEE.
- Marcelino, M., Gomes, A., Dimitrov, N. and Mendes, A., 2004, June. Using a computer-based interactive system for the development of basic algorithmic and programming skills. In *International Conference on Computer Systems and Technologies (e-Learning)* (pp. 8-6).
- Mendes, A.J., Gomes, A., Esteves, M., Marcelino, M.J., Bravo, C. and Redondo, M.A., 2005. Using simulation and collaboration in CS1 and CS2. *ACM SIGCSE Bulletin*, 37(3), pp.193-197.
- Scott, A., n.d.. *Progranimate, Program Visualisation and Animation for Novices*. [online] Available at: <<http://www.progranimate.com/aboutProgranimate/aboutMain.html>> [Accessed 28 February 2016].
- Scott, A., Watkins, M. and McPhee, D., 2007. A Step Back From Coding-An Online Environment and Pedagogy for Novice Programmers. In *Proceedings of the 11th Java in the Internet Curriculum Conference* (pp. 35-41).
- Scott, A., Watkins, M. and McPhee, D., 2008a. Progranimate-A Web Enabled Algorithmic Problem Solving Application. In *CSREA EEE* (pp. 498-508).
- Scott, A., Watkins, M. and McPhee, D., 2008b, April. E-Learning For Novice Programmers; A Dynamic Visualisation and Problem Solving Tool. In *Information and Communication Technologies: From Theory to Applications, 2008. ICTTA 2008. 3rd International Conference on* (pp. 1-6). IEEE.
- Shneiderman, B., Mayer, R., McKay, D. and Heller, P., 1977. Experimental investigations of the utility of detailed flowcharts in programming. *Communications of the ACM*, 20(6), pp.373-381.
- SmartDraw, n.d.. *Flowchart Types and Uses*. [online] Available at: <<https://www.smartdraw.com/flowchart/flowchart-types.htm>> [Accessed 26 February 2016].

UK Essays, 2013. *Iterative and Incremental Development Of Software Models Information Technology Essay*. [online] Available at: <<http://www.ukessays.com/essays/information-technology/iterative-and-incremental-development-of-software-models-information-technology-essay.php?cref=1>> [Accessed 13 March 2016].

Xinogalos, S., 2013, March. Using flowchart-based programming environments for simplifying programming and software engineering processes. In *Global Engineering Education Conference (EDUCON), 2013 IEEE* (pp. 1313-1322). IEEE.

Xinogalos, S. and Satratzemi, M., 2004. Introducing novices to programming: a review of teaching approaches and educational tools. In *International Conference on Education and Information Systems, Technologies and Applications (EISTA 2004)*, Orlando, USA, on July 21 (Vol. 25, No. 2004, pp. 60-65).

APPENDICES

APPENDIX A: Interview Questions

1. How do students do their exercise when they do practical exercises on flowcharts?
 - (a) Drawing by hand
 - (b) Drawing using tools
 - (c) Others, please specify: _____

2. Do students have confusion when drawing symbols or writing codes?
 - (a) Yes
 - (b) No

3. On a scale of 1 to 5, with 1 being Very Easy and 5 being Very Hard, rate whether each of the following programming concepts are difficult to be understood by students:

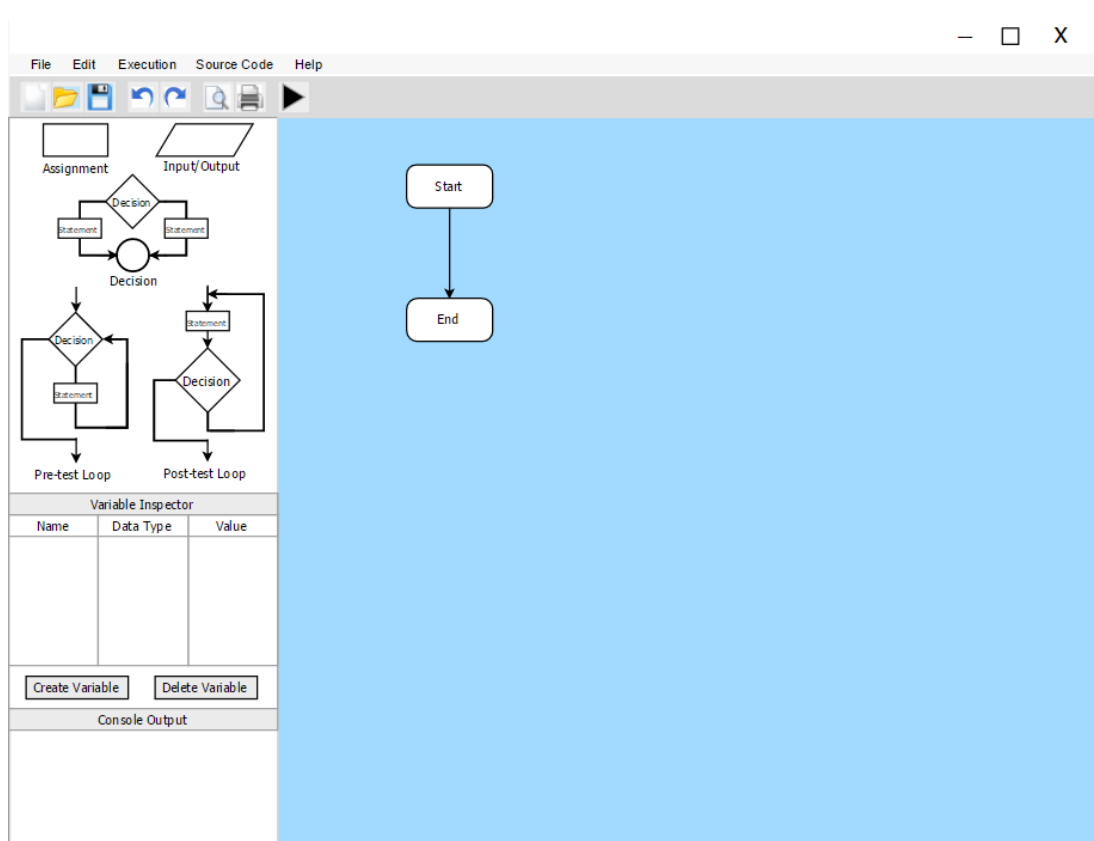
Programming Concepts	Very Easy 1	Easy 2	Normal 3	Hard 4	Very Hard 5
(a) Control Structures					
(b) Data Types					
(c) Variables					
(d) Functions					
(e) Flowchart					

(f) Flowchart Symbols					
(i) Terminal					
(ii) Assignment					
(iii) Input/Output					
(iv) Connector					
(v) Flowlines					
(vi) Decision					
(vii) Loops					

4. Are you aware that there are existing tools that could be used to teach programming to students in the form of flowcharts?
- (a) Yes
- (b) No
5. If you answered 'Yes' in Question 4, please answer the following questions:
- (a) Which tool(s) do you know of?
- (i) Progranimate
- (ii) RAPTOR
- (iii) Iconic Programmer
- (iv) Others, please specify: _____
- (b) Did you attempt to teach students to program and create flowcharts using any of these tool(s)?
- (i) Yes
- (ii) No
- (c) Do you feel that these tool(s) are suitable to be used by students to learn programming and to create flowcharts?
- (i) Yes
- (ii) No

- (d) What are your suggestions for improvement on these tool(s), if any, in order to make them more suitable to be used as part of the teaching materials?

Suggestions:



6. The design shown is an interface of a flowchart drawing tool currently under development. The flowchart drawing tool features managing flowchart projects, creating variables and defining data types, flowchart drawing, program execution and generation of C++ source code from the created flowchart. Based on the interface and these features, how useful do you think are the following (Please elaborate, if possible):

- (a) Interface Design

- (b) Flowchart Project Management

(c) Variable Declaration

(d) Data Type Definition

(e) Flowchart Drawing

(f) Program Execution

(g) Source Code Generation from Flowchart

7. Do you think the usage of this tool in learning programming and drawing flowchart will improve learning among students?
- (a) Yes
 - (b) No
 - (c) Maybe
 - (d) I don't know

APPENDIX B: Interview Replies

Participant 1 – A **Participant 2 – B** **Participant 3 – C**

1. How do students do their exercise when they do practical exercises on flowcharts?
 - (a) Drawing by hand **A B C**
 - (b) Drawing using tools
 - (c) Others, please specify: _____

2. Do students have confusion when drawing symbols or writing codes?
 - (a) Yes **A B C**
 - (b) No

3. On a scale of 1 to 5, with 1 being Very Easy and 5 being Very Hard, rate whether each of the following programming concepts are difficult to be understood by students:

Programming Concepts	Very Easy 1	Easy 2	Normal 3	Hard 4	Very Hard 5
(a) Control Structures			A	B	C
(b) Data Types	A	B	C		
(c) Variables	A	B C			
(d) Functions				A B	C
(e) Flowchart		A	B	C	

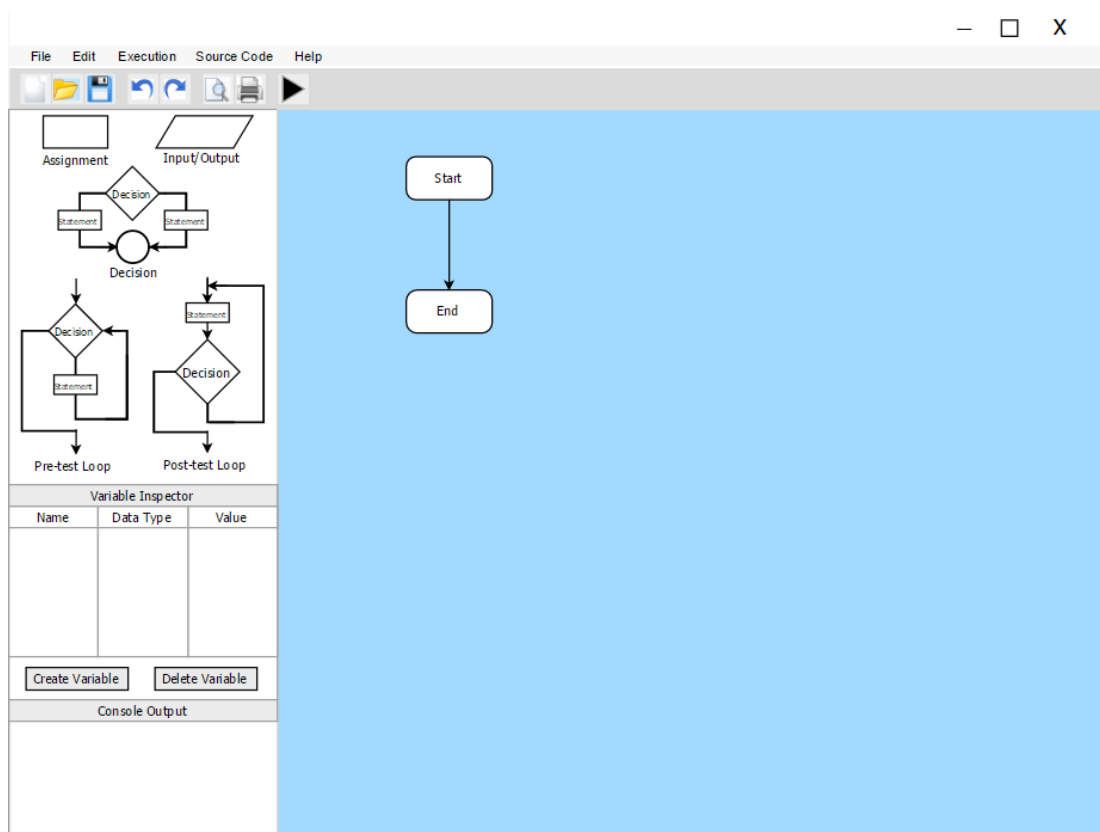
(f) Flowchart Symbols					
(i) Terminal	B	A		C	
(ii) Assignment		A	B	C	
(iii) Input/Output		A	B	C	
(iv) Connector		A	B	C	
(v) Flowlines		A		B C	
(vi) Decision		A		B C	
(vii) Loops			A	B C	

4. Are you aware that there are existing tools that could be used to teach programming to students in the form of flowcharts?
- (a) Yes **B C**
- (b) No **A**
5. If you answered 'Yes' in Question 4, please answer the following questions:
- (a) Which tool(s) do you know of?
- (i) Progranimate
- (ii) RAPTOR
- (iii) Iconic Programmer **B**
- (iv) Others, please specify: **C – Cannot remember the name of the tools**
- (b) Did you attempt to teach students to program and create flowcharts using any of these tool(s)?
- (i) Yes
- (ii) No **B C**
- (c) Do you feel that these tool(s) are suitable to be used by students to learn programming and to create flowcharts?
- (i) Yes **B C**
- (ii) No

- (d) What are your suggestions for improvement on these tool(s), if any, in order to make them more suitable to be used as part of the teaching materials?

Suggestions:

- (i) **B – Do not be too restricted in the level and symbol used**
- (ii) **C – The key problem is that the student can't visualise the flow of program execution. Suggest providing animated diagram to show execution flow of their design algorithm. It can help them counter-check the correctness of the algorithm.**



6. The design shown is an interface of a flowchart drawing tool currently under development. The flowchart drawing tool features managing flowchart projects, creating variables and defining data types, flowchart drawing, program execution and generation of C++ source code from the created flowchart. Based on the interface and these features, how useful do you think are the following (Please elaborate, if possible):

(a) Interface Design

- (i) **A – Look complicated, no idea how to start**
- (i) **B – Good**
- (i) **C – No title for each section.**
- (ii) **No section to show source code.**
- (iii) **Each section can be closed/minimised and zoom level can be adjusted, so that user can view one or two section(s) in larger/clearer view.**

(b) Flowchart Project Management

- (i) **A – No idea**
- (i) **B – ???**
- (i) **C – No suggestion.**

(c) Variable Declaration

- (i) **A – Put under variable inspector column?**
- (i) **B – Ok, good. But any differentiation between local or global variable?**
- (i) **C – When to declare variable in scope? In OO, the variable can be declared right before it is being used; not necessary start at the beginning of the function/method.**

(d) Data Type Definition

- (i) **A – No idea**
- (i) **B – Good. But should have choice as well.**
- (i) **C – No issue found.**

(e) Flowchart Drawing

- (i) **A – No idea**
- (i) **B – Should be automatic right?**
- (i) **C – Suggest that the control objects (assignment/input/output/decision/loop) can be clicked and dragged into drawing workspace, and then the user only need to modify the statements (e.g. add $z \leftarrow x$**

+ y for assignment, add $x > y$ for decision, etc.); the flowlines/arrows will be redrawn automatically.

(f) Program Execution

- (i) **A** – No idea
- (i) **B** – OK
- (i) **C** – Suggest providing animation to show execution flow.

(g) Source Code Generation from Flowchart

- (i) **A** – No idea
- (i) **B** – Good
- (i) **C** – It will be good to show flowchart and source code side-by-side.

7. Do you think the usage of this tool in learning programming and drawing flowchart will improve learning among students?
- (a) Yes
 - (b) No
 - (c) Maybe **A B C**
 - (d) I don't know

APPENDIX C: Pre-Evaluation Survey Questions

FYP II Survey

I am a final year UTAR student currently undergoing my Final Year Project Part 2 (FYP II), and I am currently developing a flowchart drawing tool for new students learning the basics of programming.

The purpose of this survey is to gather feedback from students who recently learnt the basics of programming in order to gain ideas on certain areas relating to the learning process of programming.

I would like to thank you personally and on behalf of other UTAR students who would benefit from your assistance in doing this survey. Your participation is very much appreciated. None of the questions posted here will require any information about you, and any responses given here will be kept for use only for FYP II.

* Required

Survey Questions

1. On a scale of 1 to 5, with 1 being least difficult and 5 being most difficult, how hard is it to learn programming and draw flowcharts? *

	1	2	3	4	5	
Least Difficult						Most Difficult

2. How do you draw flowcharts for practicals and assignments requiring you to draw flowcharts? * (Choose 1 only)

- Hand-Drawing
 Microsoft Visio
 Paint
 Flowchart drawing tools
 Other: _____

3. On a scale of 1 to 5, with 1 being least difficult and 5 being most difficult, how hard is it to draw flowcharts using the selected tool? *

	1	2	3	4	5	
Least Difficult						Most Difficult

4. What are the problems that you commonly face when using the selected tool to draw flowcharts? * (Check all that apply)

- Too much drawing
- Many steps to create each symbol
- Confusing interface
- Cannot load and save flowchart
- Other: _____

5. What additional features would you like to see on the tool you used to draw flowcharts? * (Check all that apply)

- More user-friendly interface
- Flowchart execution
- Variables declaration
- Source code generation
- Create, load and save flowchart project
- Other: _____

6. On a scale of 1 to 5, with 1 being least difficult and 5 being most difficult, rate whether the following are difficult to understand? *

	1	2	3	4	5
Decision (if...else)					
Loop (for loop, while loop, do...while loop)					
Data Type (bool, char, float, int, string)					
Variable (naming convention, usage)					
Flowchart Symbols (Assignment, Input/Output, Decision, Counter-Controlled Loop, Pre-test Loop, Post-test Loop)					

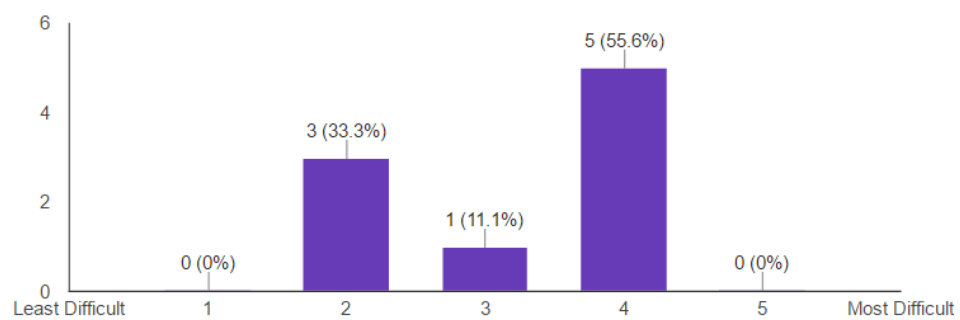
7. On a scale of 1 to 5, with 1 being least important and 5 being most important, rate the importance of having the following features in a flowchart drawing tool? *

	1	2	3	4	5
User-friendly interface					
Create flowcharts quickly and easily					
Flowchart execution					
Declare variables and data types					
Generate source code from flowchart					
Create, load and save flowchart project					
Undo and redo added and removed flowchart symbols					

APPENDIX D: Pre-Evaluation Survey Results

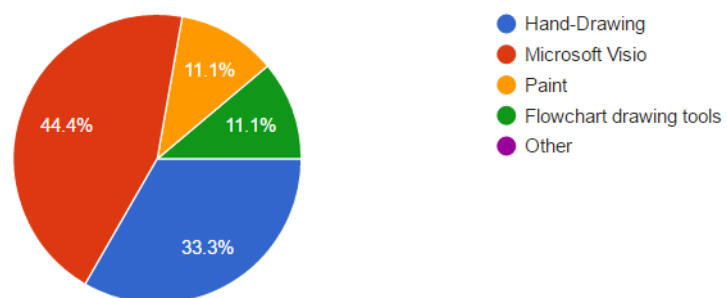
On a scale of 1 to 5, with 1 being least difficult and 5 being most difficult, how hard is it to learn programming and draw flowcharts?

(9 responses)



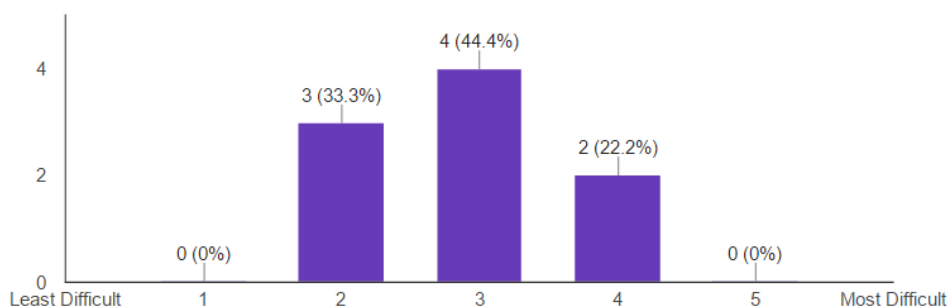
How do you draw flowcharts for practicals and assignments requiring you to draw flowcharts?

(9 responses)

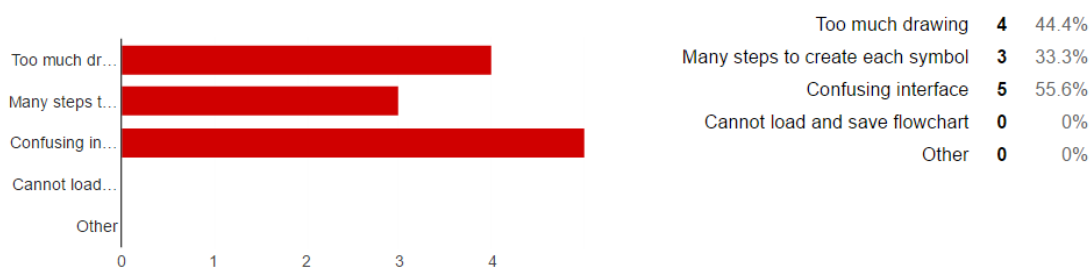


On a scale of 1 to 5, with 1 being least difficult and 5 being most difficult, how hard is it to draw flowcharts using the selected tool?

(9 responses)

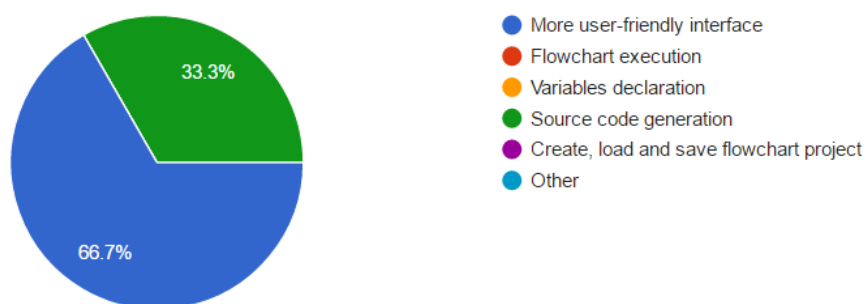


What are the problems that you commonly face when using the selected tool to draw flowcharts?



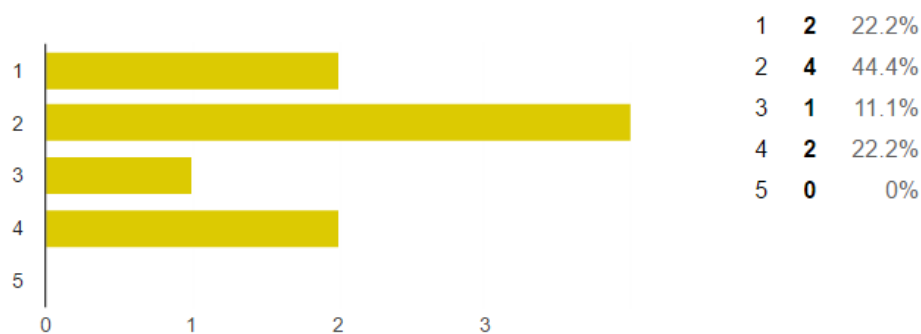
What additional features would you like to see on the tool you used to draw flowcharts?

(9 responses)

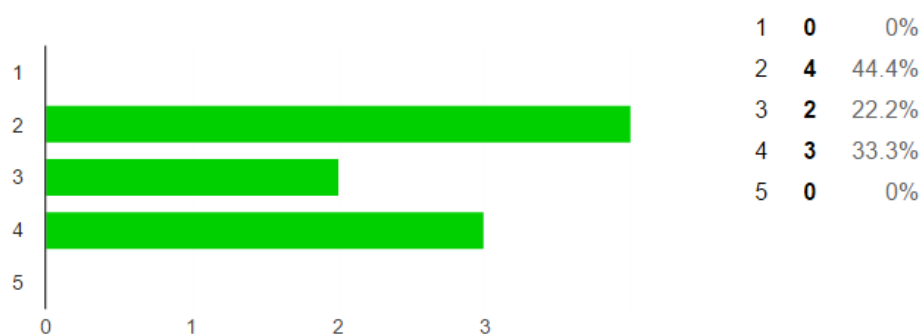


From a scale of 1 to 5, with 1 being the least difficult and 5 being the most difficult, rate whether the following are difficult to understand:

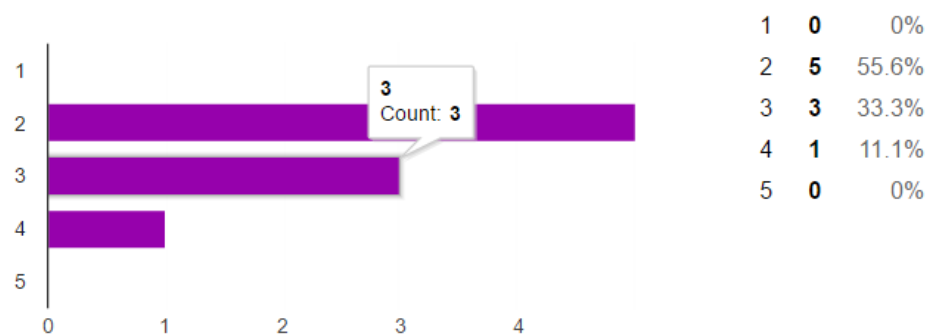
Decision (if...else)



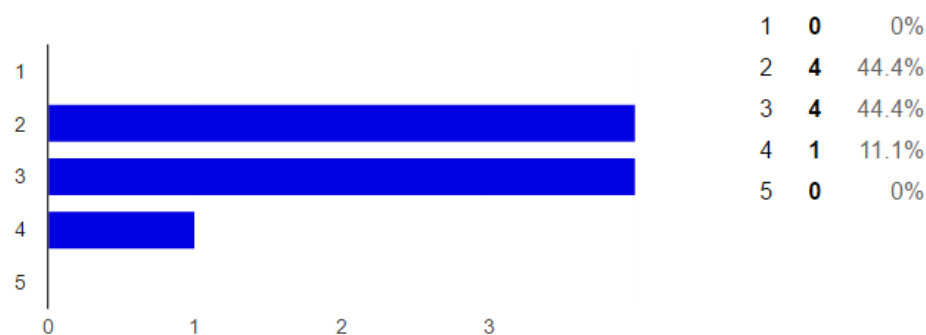
Loop (for loop, while loop, do...while loop)



Data Type (bool, char, float, int, string)

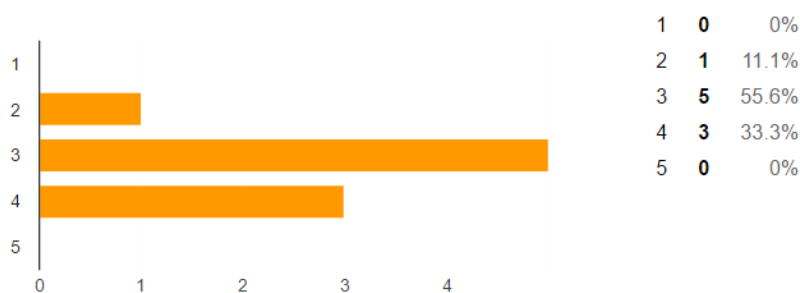


Variable (naming convention, usage)



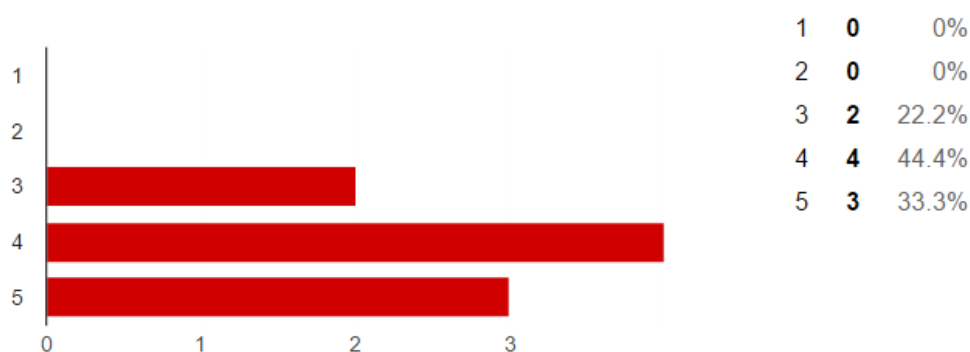
Flowchart Symbols

(Assignment, Input/Output, Decision, Counter-Controlled Loop, Pre-test Loop, Post-test Loop)

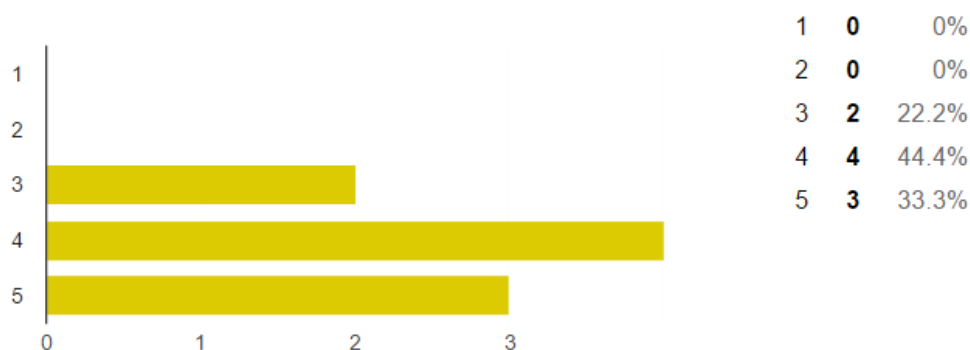


From a scale of 1 to 5, with 1 being the least important and 5 being the most important, rate the importance of having the following features in a flowchart drawing tool:

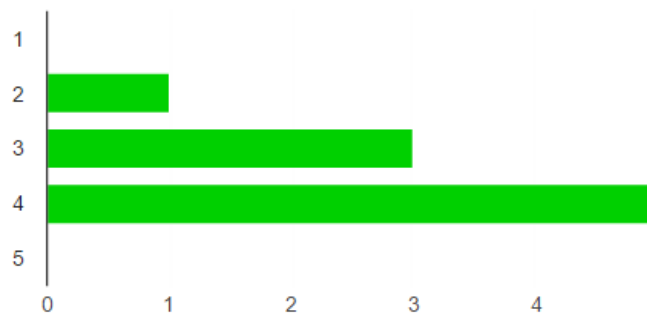
User-friendly interface



Create flowcharts quickly and easily

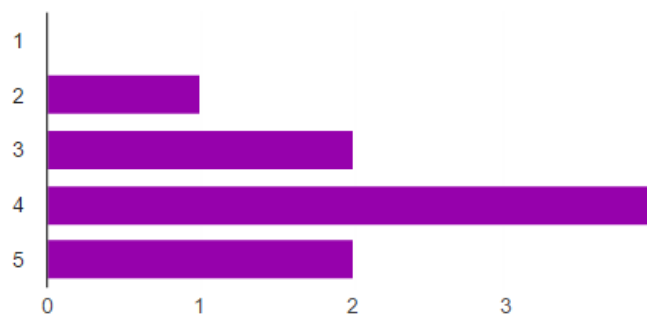


Declare variables and data types



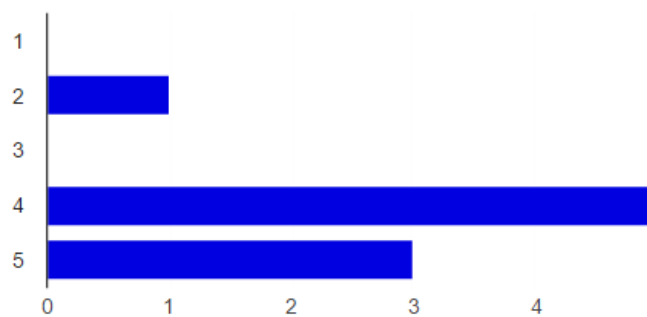
1	0	0%
2	1	11.1%
3	3	33.3%
4	5	55.6%
5	0	0%

Flowchart execution



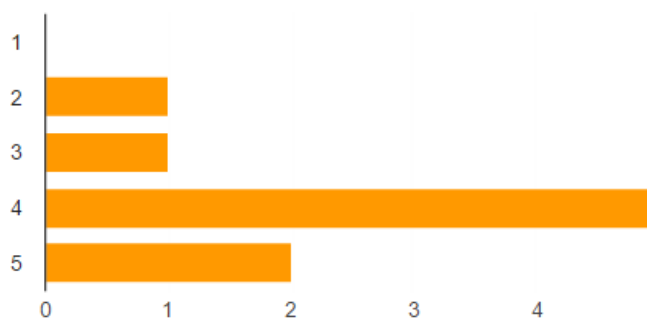
1	0	0%
2	1	11.1%
3	2	22.2%
4	4	44.4%
5	2	22.2%

Generate source code from flowchart



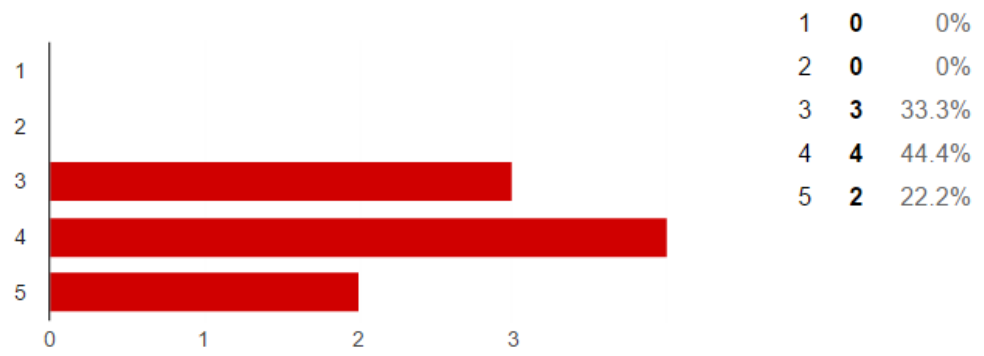
1	0	0%
2	1	11.1%
3	0	0%
4	5	55.6%
5	3	33.3%

Create, load and save flowchart project



1	0	0%
2	1	11.1%
3	1	11.1%
4	5	55.6%
5	2	22.2%

Undo and redo added and removed flowchart symbols



APPENDIX E: Evaluation Questions

1. Design a flowchart that reads an integer input from the user and determines the output to display based on the input entered. If the input entered is bigger than 5, display “Value greater than 5”, else display “Value not greater than 5”. Enter the input(s) required into the Input Window, and then execute the flowchart to make sure the output displayed is correct. Save the flowchart project after you completed it, and then close the application.
2. Load the project you saved. Modify the flowchart by replacing the Decision symbol with a counter-controlled loop. The loop will repeat for the number of times of the user input, and prints out the current loop count. Generate the C++ source code for it and check whether the generated source code is correct.

APPENDIX F: Post-Evaluation Feedback Questions

Flowchart Drawing Tool Evaluation Feedback

This evaluation is used to collect feedback on the flowchart drawing tool being tested.

* Required

1. On a scale of 1 to 5, with 1 being Does Not Fulfil and 5 being Strongly Fulfil, rate whether the flowchart drawing tool fulfilled the following criteria? *

	1	2	3	4	5
User-friendly interface					
Flowcharts could be created quickly and easily					
Variables and data types can be declared					
Flowchart could be executed					
Source code generation					
Create, load and save flowchart project					
Undo and redo flowchart symbols					

2. What do you like best about the tool? * (Check all that apply)

- User-friendly interface
- Flowcharts could be created quickly and easily
- Variables and data types can be declared
- Flowchart could be executed
- Source code generation
- Create, load and save flowchart project
- Undo and redo flowchart symbols
- Other: _____

3. What do you like least about the tool? * (Check all that apply)

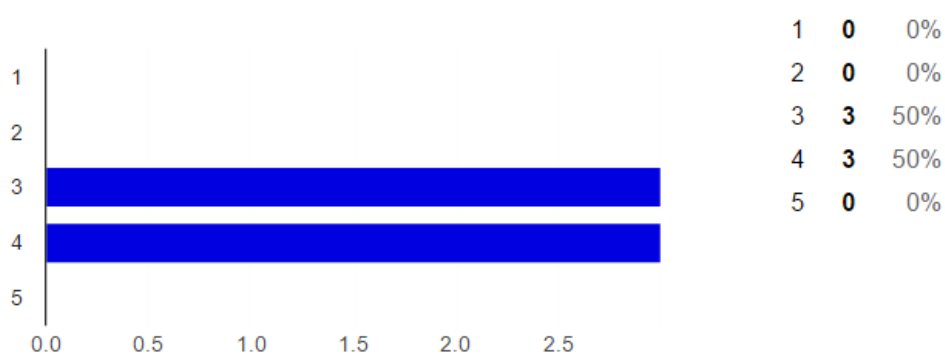
- User-friendly interface
- Flowcharts could be created quickly and easily
- Variables and data types can be declared
- Flowchart could be executed
- Source code generation
- Create, load and save flowchart project
- Undo and redo flowchart symbols
- Other: _____

4. (Optional) Provide any feedbacks or comments for the flowchart drawing tool.

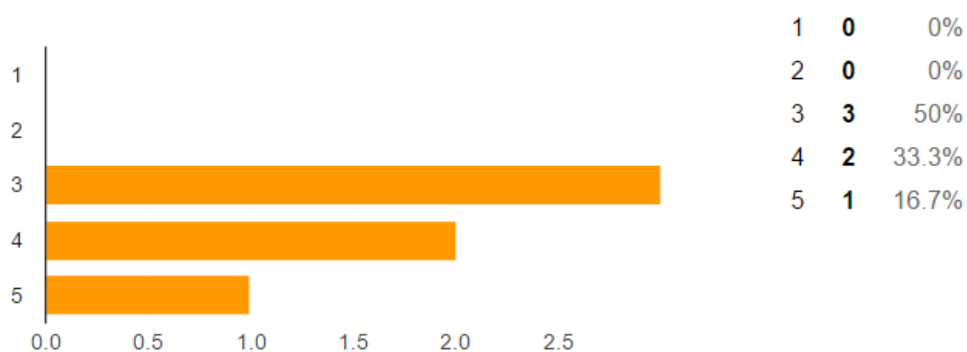
APPENDIX G: Post-Evaluation Survey Results

From a scale of 1 to 5, with 1 being Does Not Fulfil and 5 being Strongly Fulfil, rate whether the flowchart drawing tool fulfilled the following criteria:

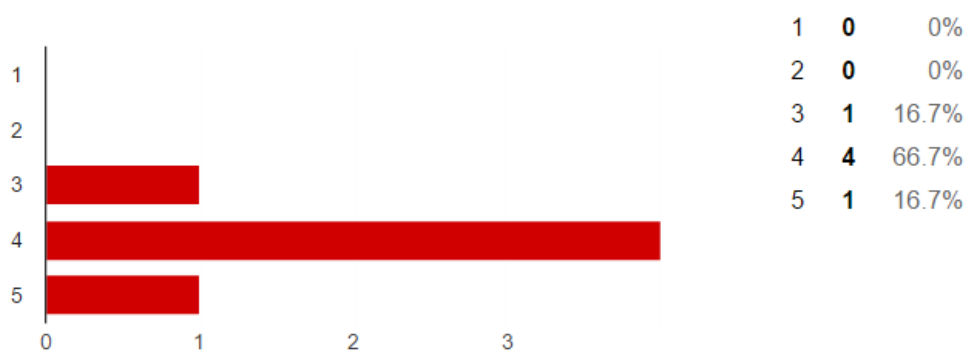
User-friendly interface



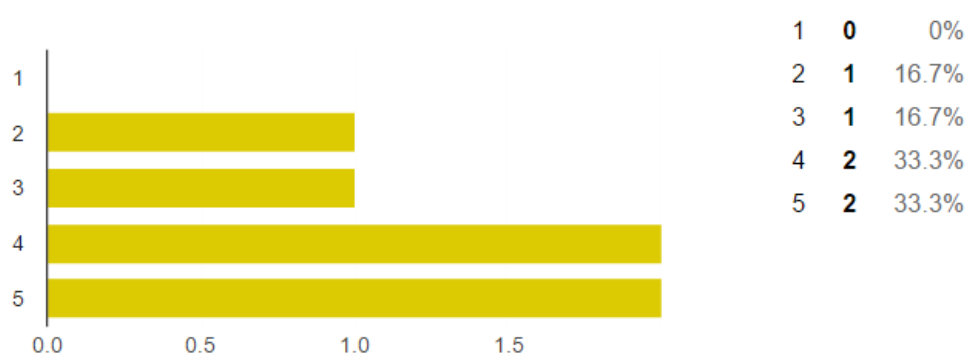
Flowcharts could be created quickly and easily



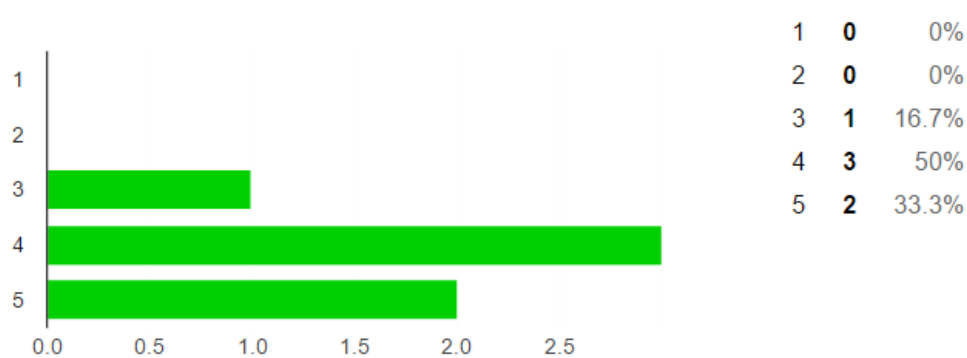
Variables and data types can be declared



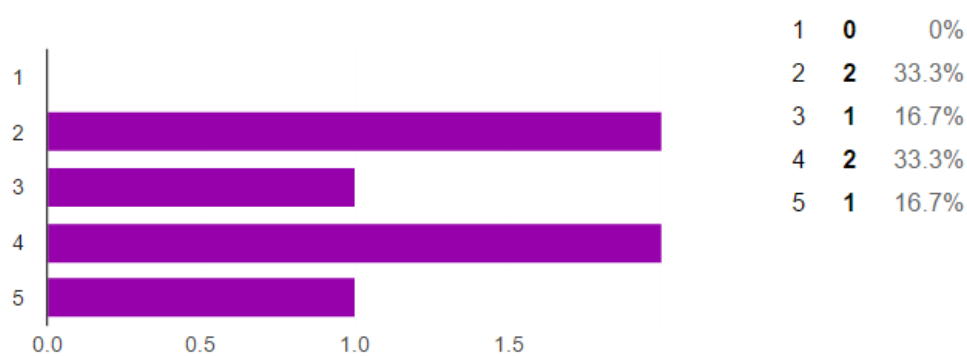
Flowchart could be executed



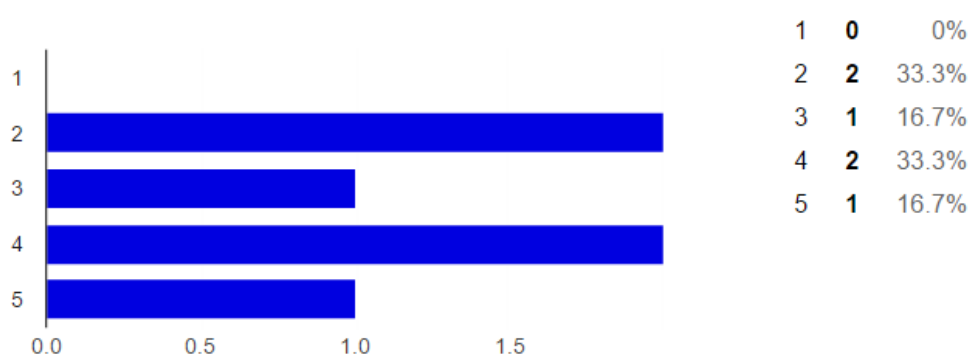
Source code generation



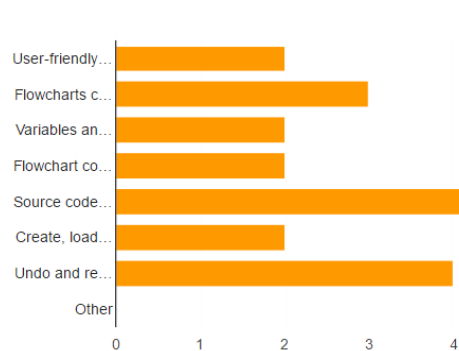
Create, load and save flowchart project



Undo and redo flowchart symbols

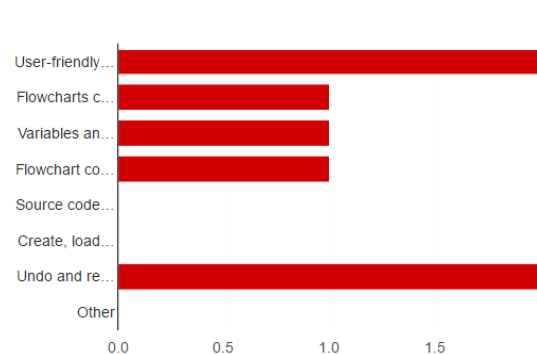


What do you like best about the tool?



User-friendly interface	2	33.3%
Flowcharts could be created quickly and easily	3	50%
Variables and data types can be declared	2	33.3%
Flowchart could be executed	2	33.3%
Source code generation	5	83.3%
Create, load and save flowchart project	2	33.3%
Undo and redo flowchart symbols	4	66.7%
Other	0	0%

What do you like least about the tool?



User-friendly interface	2	33.3%
Flowcharts could be created quickly and easily	1	16.7%
Variables and data types can be declared	1	16.7%
Flowchart could be executed	1	16.7%
Source code generation	0	0%
Create, load and save flowchart project	0	0%
Undo and redo flowchart symbols	2	33.3%
Other	0	0%

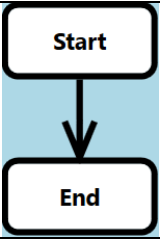
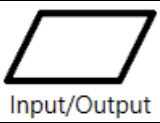
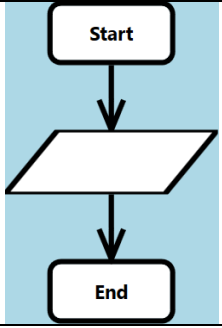
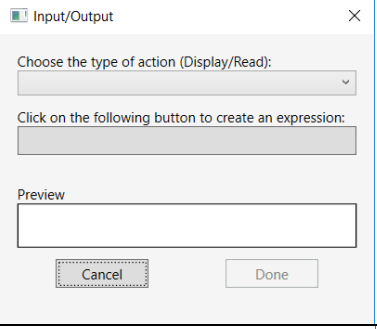
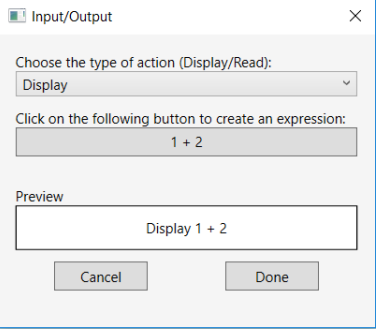
(Optional) Provide any feedbacks or comments for the flowchart drawing tool.

Function to identify user text input and convert into expression would be nice.

clear instruction should be given to user

APPENDIX H: Storyboards

Table H.1: Draw a Flowchart Storyboard

No.	Scene	Step
1.		Create a new project.
2.		A flowchart with initial symbols shown is created.
3.		Select the flowchart symbol shown, drag and drop it on the arrow in the flowchart.
4.		The flowchart will now look like this.
5.		A dialogue box pops up as shown when the flowchart symbol was added.
6.		Select the type of action and enter the expression shown, and press “Done” to close the dialogue box.

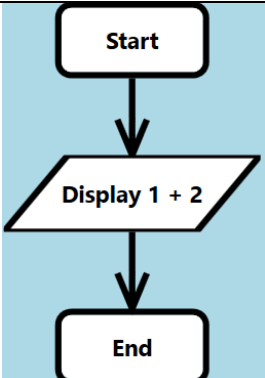
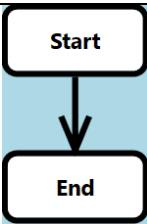
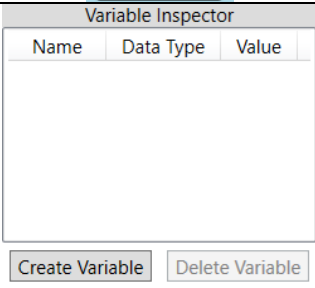
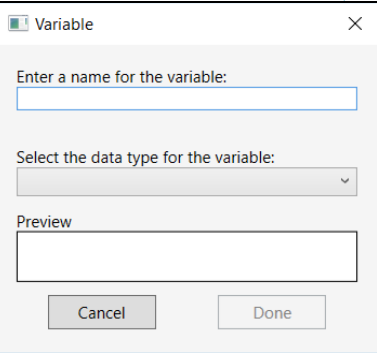
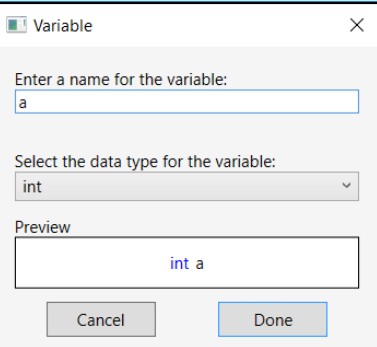
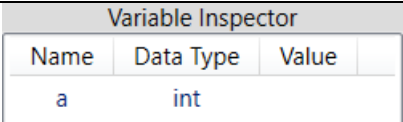

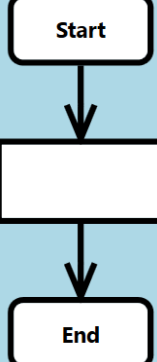
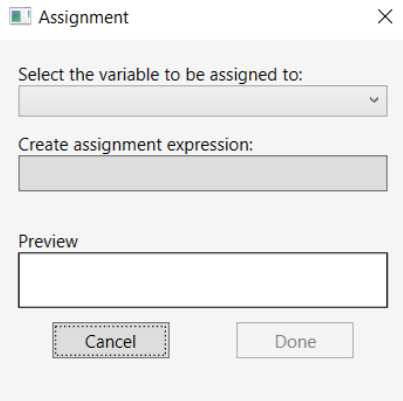
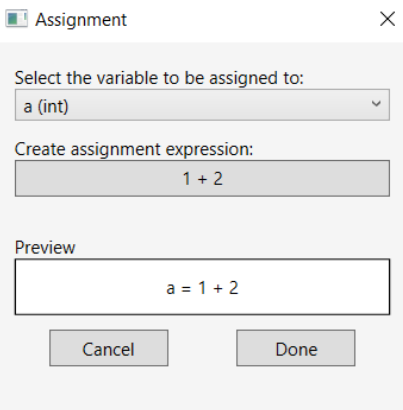
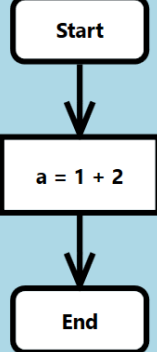
7.		The flowchart is completed and will now look like this.
----	---	---

Table H.2: Create, Edit and Display Variable Storyboard

No.	Scene	Step
1.		Create a new project.
2.		A flowchart with initial symbols shown is created.
3.		Click on the “Create Variable” button in the variable inspector.
4.		A dialogue box pops up as shown.
5.		Enter the values shown and press “Done” to close the dialogue box.

6.		The new variable will appear in the variable inspector as shown.
7.	 <p>Assignment</p>	Select the flowchart symbol shown, drag and drop it on the arrow in the flowchart.
8.		The flowchart will now look like this.
9.		A dialogue box pops up as shown when the flowchart symbol was added.
10.		Select the variable and enter the expression shown, and press “Done” to close the dialogue box.
11.		The flowchart will now look like this.

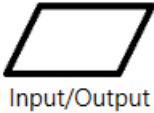
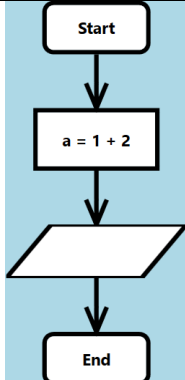
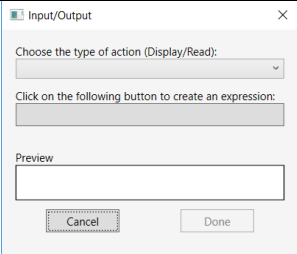
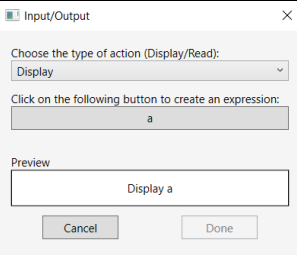
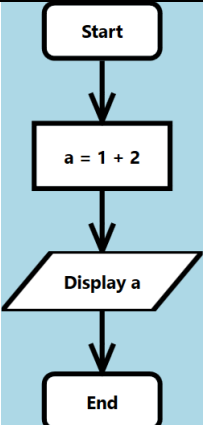

12.	 <p>Input/Output</p>	<p>Select the flowchart symbol shown, drag and drop it on the arrow connecting to the “End” symbol in the flowchart.</p>
13.		<p>The flowchart will now look like this.</p>
14.		<p>Double-click on the newly added flowchart symbol. A dialogue box pops up as shown.</p>
15.		<p>Select the type of action and enter the expression shown, and press “Done” to close the dialogue box.</p>
16.		<p>The flowchart is completed and will now look like this.</p>

Table H.3: Execute Flowchart Storyboard

No.	Scene	Step
1.	 	<p>Click on the “Run All” menu item or shortcut icon shown to execute the flowchart. The flowchart will be executed</p>

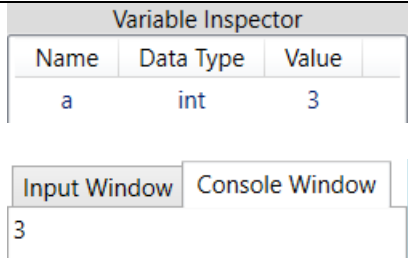
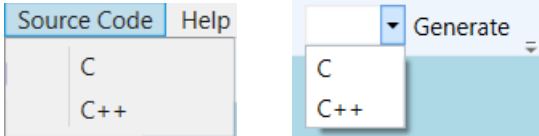
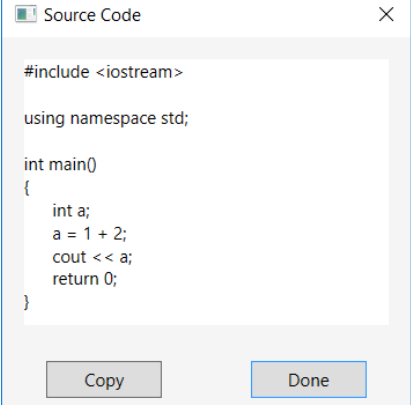
		completely once by the tool.
2.	 <p>The screenshot shows two windows. The 'Variable Inspector' window has a table with three columns: Name, Data Type, and Value. The row contains 'a', 'int', and '3'. Below it, the 'Input Window' contains the number '3' and the 'Console Window' is empty.</p>	The final value of the variable is updated, and all the outputs from the flowchart are displayed as shown.

Table H.4: Generate Source Code Storyboard

No.	Scene	Step
1.	 <p>The screenshot shows a 'Source Code' dialog with a 'Generate' button. A dropdown menu is open, showing 'C' and 'C++' options, with 'C++' selected.</p>	Select the C++ menu item or select it from the drop-down list and press “Generate”.
2.	 <p>The screenshot shows the 'Source Code' dialog with the following code:</p> <pre>#include <iostream> using namespace std; int main() { int a; a = 1 + 2; cout << a; return 0; }</pre> <p>Buttons for 'Copy' and 'Done' are visible at the bottom.</p>	A source code dialog pops up, and the C++ source code for the flowchart is displayed as shown.

APPENDIX I: Use Case Descriptions

Table I.1: “Create Project” Use Case Description

Use Case Name:	Create Project	ID:	1	Importance Level:	High	
Primary Actor:	Student	Use Case Type:				Detail, Essential
Stakeholders and Interests: Student – wants to create a new project.						
Brief Description: This use case describes about how a new project is created.						
Trigger: Student wants to create a new project. Type:						
Relationships: Association: Student Include: Extend: Generalization:						
Normal Flow of Events: <ol style="list-style-type: none"> 1. Student selects to create a new project. 2. System displays a directory dialogue box. 3. Student selects the directory to save the project in. 4. Student enters the name for the project. 5. Student confirms the project creation process. 6. System creates a new project and displays the default flowchart. 						
SubFlows: 1.1 Student saves the currently active project.						
Alternate/Exceptional Flows:						

Table I.2: “Load Project” Use Case Description

Use Case Name:	Load Project	ID:	2	Importance Level:	High	
Primary Actor:	Student	Use Case Type:				Detail, Essential
Stakeholders and Interests: Student – wants to load an existing project.						
Brief Description: This use case describes about how an existing project is loaded.						
Trigger: Student wants to load an existing project. Type:						

<p>Relationships:</p> <p>Association: Student</p> <p>Include:</p> <p>Extend:</p> <p>Generalization:</p>
<p>Normal Flow of Events:</p> <ol style="list-style-type: none"> 1. Student selects to open an existing project file. 2. System displays a directory dialogue box. 3. Student browses to the directory with the project file. 4. Student selects the project file to load. 5. Student confirms the project loading process. 6. System loads the existing project and displays the existing flowchart and variables.
<p>SubFlows:</p> <ol style="list-style-type: none"> 1.1 If there is an active project with unsaved changes, student saves the project.
<p>Alternate/Exceptional Flows:</p>

Table I.3: “Save Project” Use Case Description

Use Case Name: Save Project	ID: 3	Importance Level: High
Primary Actor: Student	Use Case Type: Detail, Essential	
Stakeholders and Interests: Student – wants to save a project.		
Brief Description: This use case describes about how a project is saved.		
Trigger: Student wants to save a project. Type:		
<p>Relationships:</p> <p>Association: Student</p> <p>Include:</p> <p>Extend:</p> <p>Generalization:</p>		
<p>Normal Flow of Events:</p> <ol style="list-style-type: none"> 1. Student selects to save the currently active project. 2. System displays a directory dialogue box. 3. Student selects the directory to save the project in. 4. Student enters the name for the project. 5. Student confirms the project saving process. 6. System saves the currently active project. 		
<p>SubFlows:</p> <ol style="list-style-type: none"> 2.1 If the currently active project is already associated with a project file, the project saving process completes immediately. The rest of the steps are skipped. 5.1 If saving on an existing file, the student confirms to save again. 		
Alternate/Exceptional Flows:		

Table I.4: “Execute Flowchart” Use Case Description

Use Case Name:	Execute Flowchart	ID:	4	Importance Level:	High
Primary Actor:	Student	Use Case Type:	Detail, Essential		
Stakeholders and Interests: Student – wants to execute the flowchart.					
Brief Description: This use case describes about how a flowchart is executed.					
Trigger: Student wants to execute the flowchart. Type:					
Relationships: Association: Student Include: Extend: Generalization:					
Precondition: A flowchart has already been created.					
Normal Flow of Events: <ol style="list-style-type: none"> 1. Student selects to execute the flowchart. 2. System executes the flowchart completely once. 3. The system displays the outputs produced and updates the variables’ values in the variable inspector. 					
SubFlows:					
Alternate/Exceptional Flows:					

Table I.5: “Print Flowchart” Use Case Description

Use Case Name:	Print Flowchart	ID:	5	Importance Level:	High
Primary Actor:	Student	Use Case Type:	Detail, Essential		
Stakeholders and Interests: Student – wants to print out the flowchart.					
Brief Description: This use case describes about how a flowchart is printed out.					
Trigger: Student wants to print the flowchart. Type:					
Relationships: Association: Student Include: Extend: Generalization:					
Precondition: A flowchart has already been created.					
Normal Flow of Events: <ol style="list-style-type: none"> 1. Student selects to print the flowchart. 2. System displays a printing dialogue box. 3. Student confirms to print the flowchart. 4. The flowchart is printed out from a printer in pdf format. 					
SubFlows: 3.1 Student modifies the printing configuration.					
Alternate/Exceptional Flows: 1.1 Student does not print the flowchart.					

Table I.6: “Add Symbol” Use Case Description

Use Case Name:	Add Symbol	ID:	6	Importance Level:	High
Primary Actor:	Student	Use Case Type:	Detail, Essential		
Stakeholders and Interests: Student – wants to add a new symbol to the flowchart.					
Brief Description: This use case describes about how a new symbol is added to the flowchart.					
Trigger: Student wants to add a new symbol to the flowchart. Type:					
Relationships: Association: Student Include: Drag-and-drop symbol Extend: Generalization:					
Precondition: A flowchart has already been created.					
Normal Flow of Events: 1. Student selects the types of flowchart symbol to add. 2. Student selects an area in the flowchart to add the flowchart to. 3. System displays the flowchart with the symbol added into the selected area.					
SubFlows:					
Alternate/Exceptional Flows: 1.1 Student does not add the symbol to the flowchart.					

Table I.7: “Edit Symbol” Use Case Description

Use Case Name:	Edit Symbol	ID:	7	Importance Level:	High
Primary Actor:	Student	Use Case Type:	Detail, Essential		
Stakeholders and Interests: Student – wants to edit a symbol that has been added to the flowchart.					
Brief Description: This use case describes about how a symbol that has been added to the flowchart is edited.					
Trigger: Student wants to edit a symbol that has been added to the flowchart. Type:					
Relationships: Association: Student Include: Extend: Generalization:					
Precondition: A flowchart has already been created and a symbol has been added to the flowchart.					
Normal Flow of Events: 1. Student selects a flowchart symbol to be edited. 2. System displays a statement dialogue box. 3. Student modifies the statement of the symbol. 4. Student confirms the changes made to the symbol. 5. System displays the symbol with the updated statement in the flowchart.					

SubFlows:
Alternate/Exceptional Flows: 3.1 Student does not change the statement of the symbol.

Table I.8: “Remove Symbol” Use Case Description

Use Case Name: Remove Symbol	ID: 8	Importance Level: High
Primary Actor: Student	Use Case Type: Detail, Essential	
Stakeholders and Interests: Student – wants to remove a symbol from the flowchart.		
Brief Description: This use case describes about how a symbol is removed from the flowchart.		
Trigger: Student wants to remove a symbol from the flowchart.		
Type:		
Relationships: Association: Student Include: Extend: Generalization:		
Precondition: A flowchart has already been created and a symbol has been added to the flowchart.		
Normal Flow of Events: 1. Student selects a flowchart symbol to be removed. 2. Student removes the flowchart symbol. 3. System displays the flowchart with the symbol removed.		
SubFlows:		
Alternate/Exceptional Flows: 2.1 Student retains the flowchart symbol.		

Table I.9: “Create Variable” Use Case Description

Use Case Name: Create Variable	ID: 9	Importance Level: High
Primary Actor: Student	Use Case Type: Detail, Essential	
Stakeholders and Interests: Student – wants to create a new variable.		
Brief Description: This use case describes about how a new variable is created.		
Trigger: Student wants to create a new variable.		
Type:		
Relationships: Association: Student Include: Declare Data Type Extend: Generalization:		
Precondition: A flowchart has already been created.		

Normal Flow of Events: <ol style="list-style-type: none"> 1. Student selects to create a new variable. 2. System displays a variable dialogue box. 3. Student fills in the details of the variable. 4. Student confirms to create the variable. 5. System displays the new created variable in the flowchart.
SubFlows: <ol style="list-style-type: none"> 2.1 Student declares the data type for the variable. 2.2 Student enters the name for the variable.
Alternate/Exceptional Flows: <ol style="list-style-type: none"> 2.1 Student does not create the variable.

Table I.10: “Edit Variable” Use Case Description

Use Case Name: Edit Variable	ID: 10	Importance Level: High
Primary Actor: Student	Use Case Type: Detail, Essential	
Stakeholders and Interests: Student – wants to edit an existing variable.		
Brief Description: This use case describes about how an existing variable is edited.		
Trigger: Student wants to edit an existing variable.		
Type:		
Relationships: Association: Student Include: Extend: Generalization:		
Precondition: A flowchart has already been created and a variable has been created.		
Normal Flow of Events: <ol style="list-style-type: none"> 1. Student selects the variable to be edited. 2. System displays the variable dialogue box. 3. Student modifies the details of the variable. 4. Student confirms changes made to the variable. 5. System displays the updated variable in the flowchart. 		
SubFlows: <ol style="list-style-type: none"> 2.1 Student changes the data type of the variable. 2.2 Student changes the name of the variable. 		
Alternate/Exceptional Flows: <ol style="list-style-type: none"> 2.1 Student does not modify the variable. 		

Table I.11: “Delete Variable” Use Case Description

Use Case Name: Delete Variable	ID: 11	Importance Level: High
Primary Actor: Student	Use Case Type: Detail, Essential	

Stakeholders and Interests: Student – wants to delete an existing variable.
Brief Description: This use case describes about how an existing variable is deleted.
Trigger: Student wants to delete an existing variable. Type:
Relationships: Association: Student Include: Extend: Generalization:
Precondition: A flowchart has already been created and a variable has been created.
Normal Flow of Events: 1. Student selects on the variable to be deleted. 2. Student confirms to delete the variable. 3. System deletes the variable from the flowchart.
SubFlows:
Alternate/Exceptional Flows: 2.1 Student does not delete the variable.

Table I.12: “Generate Source Code” Use Case Description

Use Case Name: Generate Source Code	ID: 12	Importance Level: High
Primary Actor: Student	Use Case Type: Detail, Essential	
Stakeholders and Interests: Student – wants to generate the source code for the flowchart.		
Brief Description: This use case describes about how the source code for the flowchart is generated.		
Trigger: Student wants to generate the source code for the flowchart. Type:		
Relationships: Association: Student Include: Extend: Generalization:		
Precondition: A flowchart has already been created.		
Normal Flow of Events: 1. Student selects to generate the source code for the flowchart. 2. Student confirms to generate the source code for the flowchart. 3. System generates the source code for the flowchart in a text file. 4. System displays a confirmation to indicate the source code has been generated.		
SubFlows:		
Alternate/Exceptional Flows:		

Table I.13: “Undo Action” Use Case Description

Use Case Name: Undo Action	ID: 13	Importance Level: High
Primary Actor: Student	Use Case Type: Detail, Essential	
Stakeholders and Interests: Student – wants to undo an action.		
Brief Description: This use case describes about how an action is undone.		
Trigger: Student wants to undo an action. Type:		
Relationships: Association: Student Include: Extend: Generalization:		
Precondition: A flowchart has already been created.		
Normal Flow of Events: 1. Student undoes an action. 2. System reverts the flowchart back to before the action is produced.		
SubFlows:		
Alternate/Exceptional Flows: 1.1 Student does not undo the action.		

Table I.14: “Redo Action” Use Case Description

Use Case Name: Redo Action	ID: 14	Importance Level: High
Primary Actor: Student	Use Case Type: Detail, Essential	
Stakeholders and Interests: Student – wants to redo an action.		
Brief Description: This use case describes about how an action is redone.		
Trigger: Student wants to redo an action. Type:		
Relationships: Association: Student Include: Extend: Generalization:		
Precondition: A flowchart has already been created.		
Normal Flow of Events: 1. Student redoes an action. 2. System reverts the flowchart back to after the action is produced.		
SubFlows:		
Alternate/Exceptional Flows: 1.1 Student does not redo the action.		

APPENDIX J: Class Diagrams

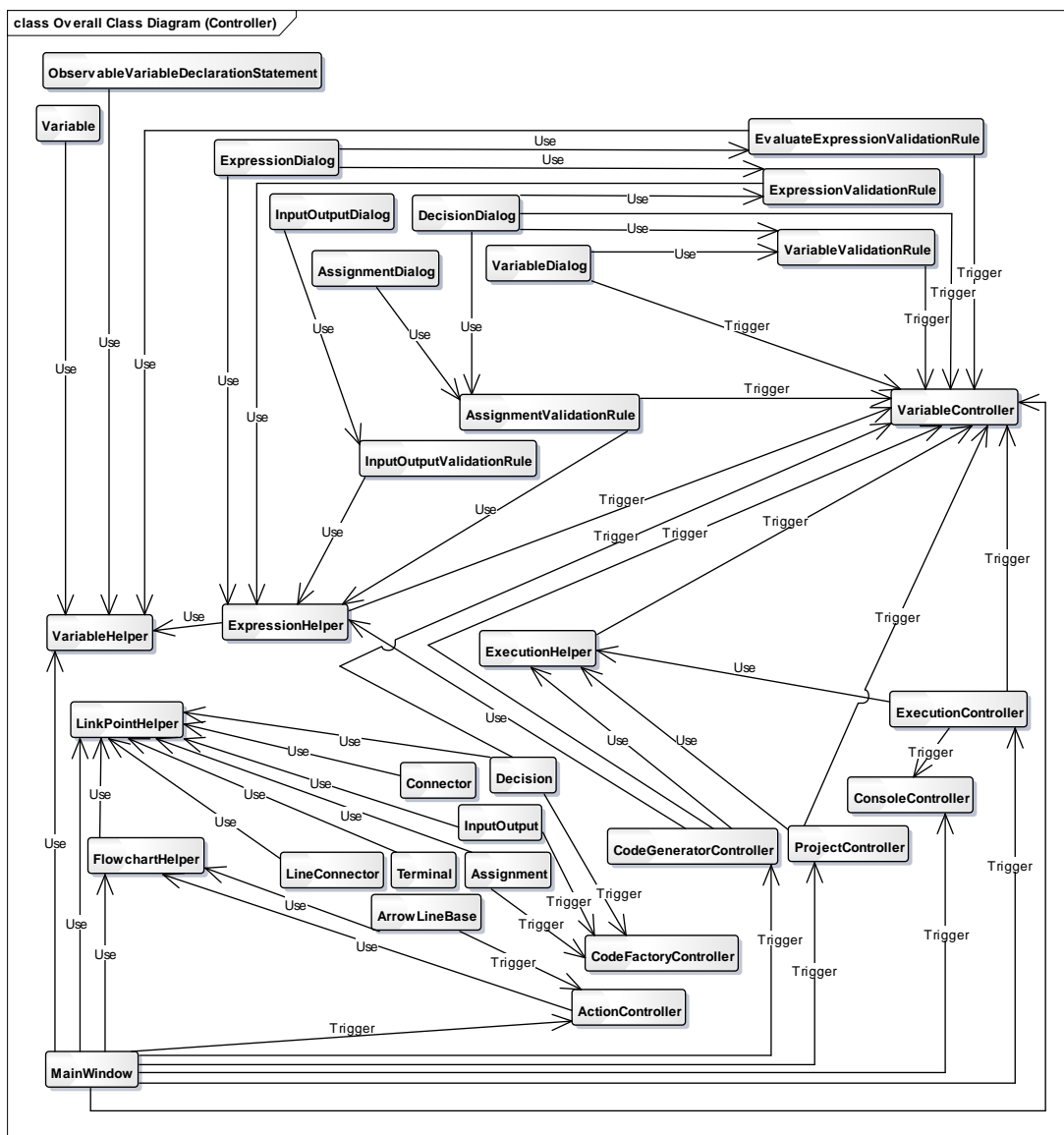


Figure J.1: Overall Class Diagram for Relationship with Controller Classes

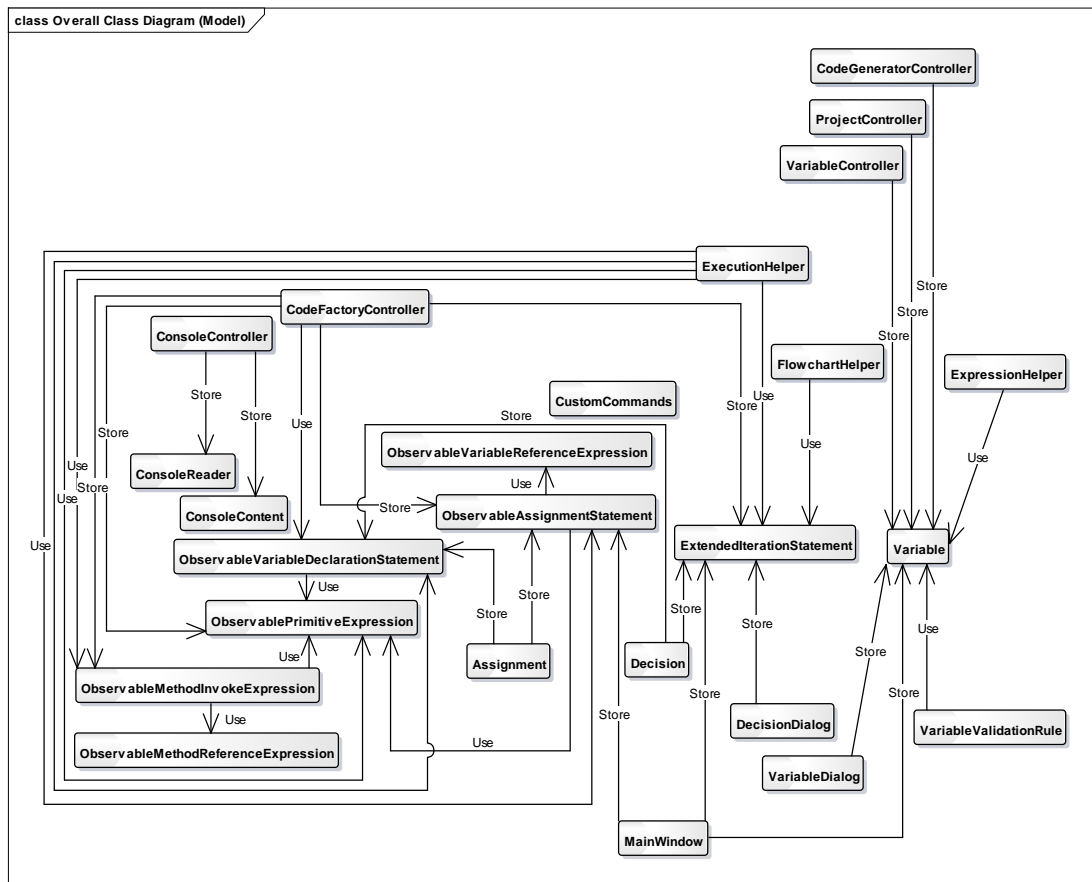


Figure J.2: Overall Class Diagram for Relationship with Model Classes

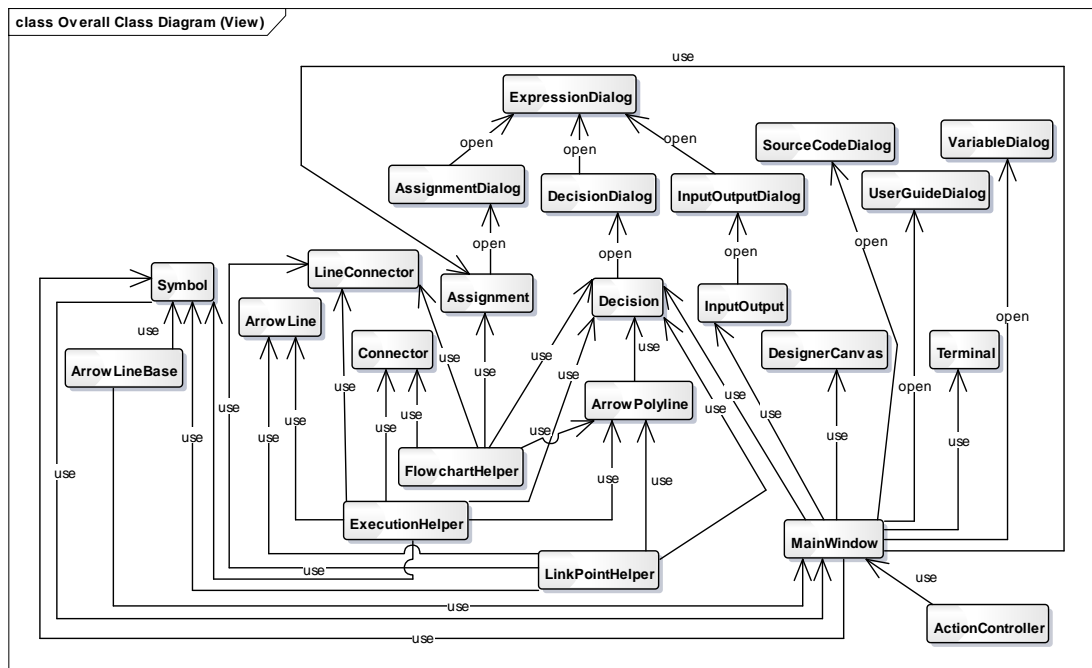


Figure J.3: Overall Class Diagram for Relationship with View Classes

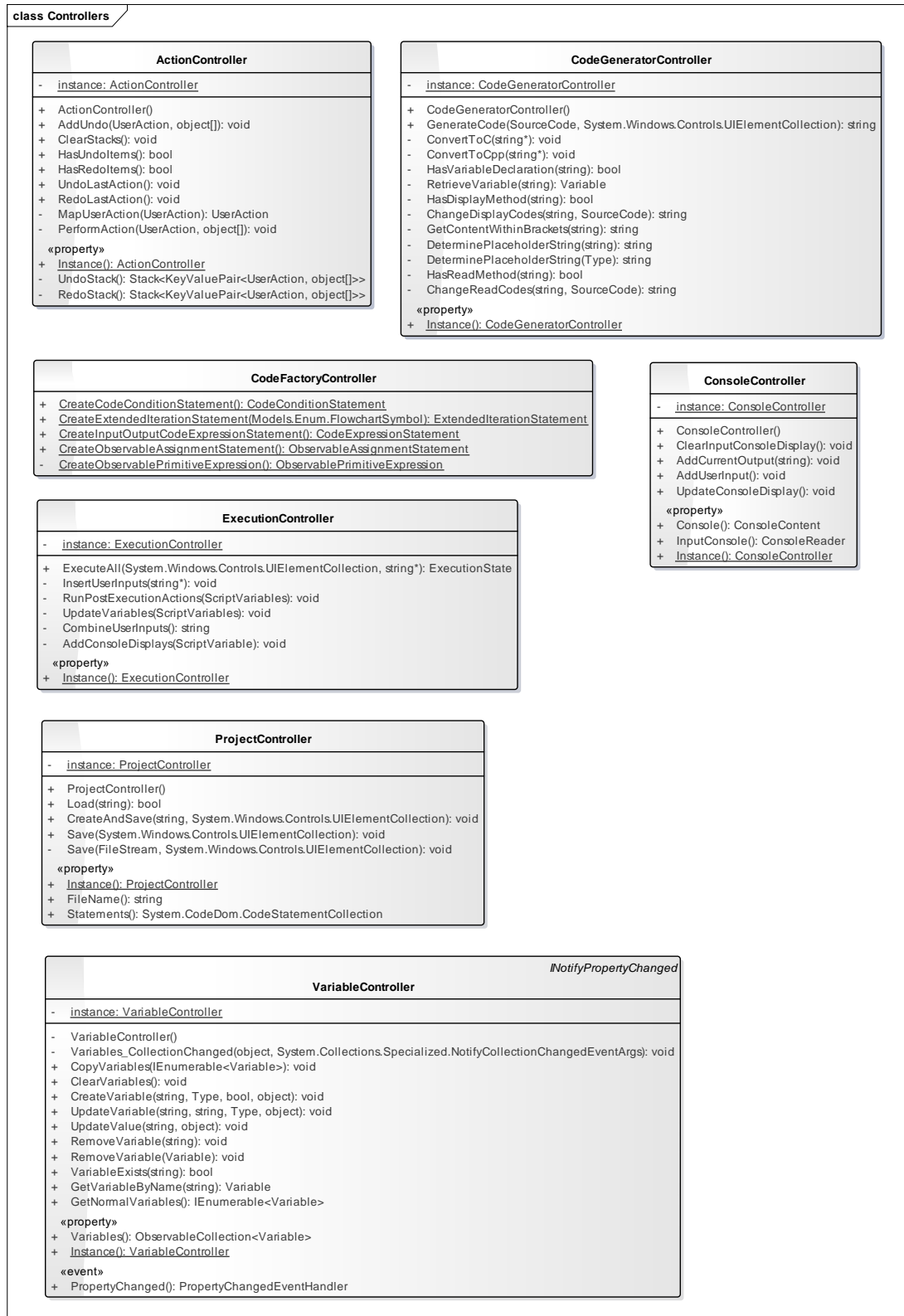


Figure J.4: Controller Classes

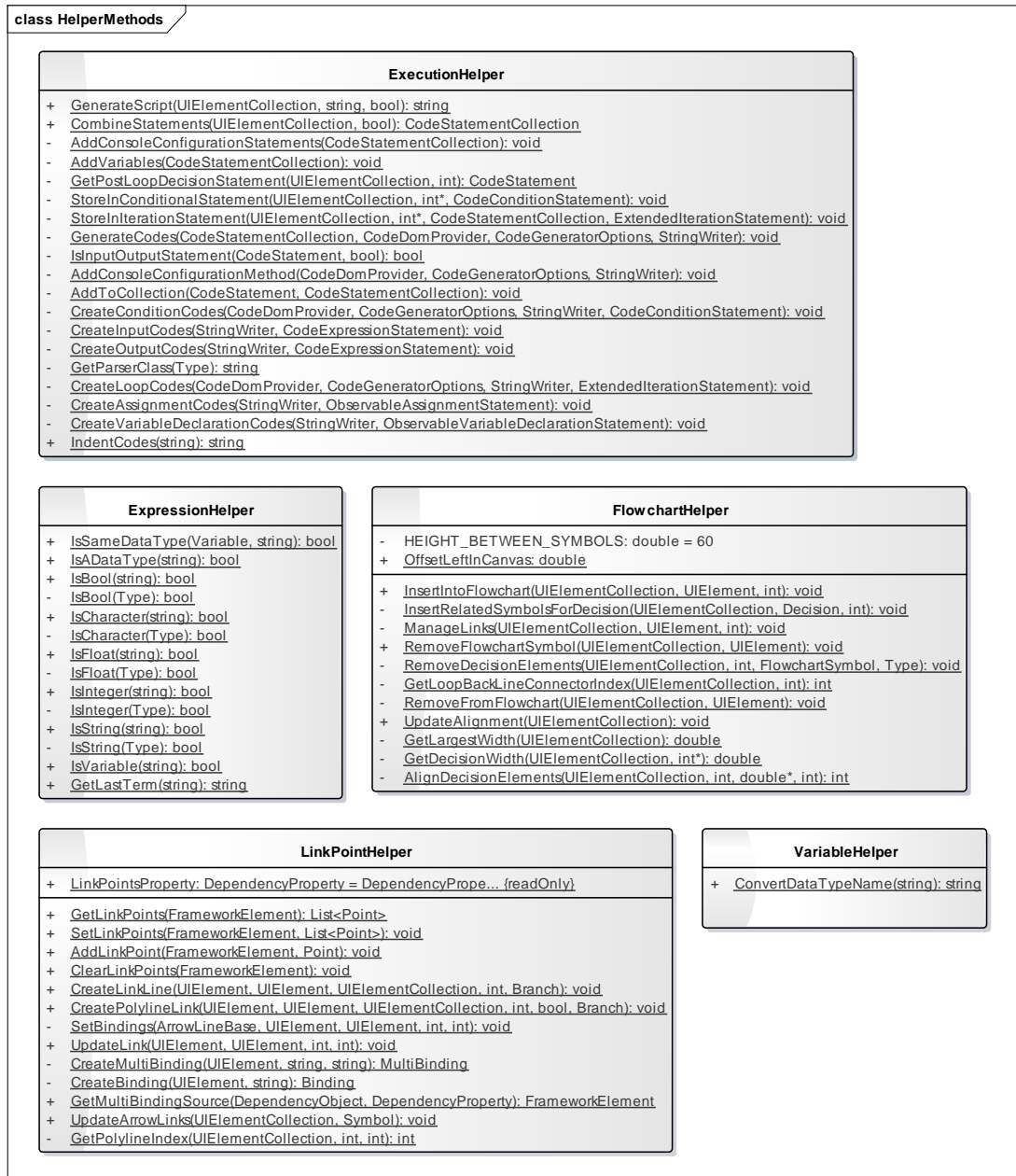


Figure J.5: Helper Classes

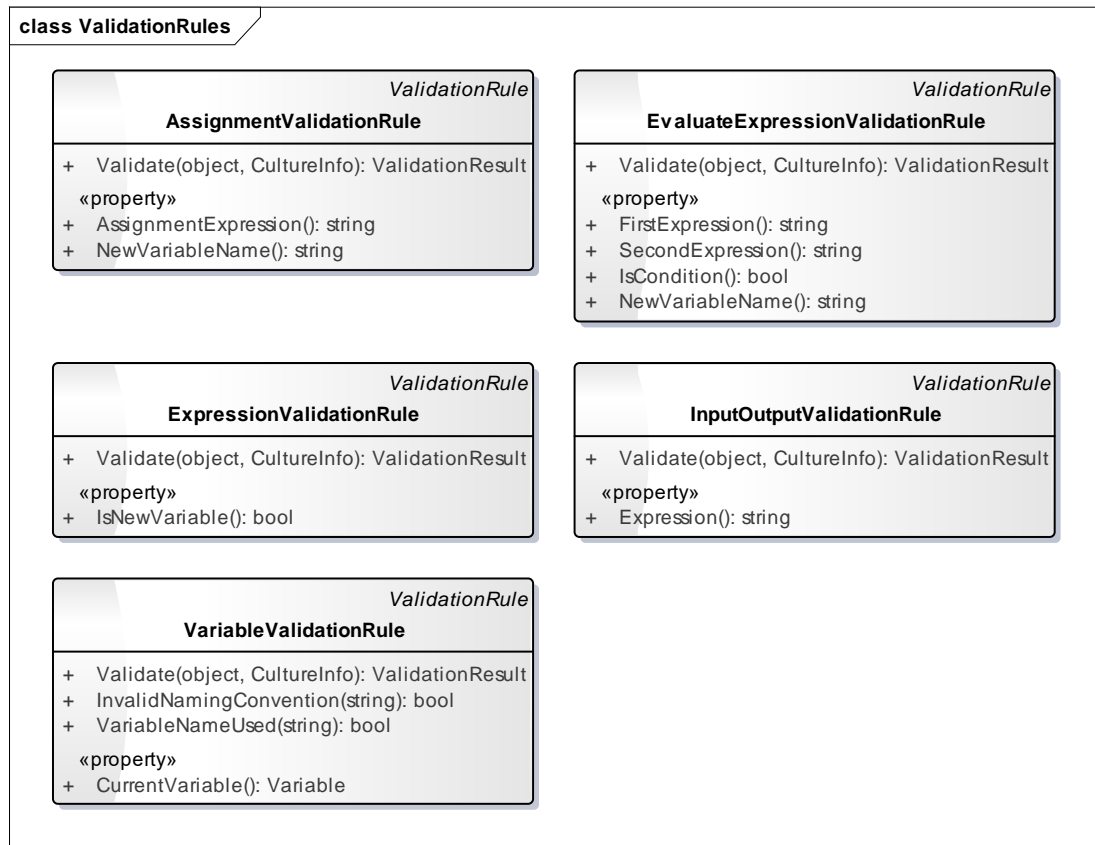


Figure J.6: Validation Rule Classes

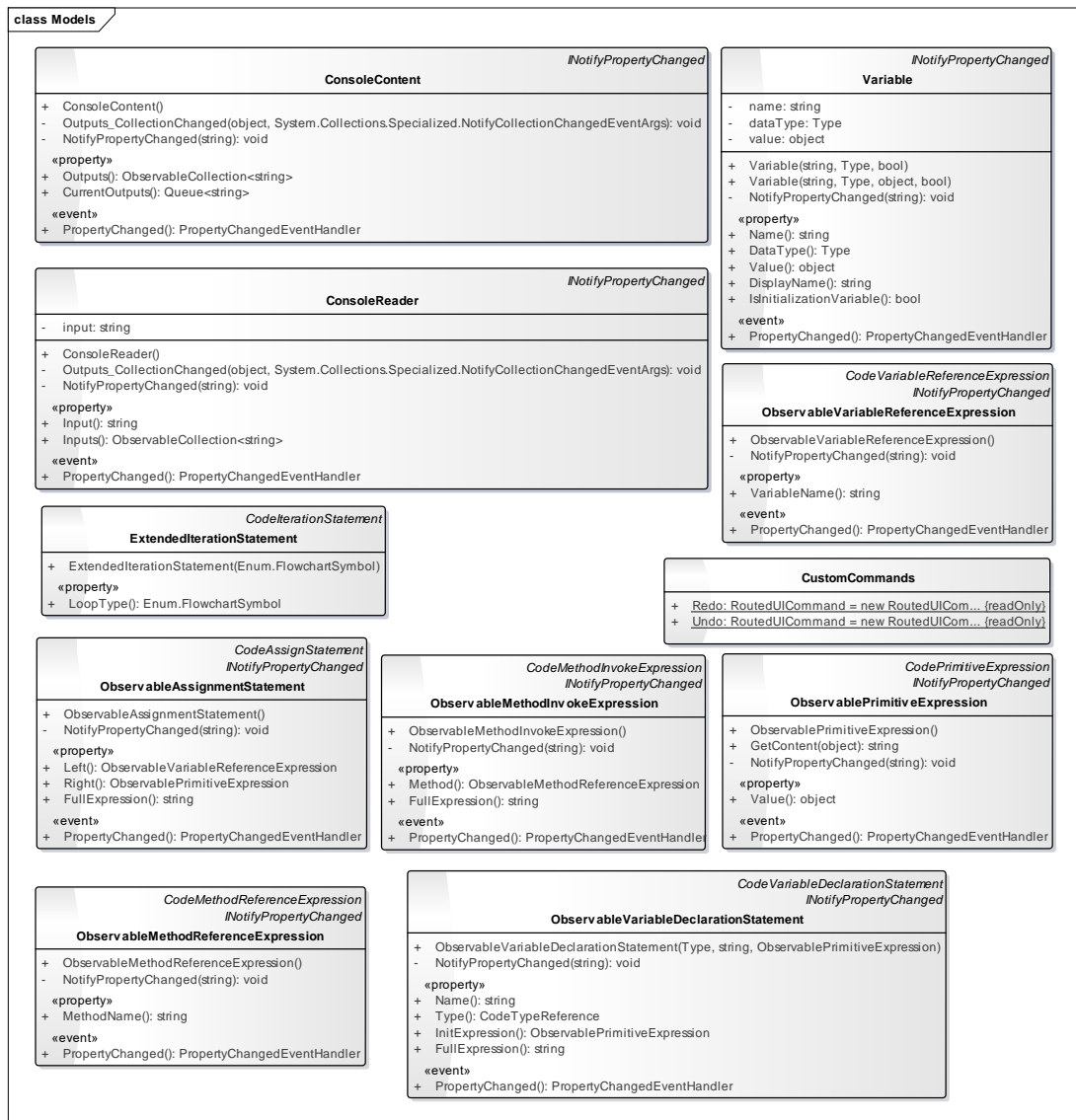


Figure J.7: Model Classes



Figure J.8: MainWindow Class

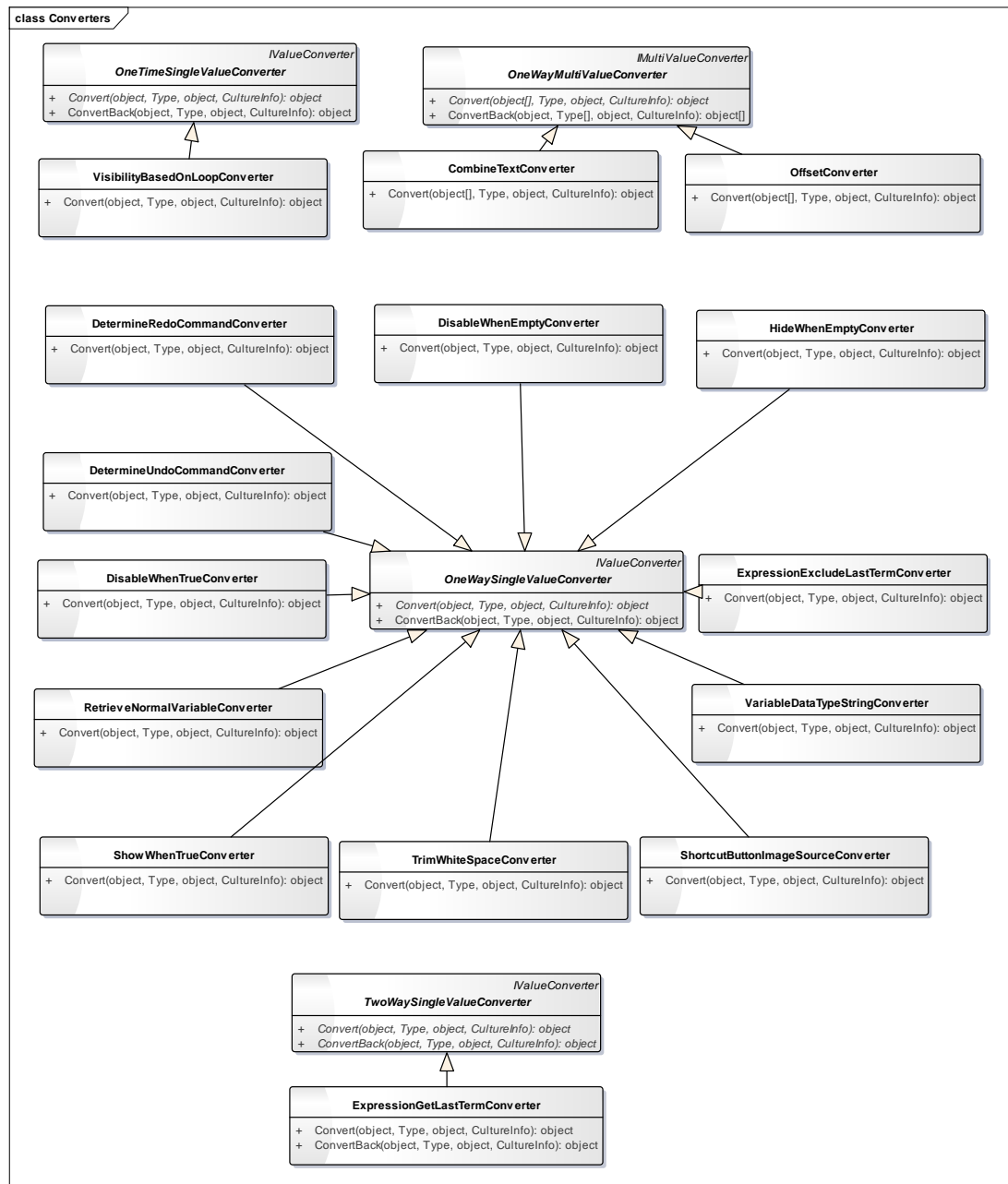


Figure J.9: Converter Classes

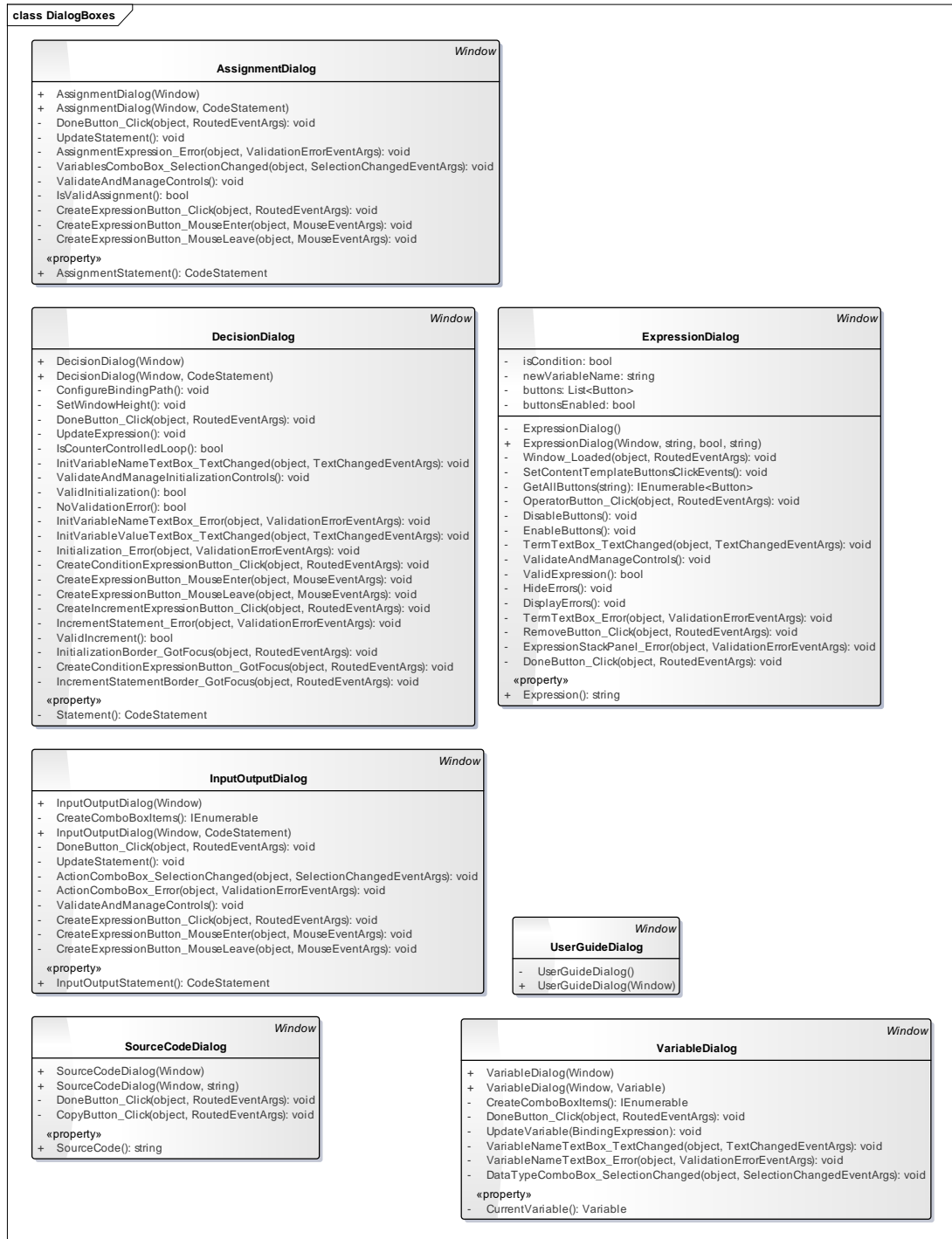


Figure J.10: Dialog Classes

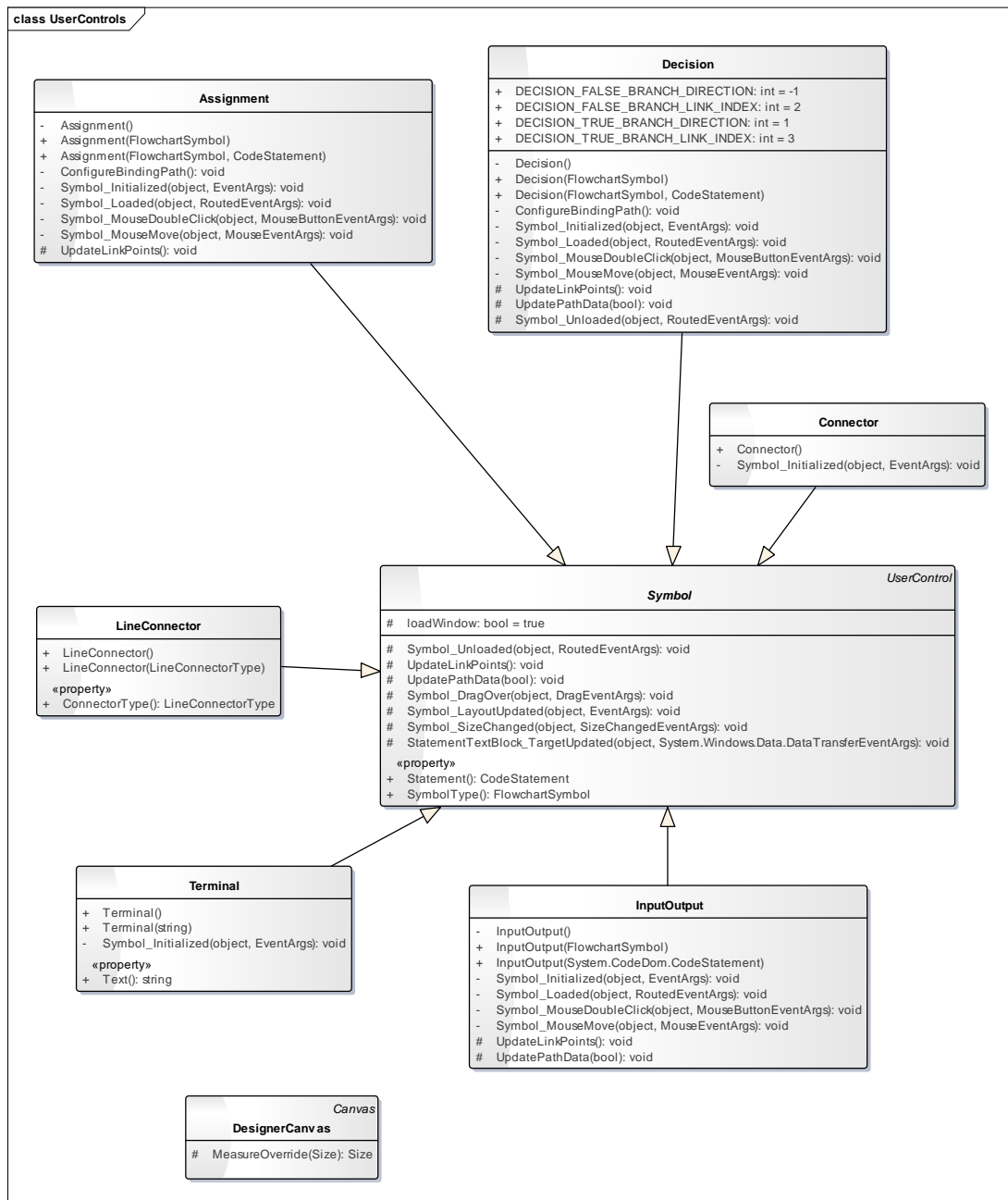


Figure J.11: Symbol User Controls Classes

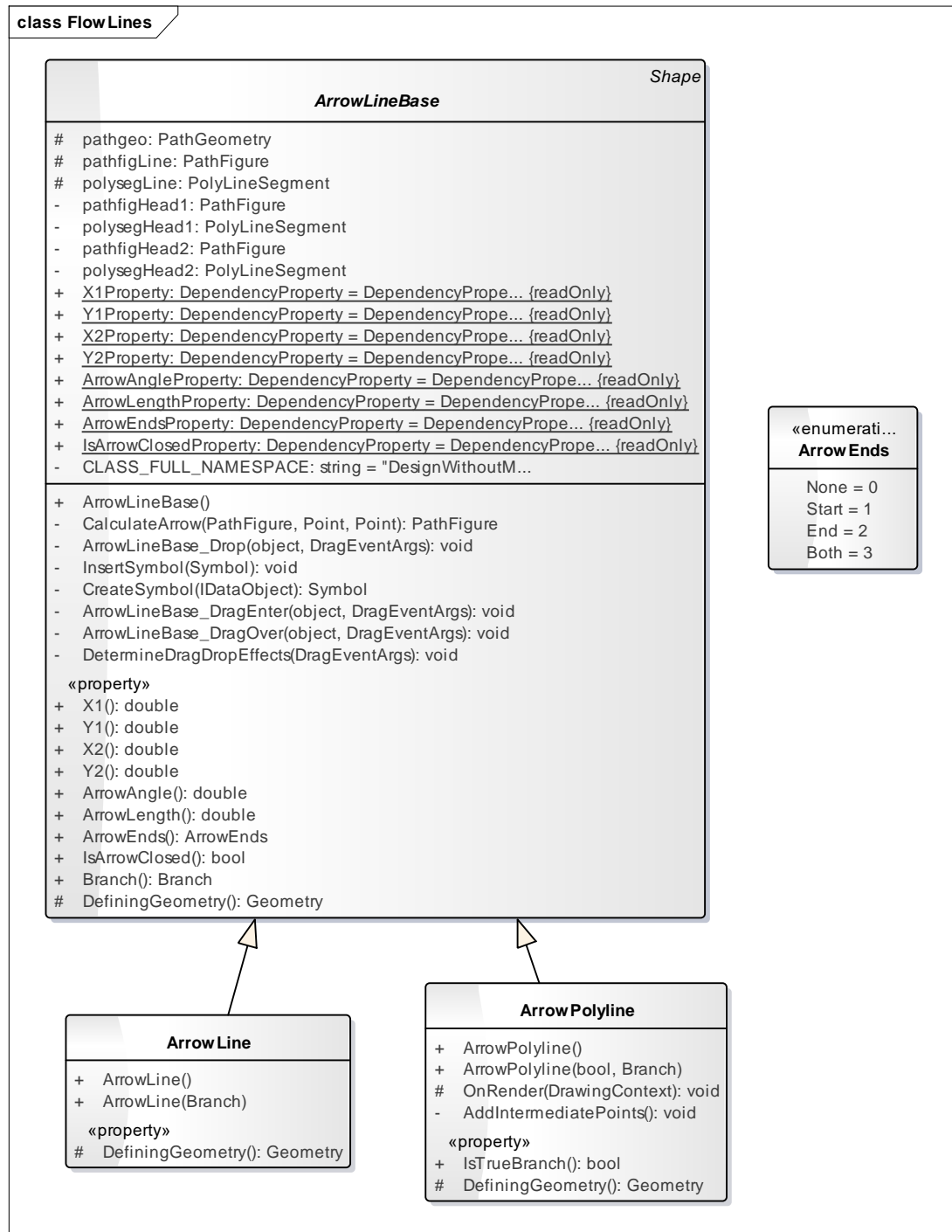


Figure J.12: Arrow Classes

APPENDIX K: Unit Tests

Table K.1: Project Management Unit Tests

Unit Test	Pass	Fail
TestCreateNewProject		
TestSaveProjectWithEmptyString		
TestSaveProjectWithFileName		
TestLoadProjectWithEmptyString		
TestLoadInexistentProject		
TestLoadProjectWithExistingFile		

Table K.2: Variable Management Unit Tests

Unit Test	Pass	Fail
TestCreateBoolVariable		
TestCreateCharVariable		
TestCreateFloatVariable		
TestCreateIntegerVariable		
TestCreateStringVariable		
TestAddVariable		
TestRetrieveVariableExist		
TestRetrieveVariableNotExist		
TestEditVariablePass		
TestEditVariableFail		
TestRemoveVariableByName		
TestRemoveVariableByInstance		

Table K.3: Flowchart Execution Unit Tests

Unit Test	Pass	Fail
TestExecuteAllWithAssignmentWithoutVariableDeclared		
TestExecuteAllWithValidAssignment		
TestExecuteAllWithInvalidAssignment		
TestExecuteAllWithDisplay		

Table K.4: Statement Management Unit Tests

Unit Test	Pass	Fail
TestAddAssignmentStatement		
TestAddInputOutputStatement		
TestAddConditionStatement		
TestAddForLoopStatement		
TestAddWhileLoopStatement		
TestAddDoWhileLoopStatement		
TestInsertAssignmentStatement		
TestInsertInputOutputStatement		
TestInsertConditonStatement		
TestInsertForLoopStatement		
TestInsertWhileLoopStatement		
TestInsertDoWhileLoopStatement		
TestRemoveAssignmentStatement		
TestRemoveInputOutputStatement		
TestRemoveConditionStatement		
TestRemoveForLoopStatement		
TestRemoveWhileLoopStatement		
TestRemoveDoWhileLoopStatement		

Table K.5: Source Code Generation Unit Tests

Unit Test	Pass	Fail
TestGenerateCCodeVariableDeclaration		

TestGenerateCppCodeVariableDeclaration		
TestGenerateCSharpCodeVariableDeclaration		
TestGenerateCCodeAssignmentStatement		
TestGenerateCppCodeAssignmentStatement		
TestGenerateCSharpCodeAssignmentStatement		
TestGenerateCCodeInputStatement		
TestGenerateCppCodeInputStatement		
TestGenerateCSharpCodeInputStatement		
TestGenerateCCodeOutputStatement		
TestGenerateCppCodeOutputStatement		
TestGenerateCSharpCodeOutputStatement		
TestGenerateCCodeConditionStatement		
TestGenerateCppCodeConditionStatement		
TestGenerateCSharpCodeConditionStatement		
TestGenerateCCodeForLoop		
TestGenerateCppCodeForLoop		
TestGenerateCSharpCodeForLoop		
TestGenerateCCodeWhileLoop		
TestGenerateCppCodeWhileLoop		
TestGenerateCSharpCodeWhileLoop		
TestGenerateCCodeDoWhileLoop		
TestGenerateCppCodeDoWhileLoop		
TestGenerateCSharpCodeDoWhileLoop		

Table K.6: Undo Redo Action Unit Tests

Unit Test	Pass	Fail
TestUndoAddAssignmentStatement		
TestRedoAddAssignmentStatement		
TestUndoAddInputOutputStatement		
TestRedoAddInputOutputStatement		
TestUndoAddConditionStatement		
TestRedoAddConditionStatement		

TestUndoAddForLoopStatement		
TestRedoAddForLoopStatement		
TestUndoAddWhileLoopStatement		
TestRedoAddWhileLoopStatement		
TestUndoAddDoWhileLoopStatement		
TestRedoAddDoWhileLoopStatement		
TestUndoRemoveAssignmentStatement		
TestRedoRemoveAssignmentStatement		
TestUndoRemoveInputOutputStatement		
TestRedoRemoveInputOutputStatement		
TestUndoRemoveConditionStatement		
TestRedoRemoveConditionStatement		
TestUndoRemoveForLoopStatement		
TestRedoRemoveForLoopStatement		
TestUndoRemoveWhileLoopStatement		
TestRedoRemoveWhileLoopStatement		
TestUndoRemoveDoWhileLoopStatement		
TestRedoRemoveDoWhileLoopStatement		
TestUndoAddVariable		
TestRedoAddVariable		
TestUndoDeleteVariable		
TestRedoDeleteVariable		