VisualPro - Visual Scripting Technologies for the Tomorrow

1st Given Edward Patch
Software Engineer Student (of BSc Year 3)
Dissertation
University of Wales Trinity St. Davids (of Mike Dacey)
Swansea, Wales
Student ID: 1801492

CONTENTS

***Abstract*—**
***Index Terms*—Visual Programming, Visual Scripting, Development**

I. INTRODUCTION

II. LITERATURE REVIEW

A. User Interface and User Experience

Research for User Interface (UI) and User Experience (UX) analyse how a new developer would use the software with little to no coding experience and what difficulties the UI and UX design may encounter. Methods of outlining and testing the UI and UX are within the Design and Feedback III section under subsection, Research Methodology III-A, page 3.

1) User Interface: To get some ideas for the user interface, the Journal 'Evaluation of a UML-Based Versus an IEC 61131-3-Based Software Engineering Approach for Teaching PLC Programming' writting by Birgit Vogel-Heuser, Martin Obermeier, Steven Braun, Kerstin Sommer, Fabian Jobst, and Karin Schweizer [1] and the Journal 'Design Issues and First Experiences with a Visual Database Editor for the extended NF^2-Data Model', author(s):- K. K*ü*spert, J. Teuhola and L. Wegner [2]. These Journals cover Unified Modeling Language (UML). The UML concept is promdomity used in relationship diagrams in databases and Visual Scripting.

The addition of UML tools for the frontend to create the Visual Scripting platform proves an excellent opportunity as there is plenty of UML Framework support. Some critical

points based on an experiment from Birgit Vogel-Heuser's Journal Article [1] are listed.

Experiment Hypothesis

To gather the answer, 'How effective is UML tools?', the hypothesis from Birgit's [1] Journal article, "Students trained in OO modelling show an imporved modelling performance.".

Experiment Environment

The experiment environment involves the following rules:-

- Software Engineering theme. - Any material created for the experiment to work, the experiment outline is set in a Software Engineering equivalent environment.
- Training Assets and Problem-Solving Exercises. - A series of training materials and problem-solving exercises were created for the experiment to be fair.
- Task Setting and Training. - The author focuses on Task Setting and Training, supported by Hierarchical Task Analysis.

Experiment Results

"The data were analyzed using three methods. First, an analysis of variance (ANOVA) was applied to test differences between the variances of several groups in order to show whether their performance changed after the training approaches and whether their performance differed between classes due to expertise level differences. Then, correlations were computed as a measure of the relation between programming/modeling performances on different performance scales. Subsequently, differences in relations between the two software engineering approaches (see hypothesis 3) were shown by computing regression models for the programming performance for both approaches." - Birgit Vogel-Heuser [1]. To answer Hypothesis 1 set by Birgit Vogel-Heuser [1] '...analysis examined the within factor (before and after the training) and additionally two between factors (notation and expertise)."

The results gathered display that the training set for learning UML in a Software Engineering sense was highly effective. However, the Birgit Vogel-Heuser [1] states that 'All participants learned from the training.' and backs it up by '...indicating that the -group had learned even more from the training because of their poor results before the training (see also Fig. 2) and their lack of prior knowledge.' This quotation suggests that UML tools found in Visual Scripting may not necessarily be complicated for beginners to learn, but they may seem more complex without proper training. This conclusion could suggest that UML methods are not well documented and could indicate that UML may not be well maintained, look at section II-A2, page 2.

2) Maintenance: The Journal by Erik Arisholm, Lionel C. Briand, Siw Elisabeth Hove and Yvan Labiche, 'The Impact of UML Documentation on Software Maintenance: An Experimental Evaluation', addresses the lack of UML Documentation on both developer-side and client-side. After learning that the individuals who participated in the experiment from section II-A1, page 1, the question arose, 'Was the UML documentation depreciated and how was the documentation withstanding on the development side?' it was vital to determine if UML tools are future-proof or will it prove expensive to maintain. After Erik Arisholm performed tests found in the Journal, the author found that 'UML documentation does not seem to provide an advantage when considering the additional time needed to modify models.' Furthermore, disregarding existing UML documentation seemed to correlate compared to the first test, thus suggesting that the UML documentation had the same effect as no UML documentation.

Things to note during this study, if the integration of UML tools within VisualPro, then expenses regarding finance, staff and resources for UML documentation for both developers and clients are necessary. However, the creation and maintenance of the software would require UML Development documentation. Yet, reverse engineering the UML libraries to create a detailed UML Development and UML Usability Documentation for both users before any development can begin on the software. This event is due to understanding how the UML tools operate and integrating the software properly to work in the future.

3) User Experience: **Difficulties of Teaching**

Within the Journal, written by Birgit Vogel-Heuser, Martin Obermeier, Steven Braun, Kerstin Sommer, Fabian Jobst, and Karin Schweizer [1], 'Evaluation of a UML-Based Versus an IEC 61131-3-Based Software Engineering Approach for Teaching PLC Programming', an experiment to test whether beginners excelled at object-orientated programming (OOP) or functional programming (FP). The quote follows:-

"Berges and Hubwieser investigated Computer Science freshmen's abilities to learn OO programming in two and a half days with as little (human) instruction as possible [27]. Examining 300 students' program code, they found that most were able to write quite satisfying programs. They identified two types of students: those who accept and apply the OO concepts, and those who prefer to program in a more traditional procedural way. They also tried to define the characteristics of object orientation to evaluate measures for program quality, e.g., one instance of a class is created." [1]

This test suggests that after three hundred students were tested with little instruction from other peers, which opens an interesting fact that most of the three hundred students wrote acceptable code in both OOP and FP categories. The test's conclusion finds two types of students, and the students either found OOP or FP easier. After analysing this specific test, beginners may have different mindsets and thinking styles.

B. Productivity and Visual Scripting

Productivity, specifically within the software engineering field, Dr. Caitlin Sadowski's [3], 'Rethinking Productivity in Software Engineering' Book states, "Productivity is a challenging concept to define, describe, and measure for any knowledge work that involves nonroutine creative tasks. Software development is a prime example of knowledge work, as it too often involves poorly defined tasks relying on extensive collaborative and creative endeavors. As in other areas of knowledge work, defining productivity in software development has been a challenge facing both researchers and

practitioners who may want to understand and improve it by introducing new tools or processes." When reading further into the chapter, the following points evaluate how we should think about productivity:-

- "Velocity: How fast works gets done."
- "Quality: How well works gets done."
- "Satisfaction: How satisfying the work is."

After thinking about this, VisualPro, in theory, should increase the rapidity of the individual's work. Suppose VisualPro changes the way Visual Scripting works today, making it lightweight and understandable compared to existing Visual Scripting (For example, Unreal Engine's Blueprints [4]), with the ability to generate any desired code. In that case, the quality and satisfaction of the developer amplify.

III. Design and Feedback

The Design and Feedback section covers the Research Methodology and the Pre-Planning, which shows the steps and the level of thought that went into deciding VisualPro's purpose and the existing software and prototypes. Some Notebook Planning may not make sense as some foreseeable problems were thought and planned ahead of time.

A. Research Methodology

1) Learning Tools: Research of obstacles and potentials help identify existing and future issues that VisualPro software could generate. The aim is for amateurs new to the development field and suggests existing problems that put a 'brick wall' in learning programming and potentials of how VisualPro can solve these issues. This topic may reflect on the critical components of UI and UX previously mentioned.
Obstacles
Potentials

2) User Interface Methods: **Unicode Markup Language** Whilst observing the book, 'Using UML: software engineering with objects and components, by Pooley R J and Perdita Stevens[5], it is important to think about why UML may make a good UI method. UML intends to meet the criteria of allowing Software Engineers to design a software platform and demonstrate it to their clients. At the same time, it gives a comprehensive plan instead of focusing on Pesudeo code, supported by the quotes, Pooley R J [5] '...help to resolve misunderstandings' and '...developer's effort can be spent on work that requires their skills, not on routine work such as making a diagram using a drawing tool;'.

3) Node Tree: Another method of creating the UI is a Node Tree that displays the objects in containers and links them together with arrows to display the object's relationship. A Visual Database Editor diagram in figure 1 illustrated by K. K*ü*spert [2]. The node tree, in this case, is to construct a database structure design. Although, this could prove an easy-to-read method for Visual Scripting. Perhaps a mix of UML and Node Tree's could benefit from displaying relationships, for example:- *Parents* $\rightarrow$ *Children*. In theory, VisualPro could potentially change Visual Scripting as it is today and demonstrate the power of the C++ Dynamic-Link Library (DLL), Programming Planner library can do.

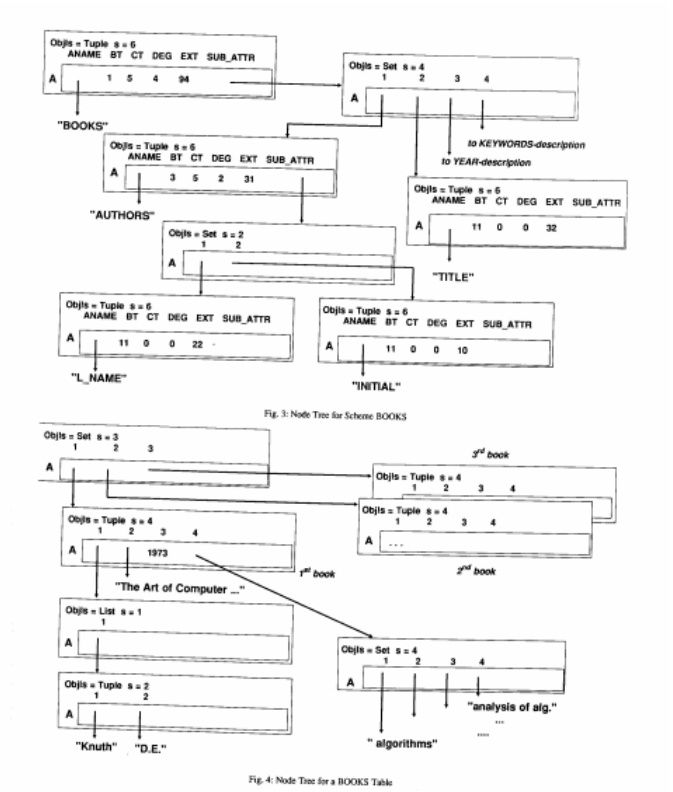

Fig. 1. Visual Database Editor Diagram [2]

4) Methods of Final Testing: **Research of Libraries** From GitHub Repositories, a list of Visual Scripting libraries are listed:-

- Language: GDScript - Author: Swarnimarun - Respository: Visual Scripting Node Library.
- Language: Python - Author: leon-thomm - Respository: Ryven.
- Language: C# - Unity Game Engine - Author ConstellationLanguage - Respository: Constellation.
- Language: Web Scripting - Author: ericabouaf - Respository: webhookit.
- Language: Web Scripting - Author: WebCabin - Respository: wcPlay.

These Visual Scripting software listed are identical to standard Game Engine Visual Scripting logic. Even though some are easy to use in the demos, a commonality is that they only allow users to control a live environment, whether it is a built-in game engine or an addition to an existing Game Engine, and they are mainly game orientated. VisualPro aims to support different platforms, such as Web, Mobile, Software, Data Analytics and Games. This aim is possible due to the diversity of users, adding new languages by entering the desired language within the XML document; though,

it requires documentation to support users in adding new languages for the program to work correctly.

Visual Scripters and Web Builders

A table of a few Web Builders and Visual Scripters demonstrates the difficulty of different platforms. Visual Scripting Software I:-

TABLE I
COMPARISON OF VISUAL SCRIPTERS

Name	Description	Difficulty: Hard (0-10) Easy	Explanation?
Unreal Engine 5 [4]	Game Engine (C++ Language)	Beginner: 3-0 — Experience: 5-10	A beginner who does not understand the principles of code may struggle over time.
Unity Engine [6]	Game Engine (C# Language)	Beginner: 2 - 6 — Experience: 8-10	With the documentation available and the simplicity of the Unity Visual Scripting design, it seems beginner-friendly.
Minecraft [7]	Game (Redstone) Represents binary coding.	Beginner: 7-10 — Experience: 10	Even though Minecraft does not create programming languages, it shows that many ages who enjoy logging onto Minecraft to make Redstone functionality, passively learn about Binary code.

After studying different Visual Scripting software, there is no portable Visual Scripting made for versatility. Both seem to only aim at Game Engines. Minecraft does not offer the ability for Visual Scripting. However, to bring attention to the Redstone feature in Minecraft, the feature is interesting for the audience, even to none-programmers. Could the VisualPro library be made into a Visual Scripting game, or specific in-game blueprints/prefabs be placed into the environment, generating code, so that the user can create programs whilst maintaining interest? Redstone accomplishes Binary teaching with Redstone passively by using on and off instead of 1s and 0s.

Visual Scripting, Unreal Engine has a 'messy' look, especially when the game logic is more clunky and complex, whereas Unity seems to have a better order. The problem with Visual Scripting in both engines is that the performance of the code is slow compared to writing the code manually. Would this uplift the software market sections if the VisualPro Scripting Pad software could generate code in the desired language with the most effective running time?

TABLE II
COMPARISON OF WEB BUILDERS

Name	Description	Difficulty: Hard (0-10) Easy	Explanation?
WordPress	Web Builder	Beginner: 8-10 — Experience: 9-10	WordPress offers many themes and plugins to the user base.
Wix	Web Builder	Beginner: 9-10 — Experience: 10	Wix is designed to help buisnesses to design webpages with no skill required.
Bootstrap Studio	Web Builder	Beginner: 5-7 — Experience: 7-10	Bootstrap Studio is a similar interface to Adobe Dreamweaver. It offers templates and allowed websites to designed. This interface is difficult with no experience with Bootstrap.

In one form, Web Builder's give an idea of what makes Visual Scripting easier. This concept is down to the research, design, and implementation of existing developers who had to look into individuals with no coding experience. If there is a correlation between designing a more straightforward Visual Scripting interface using methods that Web Builders introduced, it could update Visual Scripting and bring it into future software design. Perhaps, it could be a tool to visualise JavaScript so that users can build a website and add functionality, or create code snippets, for example, MelPy to animate 3D models within Autodesk Maya and other products, making Artists thrive more rather than relying on a Software Engineer to make their vision.

B. Pre-Planning

1) HTML Design: Figure 2 shows a HyperText Markup Language (HTML) Design of how the User Interface (UI) should look. The Empty Column iyou're like always battling with flu or something ¿p should create a new container, giving the user acknowledgement there's a new parent or sub-element with room for configuration and another drag and drop container within the class or function.

Fig. 2. HTML Design - Found at: Original Image

2) Notebook Page 1-2: Figure 3 contains numerous ideas. Page 1 contains a heading, 'Personal Assistant', which indicates the thought process of a personal assistant before the Visual Programming Scripting program progressed. After planning this, it became apparent to use a type of Artificial Intelligence (AI), which knew the basic syntax of different languages and would write or advise the user when typing. This process is very similar to TabNine (ref here).

The second page of this figure, shows a few different examples of IF statement syntax from languages to find a 'dynamic' common rather than keep it static like the previous software. The sketch below shows a name for the software and a example of how the language selection would look on UI.

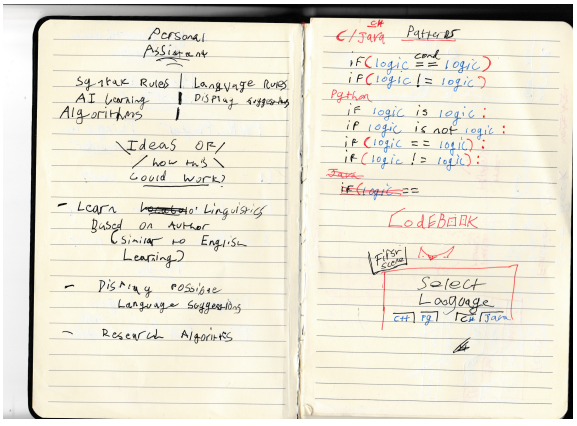

Fig. 3. Notebook Plan Page 1-2 - Found at: Original Image

3) Notebook Page 2-3: Figure 4 displays two pages of how the Visual Scripting could look. The top page shows the tools on the left, such as Logical, Structure and Scoping tools. On the right, it should show a list of current variables, functions and classes in a hierarchy style. The middle of the application is the work area.

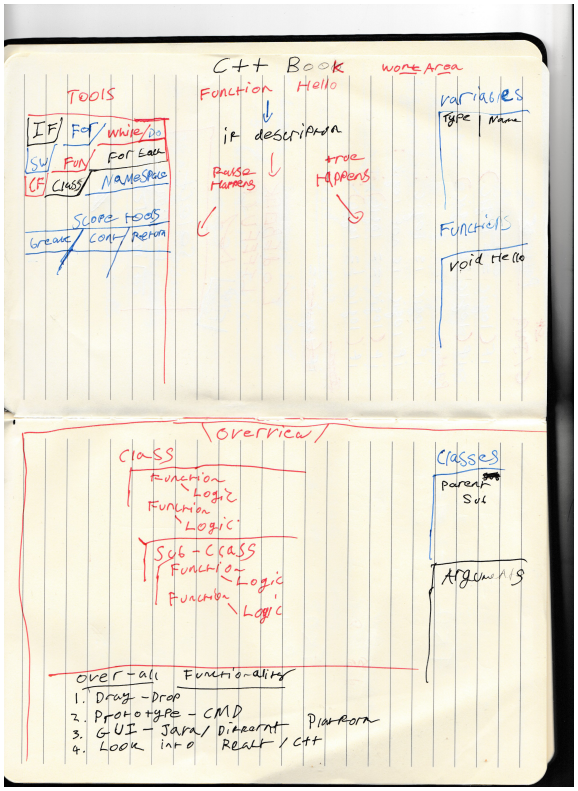

Fig. 4. Notebook Plan Page 3-4 - Found at: Original Image

4) Notebook Page 5-6: Figure 5 gives the planning of how the syntax for most programming and scripting languages. The plan points out patterns and tries to identify their names for the Extensible Markup Language (XML) and backend to comprehend. This piece of planning helped excel the development of the program's dynamic side to allow the addition of new languages to the software.

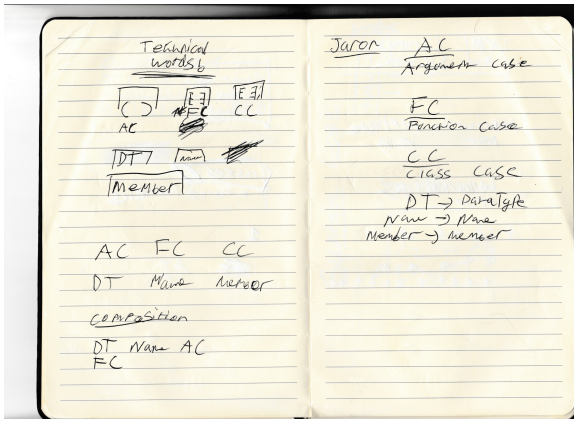

Fig. 5. Notebook Plan Page 5-6 - Found at: Original Image

5) Notebook Page 7-8: Figure 6 demonstrates how the XML would possibly look. The table shows N (Node), A (Attribute) and the Meaning. To explain what is going on, Node —0— is the header node (holds library data such as **system** or **iostream**), holds three attributes. Attribute —0— holds 'Type', which is the purpose, Attribute —1— holds the 'Name', which is the library's name. Attribute —2— holds the 'Syntax Value', which in C# it is *using ;* and in C++, the value is *#include <>* .

According to the second node —1— holds information for structure or classes in languages. This node contains three attributes. Attribute —0— holds 'Name of Struct *(or)* Class', which contains the user-defined name they have created inside VisualPro. Attribute —2— holds the 'Properties' like the member of the struct/class. Attribute —3— holds the syntax value, the open and close case of a struct/class and layout within the struct/class.

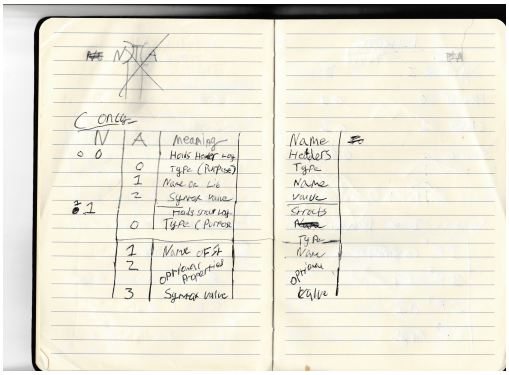

Fig. 6. Notebook Plan Page 7-8 - Found at: Original Image

6) Notebook Page 9-10: Figure 7 answers how the loops and logical statements work in the LanguageCompiler library. This planning shows how the XML nodes and the Planner List should work together. In theory, the Planner List, of what the user populates with the program's use, combines with the XML document with the chosen language.

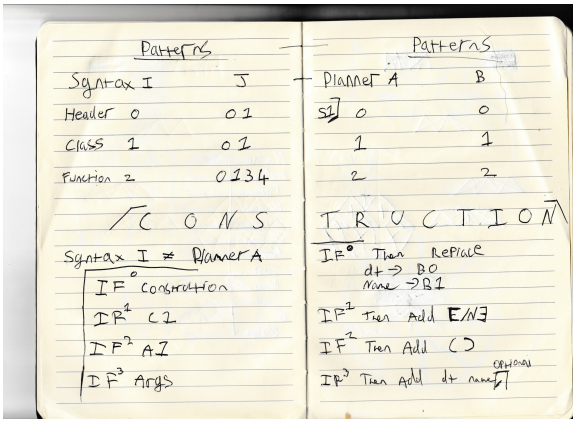

Fig. 7. Notebook Plan Page 9-10 - Found at: Original Image

7) Notebook Page 11: Figure 8 displays how the function node should work. This miniature theory is to help the developer understand how the XML document should work in a programmatical sense. This again helped when it comes to the prototypes further in this document.

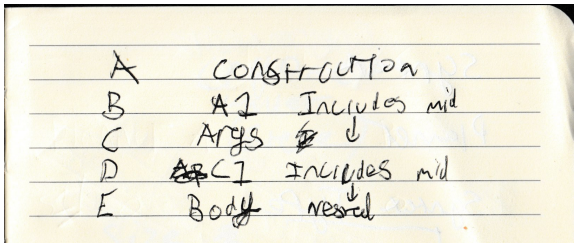

Fig. 8. Notebook Plan Page 11 - Found at: Original Image

8) Notebook Page 12-13: igure 9 shows a predicted problem is as follows:

Description:
How will the program know where to put a sub-child and how will it?
Answer:
The planner would hold its parent and sub identity, and the Triangle (30 Code) symbol tells the software where to put the child. The Triangle symbol will only appear of classes or functions that have children. This means that the program will focus on parents, close the tags, then rescan to put the children in the code. The problem before is that it would be hard to tell the program to remember when and where to close tags if it got too deep in theory.

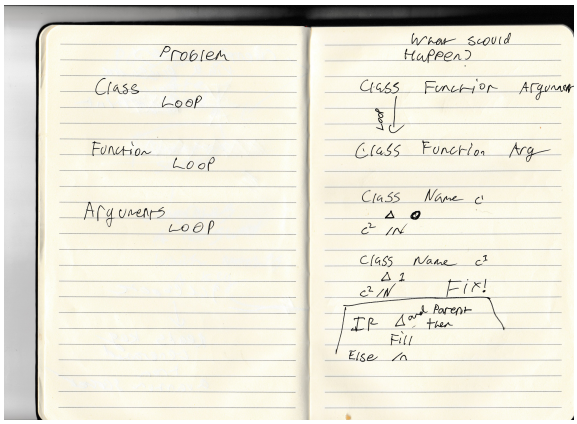

Fig. 9. Notebook Plan Page 12-13 - Found at: Original Image

C. Surveying

1) Survey Question 1: According to the chart 10, *87.5%* chose 'Programming Planner Improved', and *12.5%* chose 'None of the Above' to the question, 'As a developer, which program worked for you?'. This data suggests that most of the responses preferred 'Programming Planner Improved' out of eight.

2) Survey Question 2: Figure 11 asks the question, 'If the last option on the previous question is ticked, then explain why?'. Four responses replied with:-

- 'More flexibility' -
  This response could mean two things: the Programming Planner Improved is more flexible than the first program or that both programs would benefit from more flexibility.
- 'Both work' -
  This response indicates that both software works.
- 'The option to choose the desired programming language at the end of the programming' - A surveyee liked the option in Programming Planner Improved to select different programming languages.
- 'Because it had more options and had example code at the end of planning' - This answer describes that the Programming Planner Improved offered more options, such as argument selection and code language.

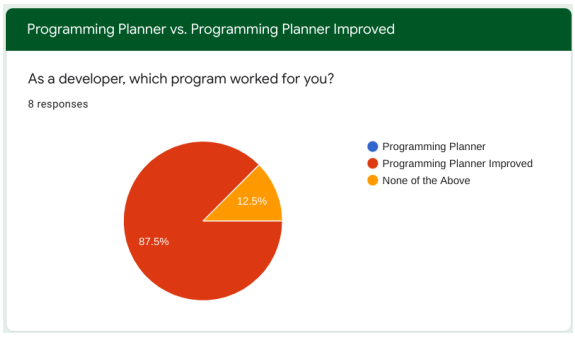

Fig. 10. Survey Question 1 - Found at: Original Image

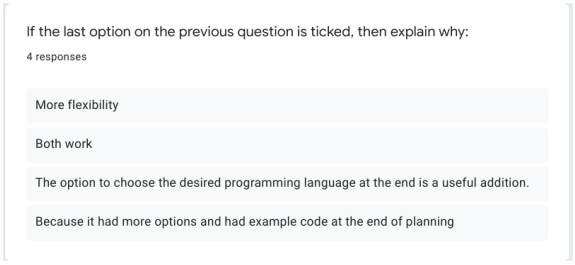

Fig. 11. Survey Question 2 - Found at: Original Image

3) Survey Question 3: Figure 12 asks the question to Surveyee's, 'Did the tutorial at the beginning of Programming Planner Improved help navigate around the application?'. This question tries to find out if the application is hard to use overall. The responses were 87.5% for yes, and 12.5% for no. Even though the most of the responses chose yes, the application may still be hard to use if, theoretically, one out of eight users found the User Experience frustrating or confusing.

4) Survey Question 4: Figure 13, 'What makes the Programming Planner Improved better than the previous product?' checkbox question, offered five qualities. These are:-

- **Ease of Use** - Aimed to find out how easy the application is easy to use. 50% of the responses think it was easier to use than the previous product. This feedback could mean to things, the first product was similar and consequently did not improve the application in this quality or that the product needs to improve in this area.

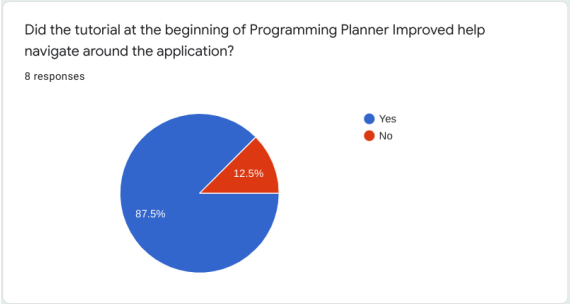

Fig. 12. Survey Question 3 - Found at: Original Image

- **Performance** - To see if the audience could notice any performance difference. This quality is vital as a Graphical User Interface (GUI) can somewhat be 'chunky'. The responses were 0% out of the responses; though this may seem negative, it is sometimes hard to notice the difference between console and desktop applications.
- **Extra Features** - The quality, 'Extra Features', displays if the extra features in the new application stood out and improved the previous application. The responses were 62.5%, which indicates that the extra features did stand out and made the application better.
- **File Structure** - The quality, 'File Structure' of the second application is revised and stores the saved file in a directory other than the ROOT directory. 25% of the responses chose this option, which means that the File Structure did not impact the application as effectively as the 'Extra Features' quality.

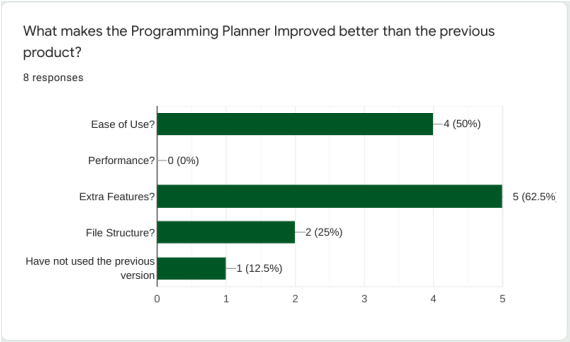

Fig. 13. Survey Question 4 - Found at: Original Image

5) Survey Question 5: Figure 14 asks the Surveyee's, 'Explain your answer' to the last question. This question will tell what the Surveyee's felt when answering the last question. The responses are as follows:

- 'More features allowed for a wider range of approaches' - This specific feedback suggests that it is better to add more functionality in the Graphical User Interface application to allow for more usability.
- 'More languages and was easier to use. Very impressed' - This response implies that the idea of a dynamic language interface to allow users to add more languages will increase User Experience.
- 'Easier to use, the better' - Visual Scripting already is complicated when looking at the Unreal Engine's Visual Scripting interface. If it is possible to create a lightweight Visual Scripting pad to allow a programmer to create code snippets, it could bring more developers into the field and increase production time already in the field.
- 'Have not used the previous version' - This suggests that the Surveyee only tested Programming Planner Improved.
- 'Allowed me to create a function structure and save it. Seemed difficult to follow (variables weren' numbered)' - According to this response, the user found it difficult to follow through and should be something to think about when making the GUI application.
- 'Easy, helpful and understandable' - This response shows that the user found the software easy to use, helpful and understanding during their experience.
- 'The extra features added to the improved version of the program improve used experience by adding expanded usability' - The response suggests that again the extra functionality helped improve the User Experience, in Programmer Planner Improved.
- 'It just had extra features and seemed more useful' - Which implies that the user found the extra features more comfortable and useful to use.

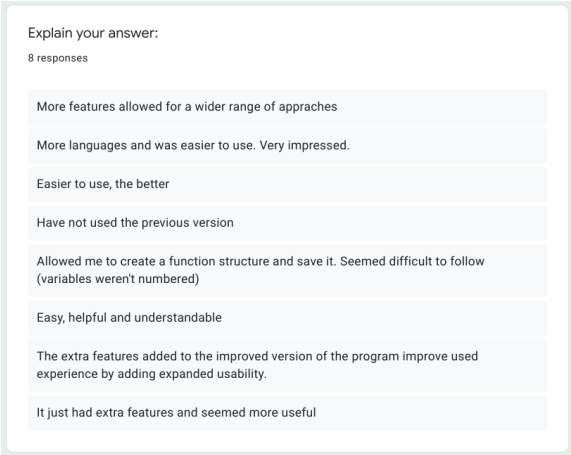

Fig. 14. Survey Question 5 - Found at: Original Image

6) Survey Question 6: Figure 15 asks the Surveyee, 'How would you rate the User Experience (overall)?'. The question tries to aim for an idea of how the UX of the two products. On a scale of one (Stressful) to ten (Relaxing), 62.5% of the responses went with option eight, 37.5% of the responses went with option seven. These figures show that the UX has room for improvement. When designing and implementing the GUI product, it is important to think about the UI and UX.

7) Survey Question 7: Figure 16 asks the question, 'How useful is Programming Planner for beginners?'. This question is done on a scale of one (Complicated) to five (Useful). The feedback is ranged from one to five and suggests that

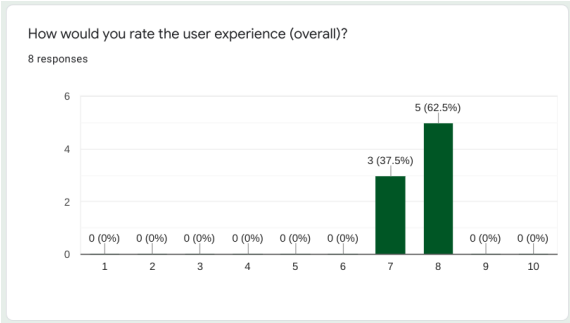

Fig. 15. Survey Question 6 - Found at: Original Image

beginners may find this platform hard work. The responses are as follows:

- **One** - 25% responses. These respondents reckon that Programming Planner is complicated for beginners.
- **Three** - 12.5% responses. These respondents thought that Programming Planner is a little complicated for beginners.
- **Five** - 50% responses. These respondents figure that Programming Planner is close to Useful for beginners.
- **Six** - 12.5% responses. These respondents consider that Programming Planner is Useful for beginners.

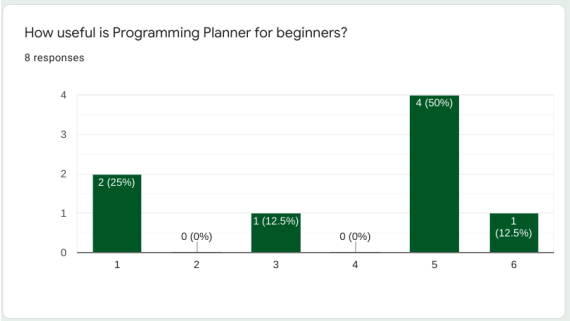

Fig. 16. Survey Question 7 - Found at: Original Image

8) Survey Question 8: Question 17 'Did this help create a structure of your favourite language faster than standard methods?'. 75% of responses said Yes, 12.5% of responses said 'Would not recommend to new programmer as they will not learn the basics' and 12.5% said 'I have not used other methods'. This chart suggests that the majority found this software to be faster than their standard methods.

9) Survey Question 9: - Question 18 'What features would you wish Programming Planner to offer?'. This question attempts to determine if the user has any requests about the software they have used. These were the responses:-

- 'When I wanted only one function on the variable creation it asked if it works with other functions but I didn't have any other functions' - This suggestion shows that the user was confused with the way the program asked the user if the variable was a argument or a standard variable.

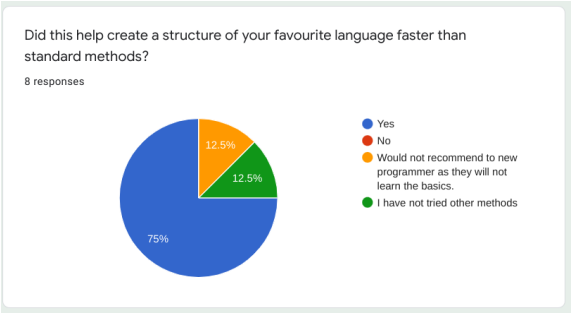

Fig. 17. Survey Question 8 - Found at: Original Image

- 'The target audience for this program will have little coding experience, and may forget what is required of a function, variable, etc. Adding examples to each page may be a good idea to reduce confusing.' - This suggestion shows that a help page or tips pop up on creating an element within the GUI program will help beginners use this software.
- 'A set of pre made open source programs that could be quickly accessed' - Perhaps, the program could offer a GitHub repository import, which will examine the code and visualise it for the user.
- 'Would be better with visualisation if writing a longer program' - The response could mean two things. It could mean that the user wanted to have an ability to overview the code within the console application, or it could mean that the user would have preferred a GUI work area.
- 'A GUI' - This response means that the user would have preferred a GUI product rather than a console application.
- 'Allow sudo code comments to be added to the function' - In the GUI application, it could have the ability for the user to enter comments.
- 'Calculator?' - This response did not seem useful to add in to the future software, as popup calculators are available in most operating systems.

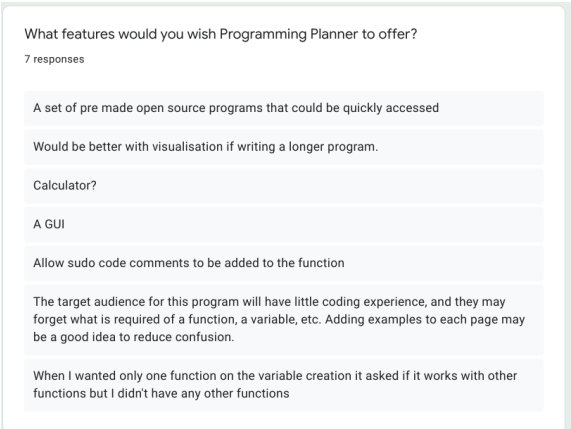

Fig. 18. Survey Question 9 - Found at: Original Image

10) Survey Question 10: Figure 19 ask 'Would a Graphical User Interface uplift the User Experience?' as a multiple-choice question. The 100% chose the yes choice. This feedback implies that they would like a GUI to work with over a console application out of the eight responses.

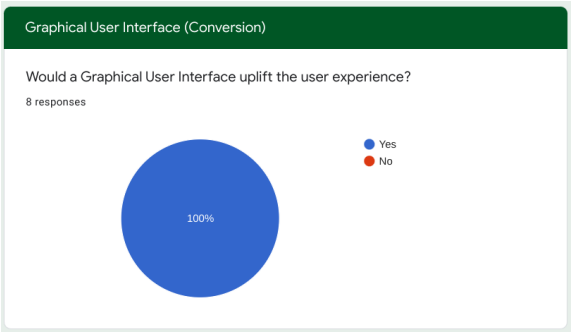

Fig. 19. Survey Question 10 - Found at: Original Image

11) Survey Question 11: A grid question asked 'According to you, how should the features of the Graphical User Interface have to work?'. The following options are: 'Clickable', 'Drag and Drop', 'Drop Down Menu' and 'Text Filled'. The details are 'Structure Elements', 'Logic Elements', 'Select Languages and Save' and 'Sub Elements'. After overlooking the table 20 'Structure Elements' looks like it favoured Drag and Drop the most by four responses. The Drag and Drop were also voted for the 'Logic Elements' by five responses. The 'Select Languages and Save' preferred the Drop Down Menu by six responses, and lastly, the 'Sub Elements' liked the idea if it was Text Filled.

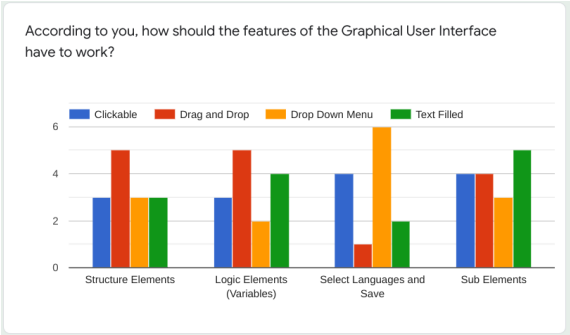

Fig. 20. Survey Question 11 - Found at: Original Image

12) Survey Question 12: This grid asks the participants, 'How would you prefer to access the software?'. The options are 'Windows Operating System', 'Linux Operating System', 'ChromeOS/Android' and 'iOS'. The details are 'Computer, Laptop or Tablet Devices', 'Web Application', 'Mobile Devices'. For the first column, the responses favour the Windows Operating System by eight. The second column, Windows Operating System, is also preferred by eight. The last column, ChromeOS, is the six responses favourites, and iOS is preferred by four. After reviewing this, the application aims for the Windows Operating System to get the most attention. Programming Planner and Programming Planner Improved works for both Windows and Linux Operating Systems.

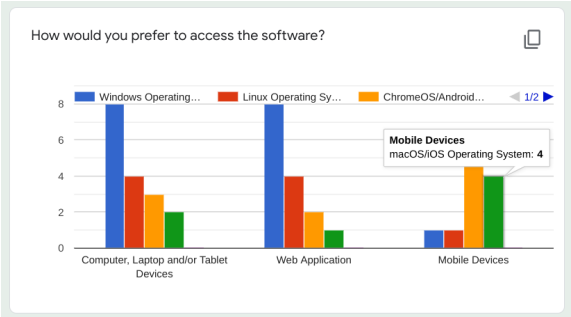

Fig. 21. Survey Question 12 - Found at: Original Image

13) Survey Question 13: Figure 22 a question, 'Any comment?' follows the last questions. The only response was 'No'.

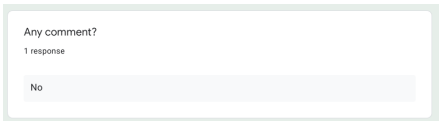

Fig. 22. Survey Question 13 - Found at: Original Image

14) Survey Question 14-15: Figures 23 and 24 shows two questions, 'How complicated is Visual Scripting? (Reference to Unreal Engine for an example).' Furthermore, 'How complicated is Website Builders?' these questions portray how existing Visual Scripting Web Builders are successful, and if it is possible to do a service that is easier to use, similar to a web builder. The question is on a scale of one (Piece of Cake) to five (Complicated). The feedback from the first question:-

- Three - 37.5% of the responses. This indicates that this percentage of the participants find Visual Scripting moderately easy.
- Four - 50% of the responses. This displays that 50% of the participants find Visual Scripting a bit more complicated than the 37.5%.
- Five - 12.5% of the responses find Visual Scripting tough.

The feedback from the second question:-

- One - 25% of the responses find Web Builders a 'Piece of Cake'.
- Two - 50% of the responses find Web Builders more or less easy.
- Three - 25% of the responses find Web Builders moderately easy.

These findings will change the research perspective to look more into the Web Builder layouts, functionality and UX to work out the GUI of the product.

15) Survey Question 16: Question 25, 'If Programming Planner had an easy-to-use drag and drop feature without input and output relationships, then will this improve Visual Scripting overall?', the question paints a picture from the participants to see if the relationships in Visual Scripting that

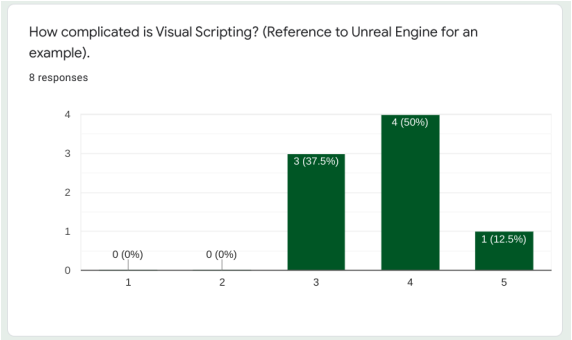

Fig. 23. Survey Question 14 - Found at: Original Image

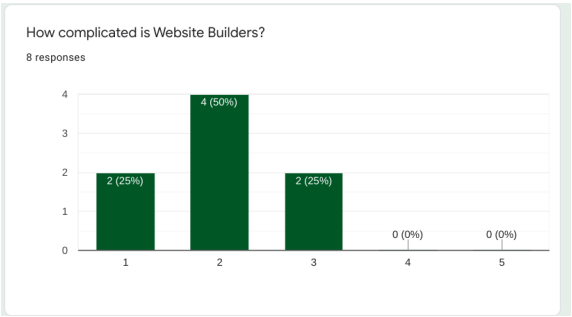

Fig. 24. Survey Question 15 - Found at: Original Image

is similar to Entity-Relationship Diagrams if they cause a problem. All eight responses replied back with yes.

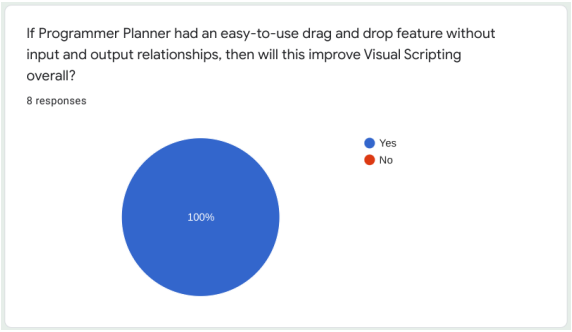

Fig. 25. Survey Question 16 - Found at: Original Image

16) Survey Question 17: Similar to the previous question to paint an image from what the participants want to see, 'Would Visual Scripting be better if it allowed users to add any language they wish and generate any language with one click of a button?'. According to the figure 26, all of the eight responses responded with yes.

17) Survey Question 18: To squeeze more information from the participants, a question, 'What was the reason for your selection?' was asked and made compulsory. The responses are as follows:-

- 'Would be easier to write programs (for some people)'
- 'A feature such as this would greatly reduce the production times involved in program development.'

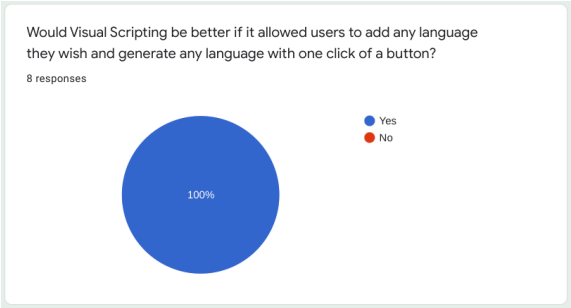

Fig. 26. Survey Question 17 - Found at: Original Image

- 'save time and allow users to focus on the details'
- 'I would like to code in any language I choose if that option was applicable.'
- 'Useful feature'
- 'Makes it more visual'
- 'Again, the easier to use, and the more features, the better'
- 'More accessibility'

After reading the feedback, a few points are to address. The current software already makes it easier to write code for some people, so it is imperative that the new software does not distract but improve the capability of writing software. The command-line may benefit users as it keeps developers in a similar environment they are probably already using. Another point to address is this question, 'Will the GUI software get too heavy with too many features making it no longer lightweight?'. The best thing about having Dynamic-Link Libraries is that developers can incorporate into Java Android, iOS, Web Applications and, but not limited to, Visual Studio Code Extensions. The backend can open a 'can of worms' to allow different interfaces to the top of the backend, like digital notebooks or other ideas. Anyone who develops on top of the backend can make any interfaces or even new scripting languages. This idea, in turn, will enable others to make it easier to use Visual Scripting and Languages to Write. For example, Python or JavaScript has the potential of being converted into C++ and vice-versa.

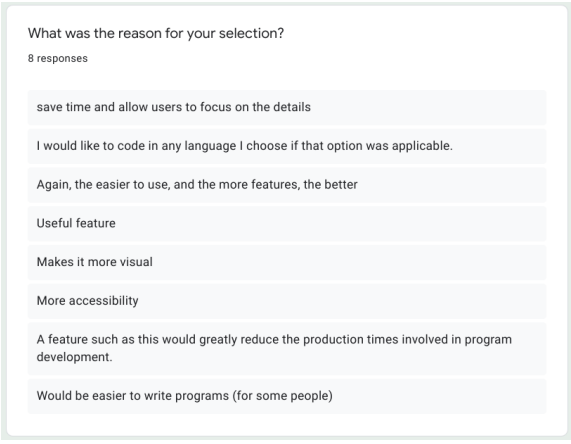

Fig. 27. Survey Question 18 - Found at: Original Image

18) Survey Question 19: Final question which asked the participant, 'Would you like to leave any extra feedback?', which hopefully gets any extra feedback that the survey missed out. Six responses, excluding two including the response 'No', are displayed below:-

- 'Introducing both drag and drop, as well as text filled features will appeal to both beginners and experienced programmers.' - The reply suggests that both Drag and Drop menus and Text Filled features will appeal to both programmers than just one of the methods used. Another idea this program opens up is creating snippets that can work in Visual Studio Extension, giving the user more flexible tools for development.
- 'A training manaul along with some begginer tutorials would really help new users get to grips with the software' - A training manual or documentation would benefit both the console and GUI applications.
- 'Command-line is hard on eyes when using. Not very good for Visual Scripting in general.' - The best thing about the console application is that it already has a language compiler, which plays a massive part to give the GUI application a backend. However, this response backs up the reasoning why the frontend matters in this project.
- 'Nothing besides a nice GUI' - Many Visual Scripting is done in GUI applications, so it was unusual for a console application to have achieved a similar result.

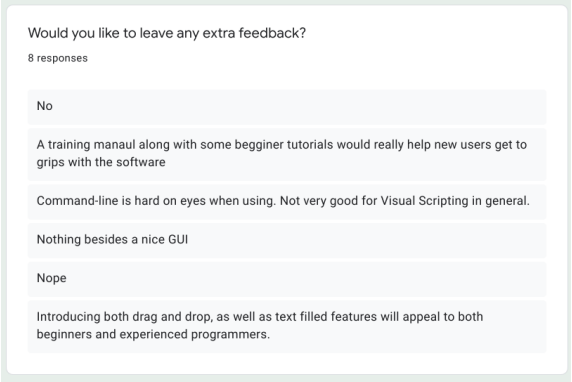

Fig. 28. Survey Question 19 - Found at: Original Image

IV. Implementation

A. Development Methodology

V. Conclusion

VI. Reflection

VII. Terminology

List of terminologies used in this document:-

- UI - User Interface.
- UX - User Experience.
- UML - Unified Modeling Languages.
- OOP - Object-Orientated Programming.
- FP - Functional Programming.
- DLL - Dynamic-link Library.
- HTML - HyperText Markup Language.
- AI - Artificial Intelligence.
- XML - Extensible Markup Language.
- GUI - Graphical User Interface.

Acknowledgement

Appreciation goes out to the survey participants, which helped better understand how VisualPro may impact consumers in the future. Having the support from these participants helped advance the UI and UX progress of VisualPro.

VIII. Reference List

[1] B. Vogel-Heuser, M. Obermeier, S. Braun, K. Sommer, F. Jobst, and K. Schweizer, "Evaluation of a UML-Based Versus an IEC 61131-3-Based Software Engineering Approach for Teaching PLC Programming," *IEEE Transactions on Education*, vol. 56, no. 3, pp. 329–335, 2013.

[2] K. Kuspert, J. Teuhola, and L. Wegner, "Design issues and first experiences with a visual database editor for the extended NF/sup 2/-data model," in *Twenty-Third Annual Hawaii International Conference on System Sciences*, vol. 2, 1990, pp. 308–317 vol.2.

[3] Dr. Caitlin Sadowski and Dr. Thomas Zimmerman, Eds., *Rethinking Productivity in Software Engineering*, 1st ed. Apress, 2019. [Online]. Available: https://doi.org/10.1007/978-1-4842-4221-6

[4] Unreal Engine, "Introduction to Blueprints." [Online]. Available: https://docs.unrealengine.com/4.27/en-US/ProgrammingAndScripting/Blueprints/GettingStarted/

[5] Pooley R J and Stevens Perdita, *Using UML: software engineering with objects and components*, second edition. ed., ser. Addison-Wesley Object Technology Series. Harlow, England: Pearson Education Limited, 2006.

[6] Unity Technologies, "Unity engine visual scripting | Game development software without coding | Unity Bolt | Unity." [Online]. Available: https://unity.com/products/unity-visual-scripting

[7] Mojang, "Minecraft - Getting Started with Redstone." [Online]. Available: https://help.minecraft.net/hc/en-us/articles/360045950932-Minecraft-Getting-Started-with-Redstone-