VISUAL DEVELOPMENT TOOL SUPPORT FOR ENHANCING PERFORMANCE AND QUALITY OF DEVELOPER WORK

U.I. Heenatigala

168223F

This dissertation submitted in partial fulfilment of the requirements for the Degree of MSc in Computer Science specializing in Software Architecture

Department of Computer Science and Engineering

University of Moratuwa
Sri Lanka
April 2020

DECLARATION

I declare that this is my own work and this dissertation does not incorporate without acknowledgement any material previously submitted for degree or Diploma in any other University or institute of higher learning and to the best of my knowledge and belief it does not contain any material previously published or written by another person except where the acknowledgement is made in the text.

Also, I hereby grant the University of Moratuwa, the non-exclusive right to reproduce and distribute my dissertation, in whole or in part in print, electronic or other medium. I retain the right to use this content in whole or part in future works (such as articles or books).

……

U.I.Heenatigala Date: ……

I certify that the declaration above by the candidate is true to the best of my knowledge and that this report is acceptable for evaluation for the Master of Science Project.

……

Dr. Indika Perera

Supervisor (University of Moratuwa)

ABSTRACT

There is a very limited number of visual development tools that support a software developer. The aim of this study was to develop a visual development tool and determine the effect of the tool on developing performance and quality of work of a developer. The research was conducted in five steps. Develop version 0.1 of the tool with basic requirements. Conduct a survey for the initial tool. Update the tool according to the survey results. Release version 1.0 of the tool to the sample with development tasks. Finally analyse the results. Thirty developers from the Srilanka were randomly selected using the convenience sampling method. Testing method was used to identify the efficiency of the developers and observation method was used to explore the quality of the work. Paired t-test in R package 2019 was used to compare the time taken to complete the given tasks using the tool and without the tool. The quality of work is measured by coding standards followed and also the structure of the code is inspected to determine how the developer has gone about solving the problem. Three tasks with varying complexities were given. Task 1 which is the least complex didn't give a significant result(P value >0.05). Task 2 and Task 3 development time analysis gave a positive result for using the tool(P value < 0.05). There are further features that need to be added to make it a product that can be used by the masses. Suggestions, a real-time preview of the generated code, more built in components and better style (css) generation are some of those features. The quality standards in readability and code reuse were improved by using the tool.

The results conclude that the importance of visual development tools to enhance the performance and quality of the developer according to the complexities of the tasks. Not only the statistical analysis but also the developer interviews also confirm this. Further, there is an added advantage that helps new vue developers to learn as well. As future work we plan to enhance the tool according to the suggestions given.

Keywords - Developer tools, Vue, Vue drag and drop, Developer performance, Code quality

DEDICATION

To all the parties

who are interested in the field of

Computer Science

ACKNOWLEDGEMENTS

I would like to express profound gratitude to my advisor, Dr. Indika Perera, for his invaluable support by providing relevant knowledge, materials, advice, supervision and useful suggestions throughout this research work. His expertise and continuous guidance enabled me to complete my work successfully.

I would like to thank all my software engineer colleagues for their help on creating a test group by doing the 3 tasks with tool and without tool and those who gave valuable feedback for further improvement of the developed tool. I would also like to thank Ms.Dilini Jayakody , for providing valuable resources and advice in the statistical analysis area of this research.

Lastly, I would like to thank Riverview Innovation Labs, for the support given me to conduct my research there and to manage my MSc research.

TABLE OF CONTENT

LIST OF TABLES

LIST OF FIGURES

CHAPTER 01

INTRODUCTION

1.1 Background of Study

Software quality is the most important factor for software development as it mainly defines customer satisfaction that is directly related to the success of a software project [1]. The CISQ software quality model defines Reliability, Performance efficiency, Security and Maintainability as the four important indicators of software quality [12]

Cost of building programming legitimately identifies with time. Following is a rundown of various degrees of hourly rates. "Enterprise Class" Custom Software Development Companies. As the biggest players in the market, Enterprise Class consultancies for the most part have hundreds if not a huge number of designers and specialists on staff and by and large work with governments and Fortune 500 organizations that can manage the cost of their out of this world rates [10].

Projects for the most part go in budget from $500,000 - $100,000,000+. Hourly rates are high as between $250 - $850 every hour, contingent upon the experience level of the designer/expert [10].

“Big Business Class” Software Development Companies. Huge Business consultancies will in general work with other enormous organizations that can't exactly manage the cost of the expenses of the endeavor class shops, yet at the same time have large spending plans. They normally have between 100 - 1,000 engineers, and a few workplaces around the globe. They're not as costly as the Enterprise Class, however they absolutely aren't modest. You can hope to pay between $200 - $300 every hour for ventures going in size from $125,000 - $5,000,000+.(kite agency)[10].

The above facts show that the software quality and efficiency is an essential requirement to the success of a software development company. Many organizations have implemented and attempted different methodologies intended to improve work efficiency and quality of software programming.

1. **Software quality assurance (SQA)** is about processors; it makes sure that the built software is up to the desired standard. It is a preventive step where processors and methodologies are used to make sure the built software is up to the desired standard.
2. **Software quality control (SQC)** is responsible for making sure features that are being shipped work according to the defined requirement set. Software quality control is more about building a better product. It is product oriented and not process oriented.
3. **Testing** is a fundamental action performed both by the developer and the software quality tester. Tests targeted specifically to the area of the program which was changed. Test driven development software development process is very popular since it makes sure the code shipped performs as expected.
4. **Agile software development** is based on having many iterations, Each iteration includes small increments of features. This makes mistakes less problematic due to there being a constant validation of the features built.
5. **Documentation and enhance collaboration -** To maintain any software there should be proper documentation. Where a new software developer can easily access and understand the architecture of the software. Software is built by teams and collaboration helps to have clear patterns used throughout the software.
6. **Use Software tools** - IDE's, Code generators, Project management tools etc [7].

Use of software tools is the area where companies spend the least amount of their budget. The average price of software tools like phpStorm costs around 200 dollars but it improves productivity of developers a lot. This is the area that this research tries

to improve by building a tool that helps developers to build more quality software efficiently.

Most of the software developers are still using the regular code editors to develop software. Software developers struggle with the two sources of the complexities. One is code and the other is the complexities in the process of producing it when designing large projects. Therefore, it should be done with considerable investigations to develop tools and techniques to manage them well [8].

Some researchers have found different visualization tools that support the distribution of software development processes like Augur device which makes visual representation of both programming artefacts and programming improvement exercises, and, essentially, permits engineers to investigate the connection between them[8].

Vue (Open Source JavaScript Library) is used to build user interfaces in websites. It was initially created by Evan you. A lot of awesome projects have been created using the vue library. But all this has been handwritten code. None of it was generated by a tool. There are a numerous number of repetitive tasks that can be done and generated using a tool. Therefore, it is worthwhile to develop a visual development tool to enhance quality and performance of a developer. Companies spend many millions to optimize and upgrade the standards of development. Investment for a tool that helps to do this is very much in need.

It is clear that quality software is what we should aim for and building quality software is very expensive. Many companies spend millions in software quality improvement methodologies. But the amount spent for development tools that improve software quality and reduces development time is close to zero compared to other expenses. This is due to the lack of visual development tools that will help to build better software in a shorter timespan. Therefore this research focuses on building a software tool that helps developers build better software in a shorter time. Further it tries to clarify the actual effect of the tool that was built. By running a comparative study using the built tool and without the built tool.

1.2 Problem Statement

Vue developers spend a lot of time typing the same thing repetitively this takes time. There are a lack of standards followed. There is no visual development tool that has been developed to aid a vue developer which will save time and enhance standards.

1.3 Objectives of the Study

1.31. Major Objectives

To develop a visual development tool to enhance performance and quality of work of a vue developer

To explore the effect of visual development tool on developing performance and quality of work of a developer

1.3.2 Specific Objectives

To compare the coding efficiency of developers using the tool and without the tool.

To evaluate quality of work by reviewing code and analysing how a problem was approached.

To find out the issues of visual development tool when using it

1.4. Hypothesis

H0 ; There is no significant difference between development time using the tool and without the tool

H1; There is a significant difference between development time using the tool and without the tool

1.5. Limitations of the Study

In this research, only 30 developers from Colombo were chosen as the sample. Plus this sample includes different levels of developers, some senior engineers and some are inexperienced engineers. Further the development tool supports only a few features and this research only takes into account software (tasks given to sample) that can be built using the features available.

Therefore, the generalization of results will be questionable as the sample size was small, and the due to the limited features and the narrowness of the tasks given to the developers.

CHAPTER 02

LITERATURE REVIEW

2.1 Vue.js

Among the frameworks, Vue.js was firstly released in 2014 for the purpose of creating a library. Though it is a recent development, it has been growing a user database. Laravel php framework suggests using Vue.js for building the frontend application in Laracasts [16]. Vue has reached 163,000 stars to date which is higher than react.js and angular.js

Vue is a progressive javascript framework that is used to facilitate the creation of interactive, stateful & reusable user interface components. It is used in big companies like Alibaba, GitLab, Grammarly, 9GAG, Behance and Laravel Spark. Vue was built by Evan You when working at google. At the time he was working with angular.js. This has influenced vue.js to be similar to angular rather than reactjs. The idea of declaratively specifying user interfaces on top of a data model, has come from frameworks like angular. The user interfaces automatically keep in sync with underlying data. The developer has to focus on the data the UI will automatically update itself according to the data. This is achieved by a reactive system. Unlike the aforementioned frameworks, Vue uses JavaScript rather than HTML to construct those user interfaces, citing flexibility as the reason for this design decision [20].

Vue.js is the best lightweight front-end framework based on MVVM mode in Web applications. The schematic diagram of MVVM modern architecture, in which ViewModel, like middleware, is responsible for communication between functions and data [23].

Vue utilizes a layout motor like that of Angular and Ember and gives two-way information authoritative between the HTML format and controller. Con-trollers in Vue are exquisitely written6 and give a simple method to tie the HTML format and controller. Vue gives an approach to characterize segments, like the Handlebars in Ember and mandates in Angular. These parts can be characterized in .vue documents,

joining JavaScript and HTML in a solitary record, like that of the JSX records in React[16].

Obviously Vue depends on a mix of parts from both Angular and React. Vue also has a framework that is called Nuxt.js which provides server-side rendering out of the box. This is a big plus for building performant and modular web applications. Most of the companies that are using Vue.js use Nuxt.js for building their websites[3].

2.2 Integrated Development Environments

Studies have been done on the impact of Integrated development environments. Integrated development environments provide a software engineer with an environment that is aware of the programming language that the developer is using. With this knowledge the IDE can provide intellisense, code snippets show errors before run time.

IDEs like phpstorm support php as the main language but do support other languages like javascript, css and html. Android Studio supports android app development and Xcode IDE supports development of ios applications. These tools are now must use when developing android and ios applications because they provide so much support for the developer.

IDE's also provides a runtime environment for your application for example phpstorm provides a server that is able to run php. Android Studio provides simulators to run the android project. This makes the developer experience much better since a real device is not needed and developers can debug their code. Further IDE's can provide different device configurations. Android studio and Xcode gives the developer to choose from many devices that are in different os versions. This helps to launch apps that support wide range devices.

Xcode contains a set of generic components for ios app development. These components are categorized into three sections - Bars, Views and controls. Bars contain components like navigation bars, search bars, status bars, tab bars etc. Views contain components like actions sheets, activity views, alerts, collections etc. Controls

contain components like system buttons, details disclosure buttons, info buttons, context menus etc. The developer is able to drag and drop these elements to create the user interface of the app. The developer then is able to attach event handlers to do these components. This is the approach the visual development that we will build in vue.js will perform.

2.3 IDE's with enhanced domain specific interaction

Further advancement on the IDE instruments is to connect with various area models. In building areas displaying of physical frameworks assumes a significant job. For engineers it is essential to show genuine articles so as to reproduce and survey their conduct. This is especially valid for the improvement of new item parts such as, wind turbines. The reasonableness and solidness of segments should be researched right off the bat in the plan procedure to stay away from costs that emerge when flawed structures show up during part testing[21].

Intelligent model approval has been actualized to help the client during improvement. Thus, the client gets quick input whether hismodels are right as indicated by the linguistic structure and semantic guidelines characterized by the language in particular. Moreover, extra semantic data and rules can be characterized in a nonexclusive manner to confine the manner in which models are created. This can help distinguish structure mistakes that can be difficult to identify when exploring the code as it were[21].

OneModelica permits to make substantial Modelica models and to use arbi-trary Modelica-consistent reproduction devices. Reproduction results can be inves-tigated inside the IDE and a programmed test system empowers test-driven turn of events and guarantees that models carry on true to form. The implemen-tation vigorously utilizes model-driven programming improvement and therefore shows that it is conceivable to make amazing IDEs for complex space explicit dialects, for example, Modelica[21]

2.4 Impact of Integrated Development Environments

The impact of integrated development environments wasn't what was expected. The assumption that it would dramatically improve over regular text editors like emacs or vim. A study was done on a group of developers to reevaluate the benefits of Visual Studio an IDE for building web apps, mobile apps by microsoft. There was actually a significant improvement but it was not so clear when other aspects like unrelated libraries which were used by the IDE and the time taken to learn and debug in the new environment were taken into account.

The ready made components that were given by the IDE needed separate learning and getting used to by the developers. A considerable time was also taken to edit the code that was generated. It was found that adopting new libraries and other frameworks may help gain productivity.

The analysis shows that a search function does not solve the problem area that takes the most time. It was only a small part of the more complex process. The complex process was the program comprehension. The research breaks down previously envisioned processes taken by software developers in solving a maintenance request into 2 steps.

1. Find the related code for the maintenance request - This includes searching for strings related to the request or function calls
2. Study the code that gives the unwanted behaviour and update the code.

The true difficulty in these two steps is not solved by a comprehensive search. The difficulty lies in the code space that has to be understood. A program contains deeply nested routine calls that often spans out to ten or more levels. With this understanding the research shows an in depth process;

1. Running through the control flow to access the related bit of code
2. Tracking the call stack of the function
3. Getting to the starting to point of the function
4. Tracking external behaviours that function is used.
5. Understanding the variables and the variable types

In summary it is not the time programmers spent on searching the search function only improved the mistakenly going to wrong areas of the code. The real difficulty was the program's comprehension [21].

2.5 History of user interface

User Interfaces have been developed with the development of Personal Computers(PC), even before the field of Human-Computer Interaction was built up. There were few published papers that showed how user interfaces have been developed over the years. There were different features and tools have been added day by day. According to these researches, graphical interfaces are mostly discussed. With the User Interface being an unequivocal factor in the expansion of PCs in the public arena and since it has become a social marvel, the time has come to illustrate its history[13].

2.6 Developing a Graphical user interface

User interface is a significant aspect which is used by software developers. Most programming that flops because of poor UIs and client experience. It is important to understand that the user understands a product, software through its user interface.

User experience is also very important. It goes hand in hand with the user interface. User experience is what a user feels when the user is using the software. Is it intuitive does it give the user emotional pleasure and does it give a good attitude towards a product? It is important to have a good user experience and user interfaces for a product to achieve its intended use.

According to Nielsen the usability of a software can be broken down to five categories. They are as follows

1. Learnability
2. Efficiency
3. Memorability
4. Errors

5. Satisfaction

Nielsen also suggests that to have a high possibility of designing a good user interface a designer can follow eleven steps. He also says that following all the steps don't guarantee a highly usable software. For the development of the visual development tool the author considered learnability and memorability to be the main attributes of the user interface design[17].

2.7 Principles of visual design

It is important to understand principles of visual design before starting the design of software. The designer should understand the platform of the software in the case of the visual development tool that platform is web. The target browser is google chrome. The tool will only support a desktop view. The following guidelines were considered.

1. Consistency - Through the application a common theme and reusable components should be used. A design system was built with colors and spacing defined.
2. Universal usability - The user interface should support different types of users from beginners to experienced developers.
3. Give proper feedback - The user should get proper feedback from the actions that were carried out. It should be visible but not in a non interruptive manner.
4. Actions should be organized - Flows of the application should show a beginning, middle and end. Flows also should contain proper feedback after each step to show the user satisfaction of completing a step. And then move the user to the next required step.
5. Error prevention - The user interface should make it difficult for the user to make errors. For example a number input - a user should be only allowed to enter numerical values. Further errors should be shown at time of action rather than later. Javascript validation of inputs can be used to do this. Before sending it to the server
6. Undo action - The user should be able to reverse the action. This allows the

developer to easily try out elements without having the fear of making a mistake.

7. Support internal locus of control - The user should have the feeling the he/she is controlling the application and understands it. Hence surprises should be avoided
8. Lower short term memory load - The short term memory in humans is very limited and this available amount should be used vicely and shouldn’t be stretched. Consistency can help to reduce short term memory load in many user interfaces.

2.8 Development tools for different situations using Java

It has been utilized as Java Beans' good parts. In normal activity of the framework, the client may produce a "bean" segment by summoning a wizard-based interface that executes strategy for naturally creating and dealing with the bean. The client utilizes the wizard to indicate data about the bean, for example, the name of the bean, the bundle it will be in, and the class it stretches out from. In light of the client input, the framework makes a bean with the name the client indicated, places it in the client's present undertaking, and shows the source code produced for the bean.

The client may outwardly connect with the bean (or other existing bean) by utilizing the visual architects to deal with the bean's properties, including: adding properties to the bean, including bound and obliged properties, adjusting properties of the bean, expelling properties from the bean, or in any event, creating a custom property manager. Likewise, the client may continue to utilize the visual fashioners to deal with the bean's occasions, including: including occasions, tuning in for occasions, and making custom occasion sets[25]

Java SCP arrangements have been utilized to create highlight models which are considered as in various phases of programming improvement and are perceived to be a significant resource in model change strategies and programming product offering advancement. This improvement was being perceived as one of the key difficulties for automated programming advancement with regards to Software Product Lines[3].

The instruments like Chemistry Development Kit (CDK) which gives normal assignments in sub-atomic informatics, including 2D and 3D rendering of concoction structures, I/O schedules, SMILES parsing and age, ring look, isomorphism checking, structure outline age, etc.using Java with preparing a web interface, just as for applications and customer side applets [24].

2.9 Android development tool

AndroidRipper, is an automated strategy that tests Android applications by means of a Graphical User Interface (GUI). It depends on a UI driven ripper that consequently investigates the application's UIs attempting to discover blames in the android application. Running the application on an open source android application demonstrated that the graphical UI based experiments had the option to identify beforehand obscure serious blames in the underline code. It demonstrates that the organized way of android ripper outflanks an arbitrary approach[2].

AndroidRipper is a prime example that visual development tools work. As mentioned in Amalfitano, Fasolino et al. 2012 it enabled detection of previously unknown errors in the code. Also, structured exploration outperformed the previously used approach[2].

2.10 What is No Code?

No code is the new wave of software development. Where the user can build their own software without writing code. Webflow is a company that gives users to design and develop websites through a visual development tool without any code. Webflow, a no-code web improvement stage, has raised a huge $72 million Series A series of financing driven by Accel. The financing esteems the organization at between $350 million and $400 million post-cash, as per Forbes [5].

Chris Wanstrath, Chief Executive Officer in GitHub said that software developers will no longer use codes in the future. It is important to know how to work with no coding to improve the efficiency in the future. They have been developing new tools and teaching them for the people who are interested in it.

Bring a stumble into the future with us. Perhaps you've just heard some buzz about no-code, or possibly this is your first introduction. In any case, information is forced and there is in every case more to learn.

BuilderX is a company that gives a web based design tool that generates React Native & React code according to the design you created. It provides a photoshop like tool that runs on the browser which allows the user to create mobile app screens. Then these screens can be exported to different codebases like react native or flutter. The success of companies like webflow and builderX confirms the value and the need for graphical tools.

2.11 Code generation tools

Code generation is a way of saving time and a way of providing standards. It gives more flexibility to the user. Since the generated code can be updated according to the developers needs. This is the main advantage of code generation than encapsulating a common functionality in a library.

For the improvement of the programme quality and to increase code reuse of a program, Many frameworks have built in command line tools to carry code generation tasks. For example the popular php framework yii contains a command line tool that enables the developers to generate models, controllers and views.

When developing a project, the requirements, automation system architecture is important to consider from task to task.[26].

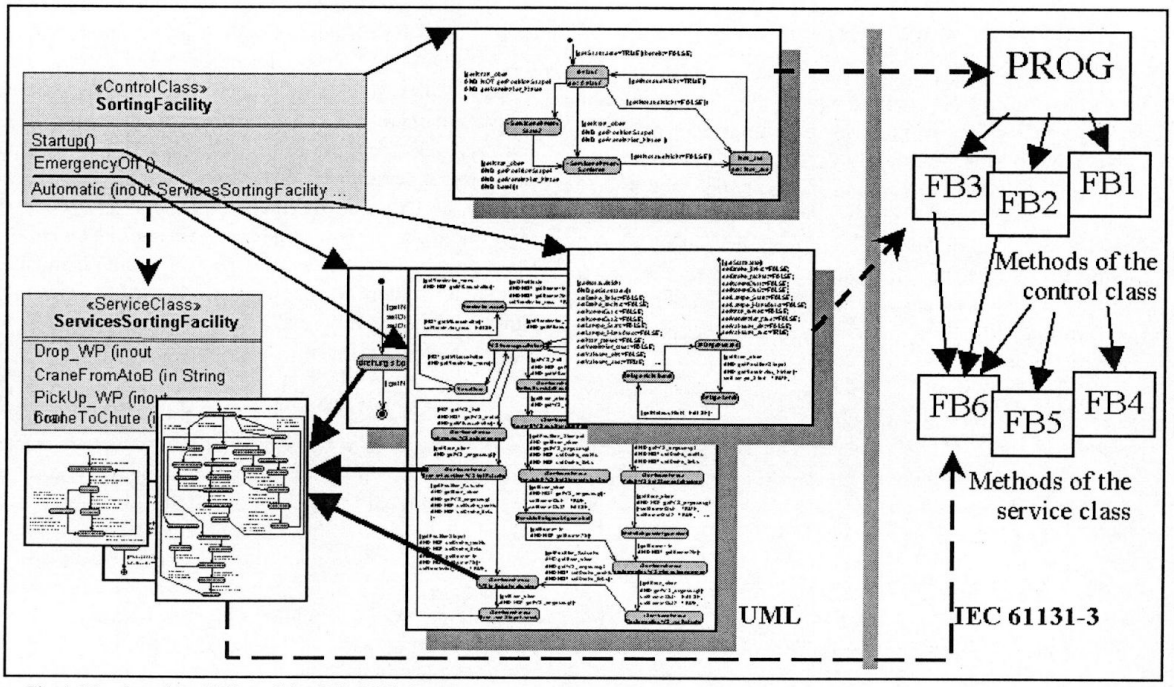


Figure 2.1 : The procedure of UML model map to IEC 61131-35

Note. Reprinted from "Automatic code generation from a UML model to IEC 61131-3 and system configuration tools", by Vogel-Heuser, B., Witsch, D., & Katzke, U. 2005, Paper presented at the 2005 International Conference on Control and Automation

This tool generates a IEC 61131-3 code from a UML diagram. The research finds that the prototype was utilized to exhibit that programmed code age for computerization innovation can be accomplished through pragmatic utilization of UML. In any case, the genuine advantages of item direction have not yet been taken. The advantages - yet in addition any issues - will get clear during application in an increasingly complex framework, This has been begun as of now. In such application instruments, for example, legacy will be utilized This will altogether disentangle the organization of variations and modules. The reconciliation of system angles is another objective. For circulated frameworks the exhibition of systems and the mapping from programming modules to various equipment structures relying upon the task size or chosen variations and the system execution is an essential. The combination of UML 2.0 with its framework charts is another working bundle. Another test is the interface

normalization to framework design or overseeing devices[26].

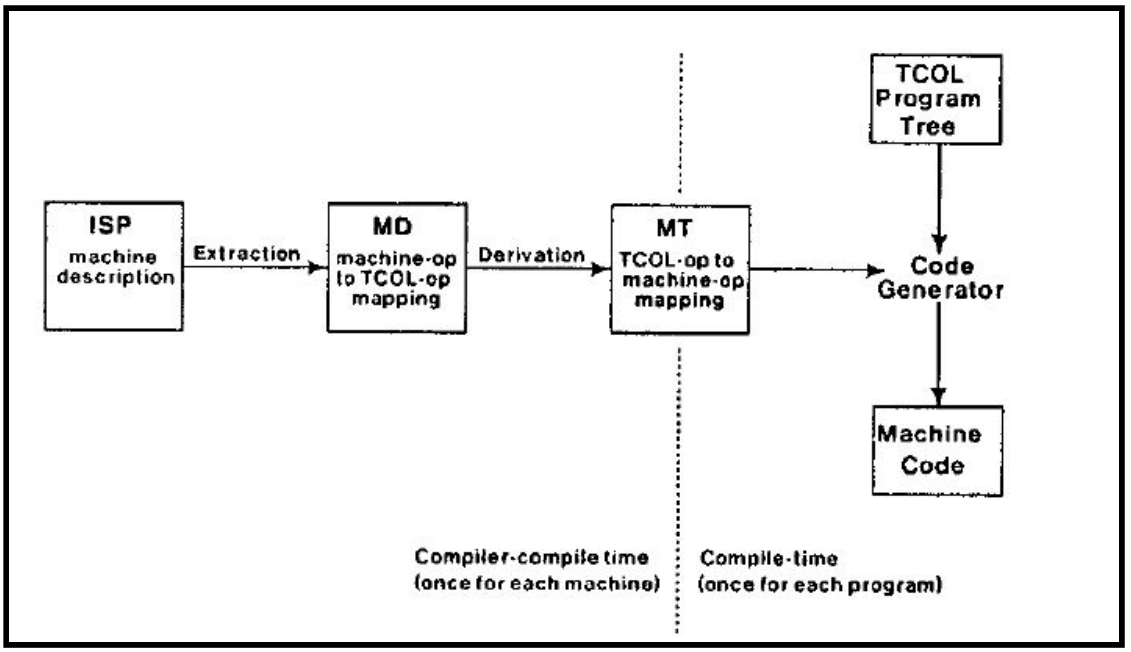


Figure 2.2 : Relationship of programs and representation proposed

Note: Reprinted from Automatic Derivation of Code Generators from Machine Descriptions ,Cattell. R., 1980,*ACM Transactions on Programming Languages and Systems TOPLAS), 2(2), 173-190.*

This sort of mapping is used during the development of the visual development tool where each component is mapped to a predefined code sample. When the developer uses a component and drags and drops it to the canvas this mapping function gets invoked to generate the related codebase[8].

The code generation tools depend on a type of formats which are considered as tree creations, which are gathered in the machine tables. A given source program is converted into a middle of the road parse-tree-like documentation by the front finish of the compiler. The code generator crosses the program tree, coordinating every hub against designs on the left-hand sides (LHSs) of the creations in the machine tables. At the point when an example coordinates, the right-hand side (RHS) of the creation determines code to be produced, unique compiler activities, for example, distribution,

or further matches to be recursively performed [8].

2.12. The effects of visual learning

Practice helps to establish previously learned behaviours. The motor skills can be gained through constant practice, accurate , proficient and can be executed more rapidly. Visual processing can be gained through more practice. Nature with a set or class of articles can prompt an expansion in our exactness in segregating and remembering them, especially in circumstances when the items are blocked or in any case hard to separate. Be that as it may, while much is comprehended about how tactile visual properties are spoken to in neural action, moderately little is thought about how experience improves their handling [18].

Proof from neurophysiological and mental examinations is meeting up to reveal insight into how we speak to and perceive objects. This survey portrays proof supporting two significant speculations: the first is that articles are spoken to in a mosaic-like structure in which items are encoded by blends of unpredictable, reusable highlights, as opposed to two-dimensional layouts, or three-dimensional models. The subsequent speculation is that change invariant portrayals of items are found out through understanding, and that this learning is influenced by the worldly arrangement where various perspectives on the articles are seen, just as by their physical appearance[27].

Learning is an unpredictable procedure. It tends to be characterized as an adjustment in manner; a generally lasting change in conduct after some time and this is achieved halfway by information. Learning can occur as a result of new achieved abilities, standards, observation, information, realities, and new data at hand[22].

These studies show the impact of visuals in learning.

2.13. Visual Development Tools

The greater part of the product configuration devices configuration to help altering, seeing, putting away, sharing and changing plans. Some products like Argo/UML, an article arranged plan device have been created with a few novel highlights that address

the recognized psychological needs of a product fashioner. It has incorporated a commonsense depiction of a few subjective hypotheses applicable to programming plan, a strategy for contriving intellectual help highlights dependent on these speculations, a bushel of psychological help includes that are exhibited with regards to a usable programming configuration device called Argo/UML, and a reusable framework for building comparative highlights into other structure tools[19].

Another work style model was created to imagine, structure and assess another age of inventive cooperation and programming configuration apparatuses planned for incorporating convenience and programming building [4].

Even though there is very limited research done on visual development tools. The surrounding literature on learning and human behaviour suggest that visual tools help learning. Further visual tools help teams discover hidden attributes that weren't found before. There is a paradigm shift of no code meaning building software without writing code. Companies who have taken advantage of this shift have become successful [28].

CHAPTER 03

METHODOLOGY

This chapter discusses how the research was carried out. The requirement gathering steps. Detailed plan of building the visual development tool, its development stages, how the preliminary study was done, how its feedback impacted, the next version of the tool and how the efficiency of the test was done using a sample.

3.1 Procedure

A preliminary study was done in order to get an understanding about visual development tools that are currently available. Tools like mockup [29] were studied.
To build and find out the effect of visual development tool, the following steps were carried out;

1. Come up with the initial requirements specification for the initial release version 0.1 of the visual development tool. Analysis were done on the - mockups[29], Xcode IDE and Android studio to gather these requirements -
2. Gather feedback from the sample using a survey through interviews. The survey consisted of four questions. The questions revolved around the developer experience and what were features that the developers would like to have. The survey was given to 10 developers from the sample of 30.
3. Analyze the feedback from the survey and draft a second version of requirements for the visual development tool
4. Update the visual development tool adding the requirements.
5. Give the version 1.0 of the tool to the sample of 30 developers with 3 tasks to be performed.
6. Conduct the experiment to find out the efficiency and quality change of the developers due to the built tool.

To investigate the performance and quality of the developers compared between the given three different tasks with the tool and without the tool.

The three tasks varied from complexity. The tasks were as follows

1. Build the user interface of a contact form (Task 1) The contact form must have the following fields
    a. Name
    b. email
    c. Subject
    d. Description
2. Build the user interface of a blog (Task 2) The interface should display a list of card components that show the information of a blog article. The website should be responsive showing 3 articles in a row in desktop view, 2 in tablet view and 1 in mobile view.
3. Build the user interface of an email client (Task 3) The interface should have a side panel with navigation links to inbox, sent items etc. The right panel should contain the list of emails.

The hypothesis to test the efficiency of the coding was made as follows;

Task1

H02; There is no significant difference between development time using the tool and without the tool in Task 1
H1; There is a significant difference between development time using the tool and without the tool in Task 1

Task2

H02; There is no significant difference between development time using the tool and without the tool in Task 2
H1; There is a significant difference between development time using the tool and without the tool in Task 2

Task 3

H02; There is no significant difference between development time using the tool and without the tool in Task 3

H1; There is a significant difference between development time using the tool and without the tool in Task 3

3.2 Selecting a sample

For the initial development of the tool 10 software engineers from the sample of 30 developers were interviewed using a convenience sampling method. 30 software engineers that have 1 to 2 years of experience in frontend development were selected using a convenience sampling method. All the selected software engineers have worked with the vue js framework.

3.3. Data Collection Method

Data was collected in two instances. After developing the version 0.1 of the tool 10 developers were interviewed to gather feedback on the tool.

The data was gathered using the testing method and observation method after the developers completed each task. There were 3 tasks that the sample had to complete.

3.4. Data Analysis Method

The data gathered from the initial interview of the 10 developers were qualitatively analyzed.

The data from the testing method was compared using the paired t-test statistical procedure. The R package was used to carry out the paired t-test. Normality of the data were tested using Histogram, QQ plots and Shapiro Wilk test. The quality of the codes were analysed using discussion and summarizing the results.

3.5. Summary

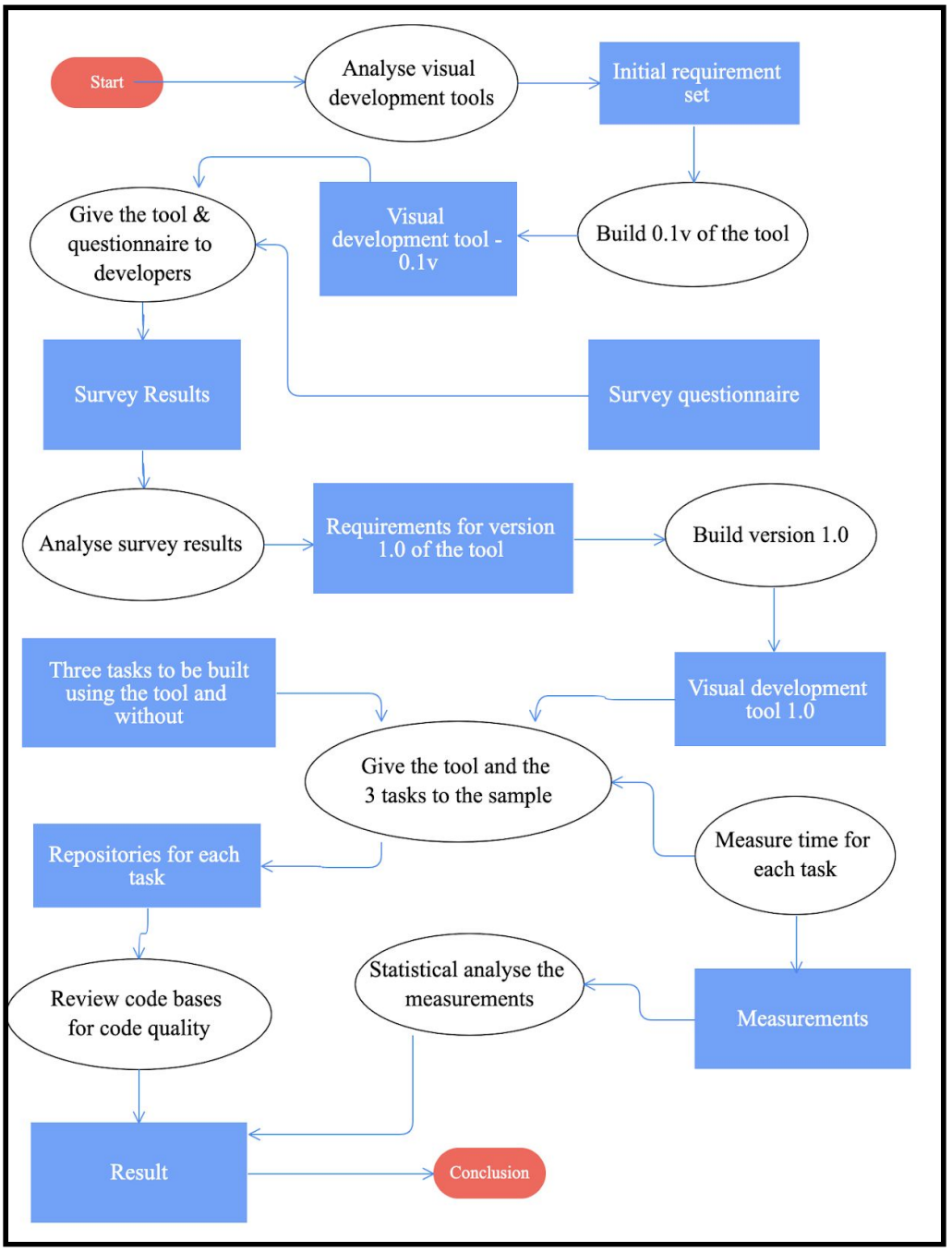


Figure 3.1 : The summary of the procedure

To summarize the above figure shows the summary of steps taken on this research. From initial requirements gathering to building the version 1.0 of the tool to finally analysing the measurement data.

CHAPTER 04

IMPLEMENTATION

This chapter discusses the initial requirements, development phases of the visual development tool. How it was incrementally built. How the surveys were created to gather user feedback. The survey results of the initial release of the tool. What were they and how these results were taken into account when upgrading the visual development tool.

4.1 Initial requirements

The initial requirements were the html tags (with form elements) that can be dragged and dropped to create a design. The supported tags list - div, image, table, input, select and checkbox. These were the same elements used by mockups.

4.2 Building the visual development tool

From the initial requirement gathering it was understood that the project needs to be broken down to smaller modules. The initial break down consisted of a client built using vue.sj and a server built with koa a node based framework.

4.2.1 Breaking down the components

From the design we broke down the page into smaller components. The breakdown was done according to the functionality. For example the form inputs were separated from the layout components.

4.2.1.1. Version 0.1 element break down

1. Layout
2. Canvas
3. Form inputs (button, radio, checkbox, select etc)
4. Div component
5. Image Component

6. Table Component

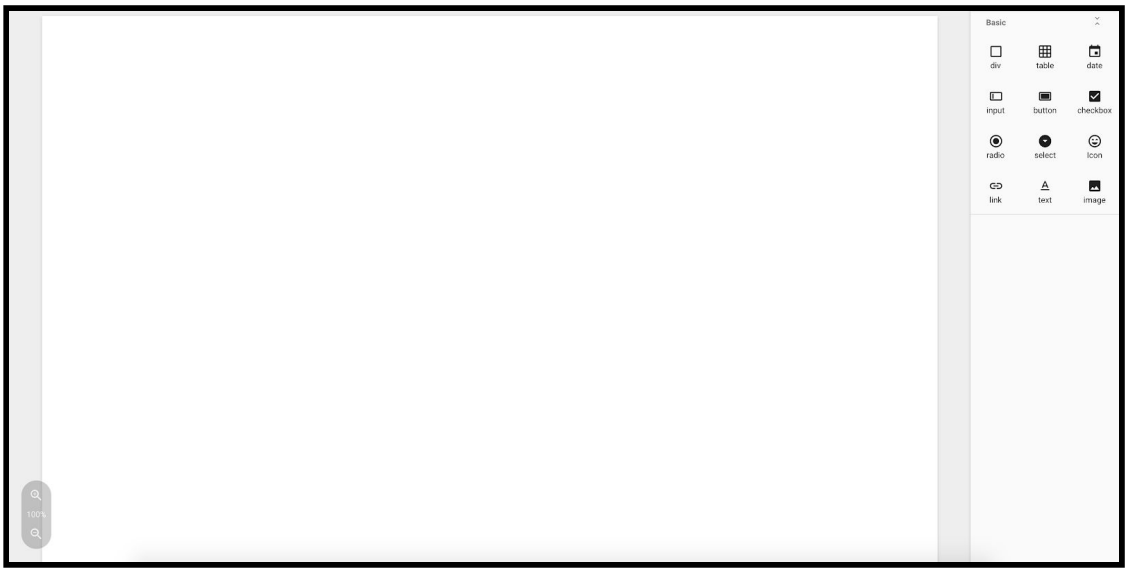


Figure 4.1 : Lay out of two panels

Layout was two panels. The left panel consists of the canvas where a user drags and drops the elements. The right panel consists of the elements listed above. This is part of the app that has been built.

The building blocks the div, table and image components were easiest to build. Even these basic components were hand coded.

head 1	head 2	head 3
data 1.1	data 2.1	data 3.1
data 1.2	data 2.2	data 3.2
data 3.1	data 3.2	data 3.3

Figure 4.2: Table component

Table component is the regular html table tag.When the software user downloads the app from the tool it becomes available for the user for further editing. The generated code looks as follows.

```
<table class="ryI7c9hL8">
  <tr class="SkxU7qc3IL">
    <th class="Bk-879qnL8">head 1</th>

    <th class="r1f8Qqc288">head 2</th>

    <th class="By7I75qhII">head 3</th>
  </tr>

  <tr class="SyN8Qqc38I">
    <td class="SkrUmcc3UU">data 1.1</td>

    <td class="rkUU7c528I">data 2.1</td>

    <td class="ryDLmcc2UU">data 3.1</td>
  </tr>

  <tr class="Sk_UQqqh8U">
    <td class="ryKIXcch8L">data 1.2</td>

    <td class="HycLmcq3IU">data 2.2</td>

    <td class="rJsLm993II">data 3.2</td>
  </tr>

  <tr class="Syn8Qcc38L">
    <td class="r1aLX9c2IU">data 3.1</td>

    <td class="Bk0Umc5hLU">data 3.2</td>

    <td class="HykeUm9qnIU">data 3.3</td>
  </tr>
</table>
```

Figure 4.3 : Generated code for table component

As you can see the user can easily modify the table component. Rest of the version one components take a similar approach. All are native html elements with styling

A group of elements inside a div built using the tool as follows.

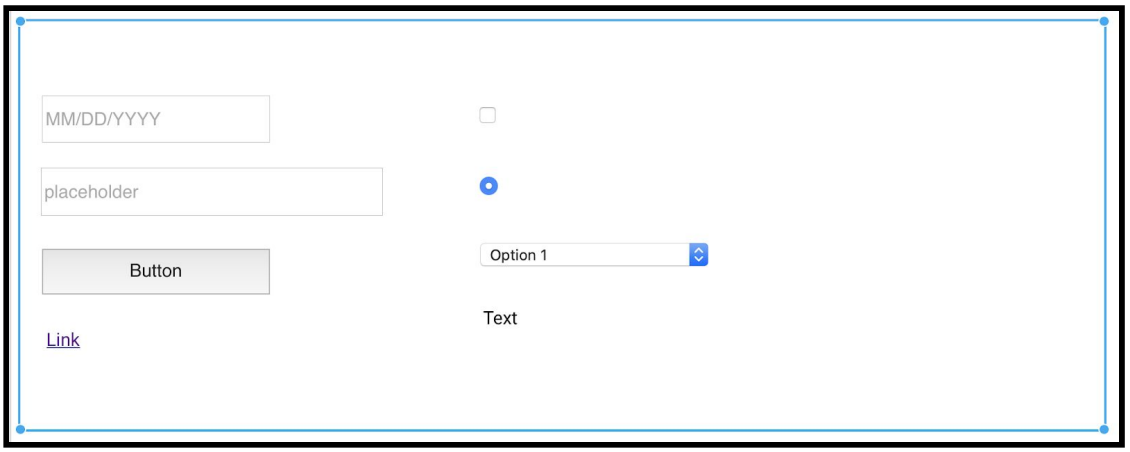


Figure 4.4 : A group of elements within in a div

The above design outputs the following code.

```
<div class="rJE61ohL8">
  <input class="rJWC1j28L" value="" type="date" placeholder="MM/DD/YYYY" overflow="hidden" text-overflow="ellipsis"/>

  <input class="H11Jej388" value="" placeholder="placeholder" overflow="hidden" text-overflow="ellipsis"/>

  <button class="ryayejh88">Button</button>

  <a class="ByNGxihII" href="#">Link</a>

  <input class="S1hXejhIU" type="checkbox"/>

  <input class="HJDVeo288" type="radio" name="default-group"/>

  <select class="H1KSxs38U">
    <option class="B1gFSgjh8I">Option 1</option>

    <option class="rybtrxohIU">Option 2</option>

    <option class="SJMKSxj28U">Option 3</option>
  </select>

  <span class="S1odeo28I">Text</span>
</div>
```

Figure 4.5: Generated code for the group of elements within a div

Note that the tool is smart enough to recognise that the components are inside the div. The components are also ordered in the way they have been placed. All components are given a unique class name so that the tool can style them individually and place them in the canvas in the right location. Generated css is not the ideal version that is possible.

Here is an example

```
<style scoped>
#ryxhWHoqI8 {
  --mdc-theme-primary: #673ab7;
  --mdc-theme-secondary: #f44336;
  --mdc-theme-background: #ffffff;
  position: relative;
  margin: auto;
  background-color: #ffffff;
  overflow: hidden;
  width: 100%;
  height: 100%;
}
.ryayejh88 {
  overflow: hidden;
  text-overflow: ellipsis;
  position: absolute;
  width: 200px;
  height: 40px;
  top: 200px;
  left: 19px;
}
.ByNGxihII {
  text-align: center;
  overflow: hidden;
  text-overflow: ellipsis;
  position: absolute;
  width: 61px;
  height: 40px;
  top: 271px;
  left: 8px;
}
.S1odeo28I {
  overflow: hidden;
  text-overflow: ellipsis;
  position: absolute;
  width: 150px;
  height: 25px;
  top: 251px;
  left: 407px;
```

Figure 4.6 : Generated styles for the above design

Note that the elements are positioned absolutely and given a top and left to position the element. This is not ideal at all. This makes it not possible to create responsive web design.

4.2.2 Building the Server

The server is built using Koa.js for node.js for node.js. Koa is another web system structured by the group behind Express, which expects to be a littler, increasingly

expressive, and progressively hearty establishment for web applications and APIs. By utilizing async capacities, Koa permits you to jettison callbacks and significantly increment blunder taking care of. Koa doesn't package any middleware inside its center, and it gives an exquisite set-up of techniques that make composing servers quick and pleasant.

Koa is a minimal framework that only provides a bare bone api. This makes it really fast which is an important feature to use this framework.

The application only consist of few routes listed below;

1. router.post('/get-access-token', getAccessToken)
2. router.post('/project', saveVueProject)
3. router.get('/project', getVueProject)
4. router.post('/generate', generate)

4.2.3. Download feature

The generate route calls the generate function with the given parameters. The logic of generating the code has been speratated to a separate file called generate.js. The user can either download a .zip or a .gg format. The generate function requires the context of the request body and the root directory. This download does not include the node_modules directory so the user has to install packages after downloading.

```
106  /**
107   * Generates a full vue application from a scaffold project,
108   * plus the app definition sent in the ctx.request.body
109   *
110   * @param {object} ctx : KoaContext object
111   * @param {object} ctx.request.body : Body of the POST request
112   *
113   * @return {zipFile} : A zip file with the generated application should be served
114   *
115   * @see {@link http://koajs.com/#context|Koa Context}
116   */
117  async function generate (ctx) {
118    try {
119      let zipFile = await generator(ctx.request.body, ROOT_DIR)
120      if (zipFile) {
121        console.log('> Download -> ' + zipFile)
122        ctx.response.status = 200
123        ctx.response.type = 'zip'
124        ctx.response.body = fs.createReadStream(zipFile)
125      }
126    } catch (e) {
127      console.error('\n> Could not complete the project generation...\n' + e)
128      process.exit(1)
129    }
130  }
```

Figure 4.7 : Example Code for Download Features

4.2.4. Summary

This set of features were the version 0.1 of the visual development tool. The client built with vue js and the backend built with kor.js. The drag and drop elements were simple html tags with custom styling. The generated code is stored in the pages directory of the project. Users are easily able to modify the code for the finer needs that may be there. The version 0.1 was given to the sample as with few basic tasks.

4.3 Preliminary Survey

Survey contained the following questions.

1. What are your thoughts of using a visual development tool?
2. What did you enjoy about the tool?
3. What are features that you used most?
4. What additional features would you like to see in the tool?

Developers were very happy to be using the tool. Most of the feedback showed that they are really happy that they don't have to write code to do the repetitive tasks. The most used features were the form elements. They have really enjoyed dragging and dropping elements to the canvas.

From the results we got from the above surveys main points were addressed for the next version of the tool. Further additional features were added by the author. The list of requirements for the version 1.0 is as follows

1. Undo / redo feature
2. Preview feature
3. Card component
4. Appbar component
5. Sidebar component.
6. Authentication (github)
7. Pages - Create multiple pages

4.4. Building version 1.0

As per requirements much of them revolved around adding more reusable complex components (card component, appbar, sidebar etc). Building reusable components that are accessible with high performance and flexibility is very time consuming. Further there are libraries that provide said components that are both performant and flexible. Hence without trying to code the above set of components it was a smart idea to use an existing vue user interface library. There existing vue ui libraries that are matured and have a lot of community support. Plus reduces the complexity of the visual development tool. Hence it was decided to use a vue UI library. There are few libraries that have been considered

1. Element UI
2. Bootstrap Vue
3. Quasar Framework
4. Vuetify

From the above list vuetify was picked due to the following reasons.

Vuetify is a Vue UI Library with beautifully handcrafted Material Components. No design skills required — everything you need to create amazing applications is at your fingertips. (https://vuetifyjs.com/en/)

The reason for using a UI library is as follows

1. Full Accessibility and Section 508 support
2. Server Side Rendering support
3. Long-term Support
4. RTL support
5. Built according standards

4.4.1. Integrating the vuetify components to the visual development tool.

Figure 4.8 : Layout with new material component

The new components were moved to a material design section in the sidebar. Apart from the suggested components additional components were added as well

List of additional components added

1. Progress Bar
2. List component

3. Drawer component

These extra components were easy to add since they too were available in the vuetify (vue ui) library.

4.4.2. Design created by the newly added components

After having the idea of interviewers, the design was built up with newly added components as follows;

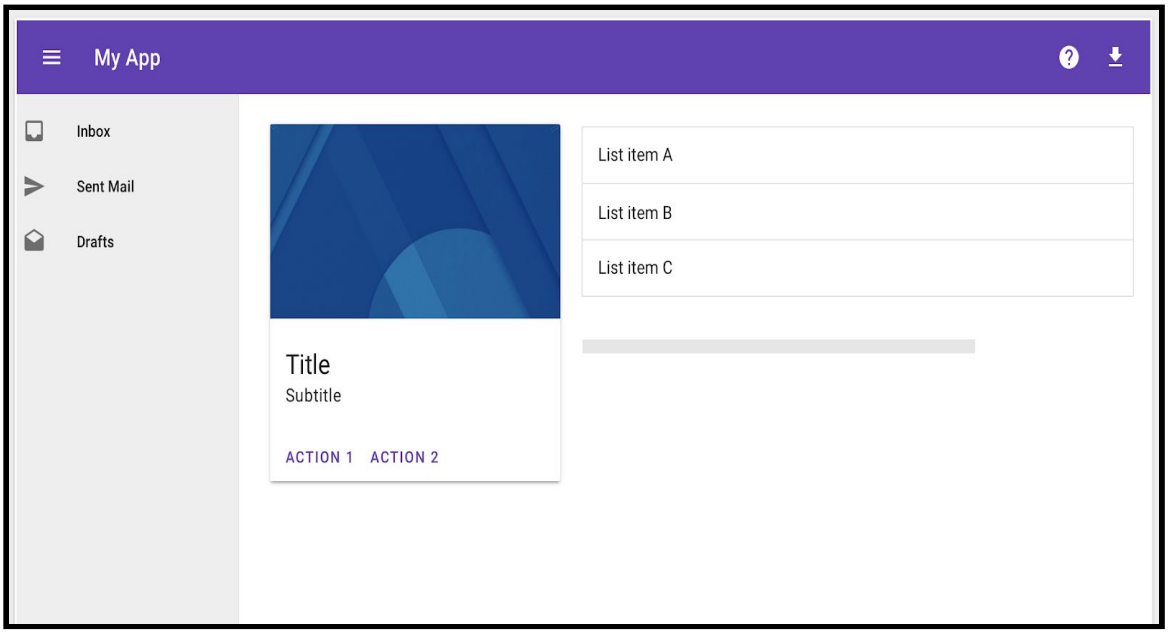


Figure 4.9 : Design built with the newly added components

The design consists of the app bar component, side panel, card component, list component and the progress bar. The card components consist of two action buttons where you can attach functions. Same goes to the side panel and the app bar, they too contain action buttons.

The generated code for the above user interface.

```
<mdc-drawer class="S1Ed087T8L" open toolbar-spacer persistent toggle-on="toggledrawer">
  <mdc-drawer-list class="HJSd0I76L8">
    <mdc-drawer-item class="S1Lu08QaU8" start-icon="inbox">Inbox</mdc-drawer-item>

    <mdc-drawer-item class="S1vd0IQpLL" start-icon="send">Sent Mail</mdc-drawer-item>

    <mdc-drawer-item class="Hyu_RUm6LI" start-icon="drafts">Drafts</mdc-drawer-item>
  </mdc-drawer-list>
</mdc-drawer>
<mdc-top-app-bar class="BJfZJPXpUI" title="My App" event="toggledrawer">
  <mdc-top-app-bar-action class="HyXW1vmTL8" icon="help"></mdc-top-app-bar-action>

  <mdc-top-app-bar-action class="Sk4W1DQaLI" icon="file_download"></mdc-top-app-bar-action>
</mdc-top-app-bar>
<mdc-card class="ByB2gPmpLU">
  <mdc-card-media class="BJUngv7p8L" height="160" src="https://material-components-web.appspot.com/images/16-9.

  <mdc-card-header class="B1wheDXpLL" title="Title" subtitle="Subtitle"></mdc-card-header>

  <mdc-card-actions class="S1_3ePQa8U">
    <mdc-card-action-button class="HyF2ew7T88">Action 1</mdc-card-action-button>

    <mdc-card-action-button class="Sy93evmpIU">Action 2</mdc-card-action-button>
  </mdc-card-actions>
</mdc-card>
<mdc-list class="SkQzbvXa8U" bordered>
  <mdc-list-item class="B1NMWvXpUI">List item A</mdc-list-item>

  <mdc-list-item class="BkSG-PXp8I">List item B</mdc-list-item>

  <mdc-list-item class="rJ8MbvmpLI">List item C</mdc-list-item>
</mdc-list>
<mdc-linear-progress class="S1lkXvQp8U" open indeterminate progress="0.25" buffer="0.5"></mdc-linear-progress>
```

Figure 4.10 : Generated code for design built with the newly added components

The generated code places the side panel component defined by the tag mdc-drawer at the top. Which consists of smaller link components that are defined by mdc-drawer-item. Following is the app bar component which consists of two actions: help and download. Next is the card component and the list component. All of these components come from the vuetify library that was integrated into the project. All of these components are accessible, performant and flexible. Hence the developer can easily manipulate the code to fit exactly his/her needs.

4.4.3. Authentication

It was clear that an authentication feature was required to manage the different projects a user might have. The authentication was built around github. The reason behind the choice is to use the github api to store the generated code directly in a github repository. Hence github was integrated as a sign on.

4.4.4 Pages

In version 0.1 of the tool it was only possible to create a single page from the tool. The developers needed a feature where they can save different pages. For example home page, about page, contact page etc. To build we created an array to store the page objects in vue state. Project state was created to store all the pages state and other related project state.

```
13 lines (11 sloc)  241 Bytes
1  import shortid from 'shortid'
2  import newPage from './pageFactory'
3
4  function newProject (title) {
5    return {
6      id: shortid.generate(),
7      title: title,
8      components: [],
9      pages: [newPage('Home', '/')]
10   }
11 }
12
13 export default newProject
```

Figure 4.11 :Example code for create pages

4.4.5. Undo / Redo feature

The user is able to undo the previous actions. The actions performed are stored in the vue.js state as a stack. The stack is popped during an undo action. The redo actions work the same. Storing the action stack in vue state makes it not persist among different sessions the only downside to this is that it does not persist between different sessions.

Apart from storing in vue state IndexedDB was used to store the actions performed by the user. This allows the state of the work done by a developer to be saved for another session. This makes the app run a little slower but that is negligible considering the functionality of preserving state among two different states.

IndexedDB was used over web storage because it allows storing larger amounts of data than web storage. IndexedDB is the API for client-side storage that was due to its capability of storing large amounts of structured data, including files/blobs. This API uses indexes to enable high-performance searches of this data.

[https://developer.mozilla.org/en-US/docs/Web/API/IndexedDB_API]

```
55    methods: {
56      undo () {
57        if (this.canUndo) {
58          this.undone.push(this.done.pop())
59          let undoState = this.done[this.done.length - 1]
60          this.$store.replaceState(cloneDeep(undoState))
61          this.$root.$emit('rebaseState')
62          this.updateCanRedoUndo()
63        }
64      },
65
66      redo () {
67        if (this.canRedo) {
68          let redoState = this.undone.pop()
69          this.done.push(redoState)
70          this.$store.replaceState(cloneDeep(redoState))
71          this.$root.$emit('rebaseState')
72          this.updateCanRedoUndo()
73        }
74      },
75
76      updateCanRedoUndo () {
77        this.$store.commit(_toggleCanUndo, this.canUndo)
78        this.$store.commit(_toggleCanRedo, this.canRedo)
79      }
80    }
```

Figure 4.12: Example code for create undo/redo features

The undo / redo feature has been built as a mixin since it is a common functionality

that can be called from anywhere. The current implementation is called from many places, for example the app bar. By calling `@click="$root.$emit('redo')"`

4.4.6. Preview Feature

The user is given the option to see how the page looks on different sized devices. Currently supports desktop view, tablet view and mobile view.

4.4.6.1. Mobile view preview.

The developer can preview both for the desktop version and the mobile version.

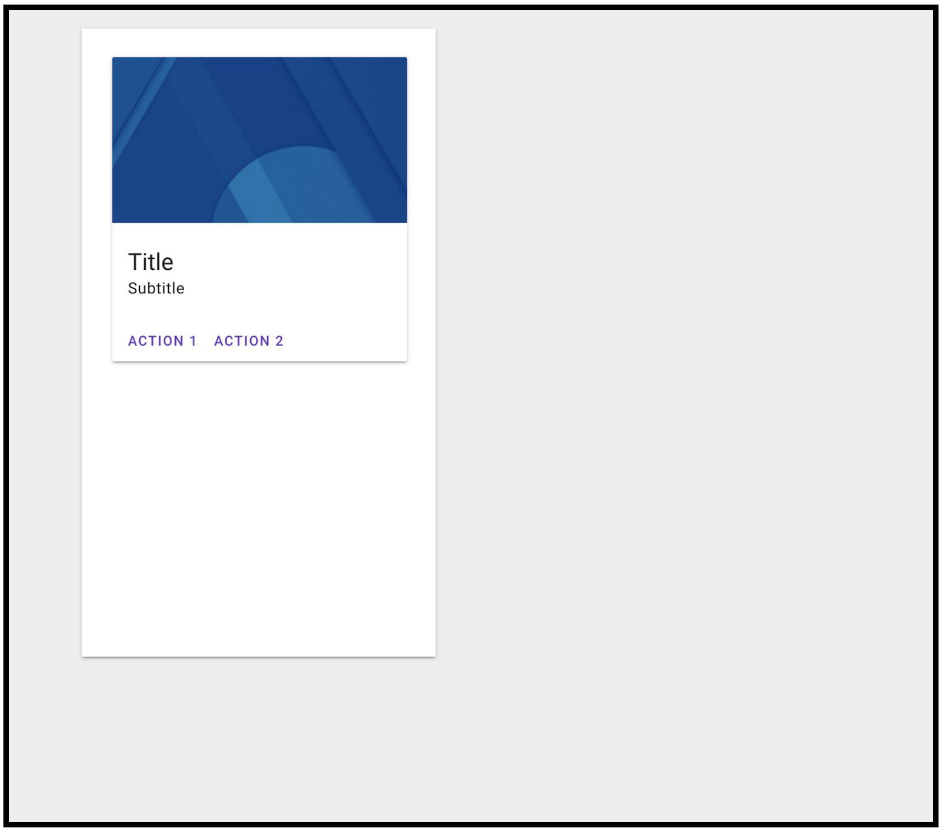


Figure 4.13 : The mobile version of the tool

4.5. Final Tool

With the above features the version 1.0 of the visual development tool was completed. The final result of the tool looked as follows.

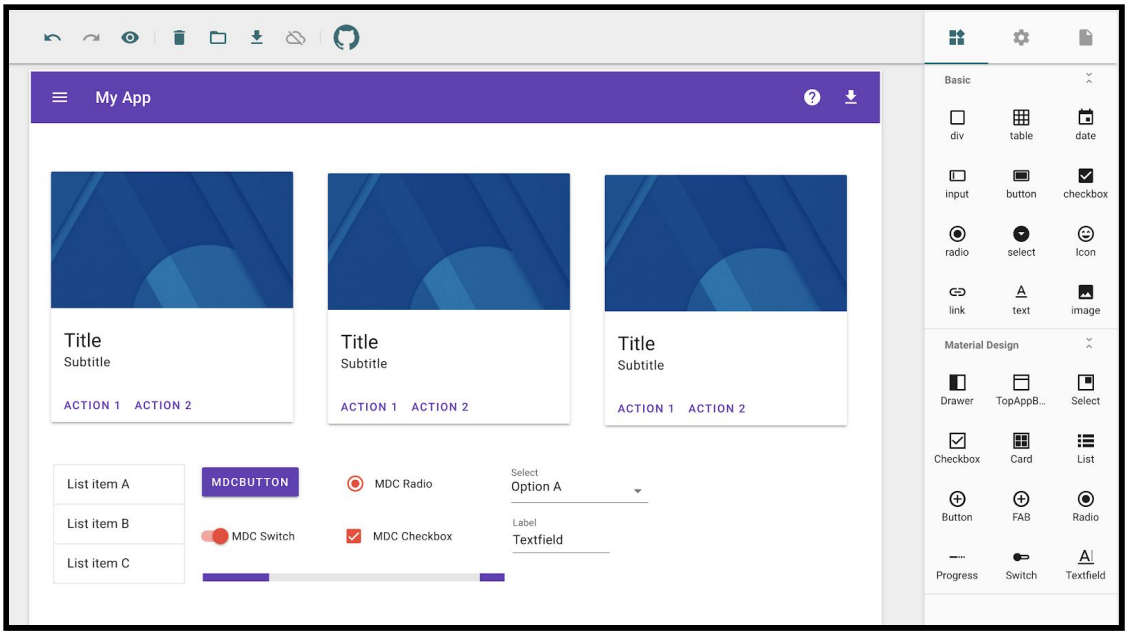


Figure 4.14 : The final visual developed tool with example design

The final developed tool consisted of 12 simple components and 12 vue material components. The tool also gives an undo, redo feature preview feature and a mobile view. Providing enough features to the developer to carry out the three tasks that will be given in the testing phase.

The tool was given to the sample of 30 developers. After the completion of their tasks the coding standards were analysed.

4.6. Summary

The visual development tool follows a simple client server architecture. The following diagram shows the main components involved in building the tool. Github is used for both authentication and storing the generated code. Koa server contains API that both connect with github and the vue client.

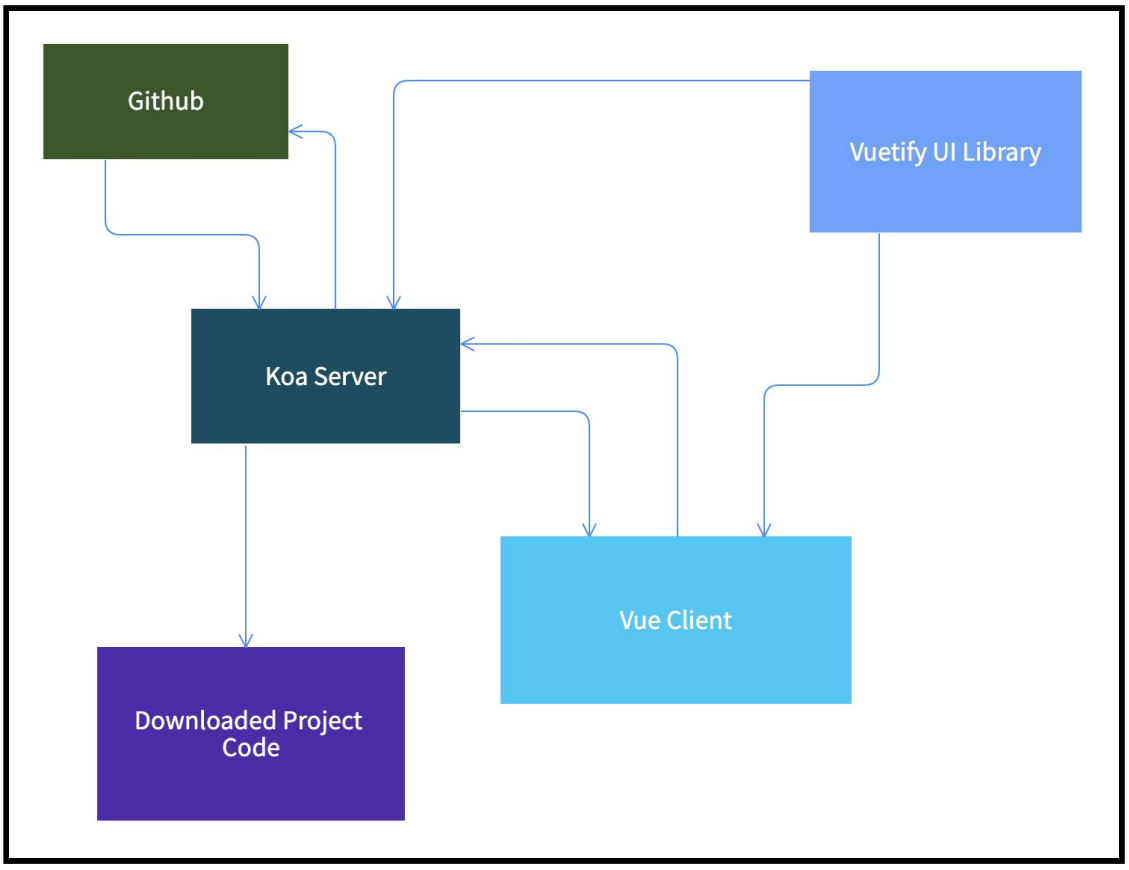


Figure 4.15 : Summary of the final product

Vuetify user interface library is the component library that contains the components that are used in the tool. It is both connected to the server and the vue client app. The final output of the tool is the downloaded project code.

CHAPTER 05

EVALUATION

This chapter discusses the results of the research data gathered from the sample of 30 developers. The chapter analyzes the results to understand the data. How the tool has performed. Has it provided enough support for the developers to write better software more efficiently. Further qualitative analysis of the code written by the developers using the tool and without will be analysed. This analysis will show if the code is actually in a higher standard than before.

5.1. Results of the initial survey

An initial survey was carried for the 0.1 version of the visual development tool to confirm the features built and to understand what other requests that the software developers might give to improve the tool. This was the first validation of the tool.

The following features were requested by most of the developers

1. Undo / Redo feature
2. Preview feature
3. Card component
4. Appbar component
5. Toolbar component

5.2. Analysis of the feedback.

Apart from the Undo / Redo feature and preview feature all other requested features were for more complex components to be added to the tool. Software engineers have more need for more higher level components than granular ones.

5.3. Analysis timing of coding of the developers using three different task

The three different tasks were given to complete using the tool and without for the thirty developers. The normality of the data were checked using Histogram, QQ plots and Shipro-Wilk tests(Appendix A). Since the data were normally distributed, the

Paired T-test in parametric tests was used to check whether there is an effect on a visual tool for coding efficiency. The comparison between the test using tool and without tool was analyzed at the 95% confidence interval. The R-package 2019 was used to analyse the data. All the codes, normality results and test results were attached in the Appendix B.

The results of three different tasks were summarized in Table 4.1 and then they were discussed separately.

Table 5.1: Summary Result Table

Tasks	Mean(Timing in Minutes)		P Value
	Without Tool	With Tool	
1	20.23	21.16	0.308
2	59.6	39.3	8.473e-14
3	74.03	65.33	1.882e-11

5.3.1. Analysis of Task 1 - Simplest task given to the developer sample.

H01; There is no significant difference between development time using the tool and without the tool in Task 1

Table 5.2 : Task 1 test results

Paired t-test results	
t-value	-1.0377
Df(degree of freedom)	29
Alternative hypothesis:	true difference in means is not equal to 0 95 percent
Confidence interval:	-2.7729513 0.9062847
Sample estimates: mean of the differences :	-0.9333333

The mean values of using without tool and with tool were 20.23min and 21.16min respectively for Task 1. When comparing the means, coding without the tool shows the lower mean time than the coding with the tool.

Also, the p value(0.308) of task 1 was greater than 0.05(P value> 0.05) which indicated that the null hypothesis was not rejected at 0.05 significance level. Therefore, there is no significant time difference shown for coding using the tool and without tool. There is no significant impact on efficiency using the tool or without the tool for codes like task 1.

5.3.2. Analysis of Task 2 - Moderately complex task given to the developers

H02; There is no significant difference between development time using the tool and without the tool in Task 2

Table 5.3 : Test results Task 2

Paired t-test results	
t value	13.211
df	29
p-value	8.473e-14
alternative hypothesis:	true difference in means is not equal to 0 95 percent
confidence interval:	17.15722-23.44278
estimates: mean of the differences :	20.3

The mean values of using without tool and with tool were 59.6 min and 39.3 min respectively for Task 2. When comparing the means, coding with the tool shows a lower mean time than coding without the tool.

The p value(8.473e-14) of task 2 was less than 0.05(P value < 0.05) which indicated that the null hypothesis is rejected at 0.05 significance level. Therefore, there is a significant positive time difference shown between the coding using the tool and without tool. The efficiency is higher using the visual development tool than without using the tool for task 2

5.3.3 Analysis of Task 3 - Most complex task given to the developer sample.

H03 : There is no significant difference between development time using the tool and without the tool in Task 3

Table 5.4: Test results of Task 3

Paired t-test results	
t	10.562
df	29
p-value	1.882e-11
alternative hypothesis:	true difference in means is not equal to 0 95 percent
confidence interval:	7.015314 10.384686
sample estimates: mean of the differences	8.7

The mean values of using without tool and with tool were 74.03 min and 65.33 min respectively for Task 3. When comparing the means, coding with the tool shows a lower mean time than the coding without the tool.

The p value(1.882e-11) of task 3 was less than 0.05(P value< 0.05) which indicated that the null hypothesis is rejected at 0.05 significance level. Therefore, there is a

significant positive time difference shown between the coding using the tool and without tool. The efficiency is higher using the visual development tool than without using the tool for tasks like Task 3.

Comparing the mean time of tasks shows an advantage of using the tool for both task 1 and 2. But for task 1 developers took less time without the tool. This can be attributed to developers being unfamiliar with the tool and couldn't take much advantage of the tools built in components because the given task (task 1) does not need the components. The statistical analysis also shows there is no significance between using the tool and not for task 1.

For both tasks 2 and 3 the mean values show developers took less time to complete the tasks with the tool. This may be due to the developers being familiar with the tool after completing task 1 and the developers were able to use the built in components in the tool to complete task 2 and 3. There was a statistical significance to both the results in task 2 and 3. The analysis also tells that benefits gained from the tool to develop task 3 was higher than task 2.

5.4. Analysis of coding standards of developers

The code that has been written by each developer was analysed with and without the tool. The analysis was done with regard to readability, indentation and code reuse. The readability, indentation and code reuse. all improved when using the visual development tool. The down side of the code quality that was generated by the tool was the styles. Styles were very unreadable and didn't follow the standards. Further the styles didn't support responsive design. The generated id and class names from the tool also were unreadable. A proper pattern must be used to make the styles more approachable

5.5 Analysis of issues and suggestion by developers given for the tool

Many of the developers gave positive feedback for the tool. They mentioned learning how to use it was very intuitive. There were also suggestions to improve the tool. That

They are listed below.

1. Ability to edit the code from within the tool.
2. Suggestions
3. Documentations
4. Increase the performance of the tool
5. Styles generated by the tool are not ideal making it difficult to edit.
6. Responsive design is not possible through tool
7. More built in components
8. Support multiple libraries - like reactjs.

The developers mentioned that with more time learning the tool and how it works they will be able to work faster with the tool. But the tool will be difficult to use with an existing project. Since the standards and rules followed in that project will be different. Further they mentioned about the generated styles. Styles were very difficult to work with and they had to remove the generated styles and write it again.

5.6. Understanding the results

The three tasks given to the sample varied in complexity. The first task was straight forward. The developers were able complete the task without the tool quicker than with the tool.

In developing the 2nd task they used the tools inbuilt card component which cut down the time in development helping them to build the user interface in a much shorter time compared to without the tool.

Task 3 there was even more significant positive result for using the tool than in task 3. Proving that the more complex the task the more support the tool gives. This is ofcourse if the tool provides the components the task needs.

The quality of the code was measured in the following criteria - readability, indententaton, unused import statements, comments, code reuse. This analysis both upsides and downsides for the tool. The styles generated by the tool was the main downside and needs more thought and development. The tasks done without the tool

had better styling and were more readable. The code reuse of the developers differed a lot by each individual. Overall the tool helped to improve the quality of the programs written.

CHAPTER 06

CONCLUSION

In this thesis, we have shown how to design and build a visual development tool that can generate vue.js code from a design created within the tool, Do a statistical analysis on the time measurements of 30 developers building 3 tasks with and without the visual development tool.

The research in this thesis was guided by 4 main objectives. To develop a visual development tool to enhance performance and quality of work of a vue developer. To explore the effect of the build tool on developing performance and quality of work of a developer. To understand if it is possible to have better software quality standards by using a tool. Finally to compare the coding efficiency of developers using the tool and without the tool.

Building the visual development tool and identifying the efficiency and quality was a success. According to the result for task one even though there wasn't a statistical significance. It can be said that the tool does not give an advantage when building a simple user interface. It can be rationalized the overhead of using and understanding the tool to build a simple user interface is not worthwhile. Which is shown from the results for task one having a lower mean time when the tool was not used.

Results for tasks two and three both have rejected the null hypothesis meaning it has given support for the developers when building task two and three. The developers tool much longer to complete task two and three compared to task one. They were able to use much more components in the visual development tool.

The research finds that for tasks that have a complex user interface the visual development tool will benefit the developer to be more efficient and write better code. This was not the case for simple user interfaces. Task one building a contact form showed that using the tool took more time from the developers than without. It was understood that the tool didn't give much advantage and it was just an extra layer that the developer had to go through.

The research has only proven that the developed tool only helped work for the tasks that were given to them during the study. It also confirms that software quality standard have been improved due to the use of the tool.

CHAPTER 07

FUTURE WORK

It is worthwhile to carry on with this project. Include more complex components that are used by software engineers. Further provide an easy way for software engineers to contribute to the software. This can be achieved by moving the drag and drop components to a separate open source repository. To build an API that allows the user to integrate their own components from their design system to the visual development tool will make way for more vast use cases that the product will be beneficial.

Allow for better style generation this can be achieved by using layout components that are built into the tool. The current version of the tool doesn't support responsive design but with predefined layout components this can be achieved. Further styled components package can be used to generate the styled components. Meaning the component itself be styled which will remove the need for external style sheets.

Integrate other user interface libraries like Element UI, Bootstrap Vue, Quasar Framework.

Give the choice to the developer to generate a vue.js app or a nuxt app. Nuxt.js is the recommended vue application framework. Nuxt applications provide more features pre built like server side rendering, automatic code splitting etc. Further it is the framework that is used by big companies that are using vue js.

Further this tool can be integrated into frameworks like blazor. Blazor provides set of components built using C# also it allows to write your client and server code both in C#.

Bootstrap is also another very popular framework to build web apps. Bootstrap lets users build mobile first responsive web apps. It also includes a grid system where the components can be properly aligned. Solving the issue of the generated code not being responsive. Further bootstrap has its own readable classes. These classes too can be used by the tool. Making styles more readable and familiar to developers.

Making the tool as an extension to existing code editors like vscode. This will allow

the tool to reach a broad range of users through the extension marketplace. This also adds the benefit of not having the cycle of downloading the code base and running it in users local machines. The code can be generated in the local machine and then can be edited using the existing editor. The prime editor to build this would be vscode since it is built using electron.

REFERENCES

1. Akbar, M. A., Sang, J., Khan, A. A., Shafiq, M., Hussain, S., Hu, H., . . . Xiang, H. (2017). Improving the quality of software development process by introducing a new methodology–AZ-model. *IEEE Access, 6*, 4811-4823.
2. Amalfitano, D., Fasolino, A. R., Tramontana, P., De Carmine, S., & Memon, A. M. (2012). *Using GUI ripping for automated testing of Android applications.* Paper presented at the 2012 Proceedings of the 27th IEEE/ACM International Conference on Automated Software Engineering.
3. Benavides, D., Segura, S., Trinidad, P., & Ruiz-Cortés, A. (2005). *Using Java CSP solvers in the automated analyses of feature models.* Paper presented at the International Summer School on Generative and Transformational Techniques in Software Engineering.
4. Campos, P., & Nunes, N. (2007). Towards useful and usable interaction design tools: CanonSketch. *Interacting with Computers, 19*(5-6), 597-613.
5. Cardello F., (2020).14 no code apps to help build your next startup. Available : https://webflow.com/blog/no-code-apps
6. *Cattell, R. (1980).* Automatic derivation of code generators from machine descriptions. *ACM Transactions on Programming Languages and Systems (TOPLAS), 2(2), 173-190.*
7. Engineering, A. S. R. D. (2018). *8 Ways to Improve Software Testing through Planning, Work Environment, Automated Testing, and Reporting* [Online]. Available:https://www.altexsoft.com/blog/engineering/8-ways-to-improve-software-testing-through-planning-work-environment-automated-testing-and-reporting/ [Accessed].
8. Frenken, M., Willemse, T. A., Océ, L. v. G., Bunte, O., & Denkers, J. (2019). Code generation and model-based testing in context of OIL
9. Froehlich, J., & Dourish, P. (2004). *Unifying artifacts and activities in a visual tool for distributed software development teams.* Paper presented at the Proceedings. 26th International Conference on Software Engineering.
10. Huang, S., Gohel, V., & Hsu, S. (2007). *Towards interoperability of UML tools for exchanging high-fidelity diagrams.* Paper presented at the Proceedings of the 25th Annual ACM international Conference on Design of

Communication.

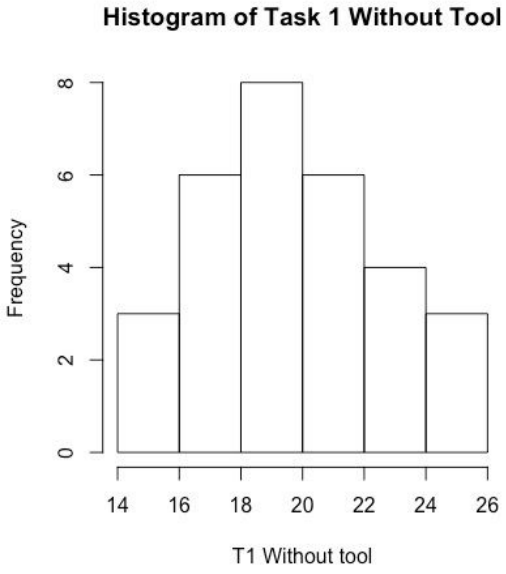
11. Jackson, D. (2020). 2019 Software Development Price Guide & Hourly Rate Comparison. Retrieved from https://www.fullstacklabs.co/blog/
12. Jacques, N. (2019). *Jump Start Vue. js*: SitePoint.
13. Jørgensen, A. H., & Myers, B. A. (2008). User interface history. In *CHI'08 Extended Abstracts on Human Factors in Computing Systems* (pp. 2415-2418).
14. Kitchenham, B., & Pfleeger, S. L. (1996). Software quality: the elusive target [special issues section]. *IEEE software, 13*(1), 12-21.
15. Kite Agency,. (2019), Software Development Hourly Rates. Retrieved from: https://www.kite.agency/software-development-hourly-rates
16. Koetsier, J. (2016). Evaluation of JavaScript frameworks for the development of a web-based user interface for Vampires. *Informatica—Universiteit van Amsterdam*.
17. Pancheri, R. (2015). Design and Implementation of a Graphical User Interface for Elektra.
18. Rainer, G., & Miller, E. K. (2000). Effects of visual experience on the representation of objects in the prefrontal cortex. *Neuron, 27*(1), 179-189.
19. Robbins, J. E. (1999). *Cognitive support features for software development tools*: University of California, Irvine.
20. Rugaber, S., Hemel, Z., & Stirewalt, K. (2013). *Live logic programming.* Paper presented at the 2013 1st International Workshop on Live Programming (LIVE).
21. Samlaus, R. (2015). *An Integrated Development Environment with Enhanced Domain-Specific Interactive Model Validation.* Linköping University Electronic Press
22. Shabiralyani, G., Hasan, K. S., Hamad, N., & Iqbal, N. (2015). Impact of Visual Aids in Enhancing the Learning Process Case Research: District Dera Ghazi Khan. *Journal of education and practice, 6*(19), 226-233.
23. Song, J., Zhang, M., & Xie, H. (2019). Design and Implementation of a Vue. js-Based College Teaching System. *International Journal of Emerging Technologies in Learning (iJET), 14*(13), 59-69.
24. Steinbeck, C., Han, Y., Kuhn, S., Horlacher, O., Luttmann, E., & Willighagen, E. (2003). The Chemistry Development Kit (CDK): An open-source Java

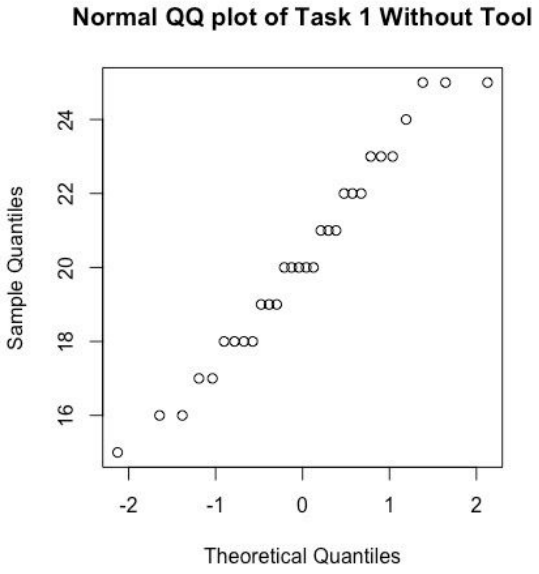
library for chemo-and bioinformatics. *Journal of chemical information and computer sciences, 43*(2), 493-500.

25. Timbol, M. (2001). Development system with visual design tools for creating and maintaining Java Beans components. In: Google Patents.
26. Vogel-Heuser, B., Witsch, D., & Katzke, U. (2005). *Automatic code generation from a UML model to IEC 61131-3 and system configuration tools.* Paper presented at the 2005 International Conference on Control and Automation.
27. Wallis, G., & Bülthoff, H. (1999). Learning to recognize objects. *Trends in cognitive sciences, 3*(1), 22-31.
28. webflow., (2020).Break the code barrier. Retrieved from https://webflow.com/responsive-website-builderr
29. https://moqups.com
30. https://vuetifyjs.com/en/

Appendix A : Test of Normality Results and test Results

Normality in Task 1: Without the tool

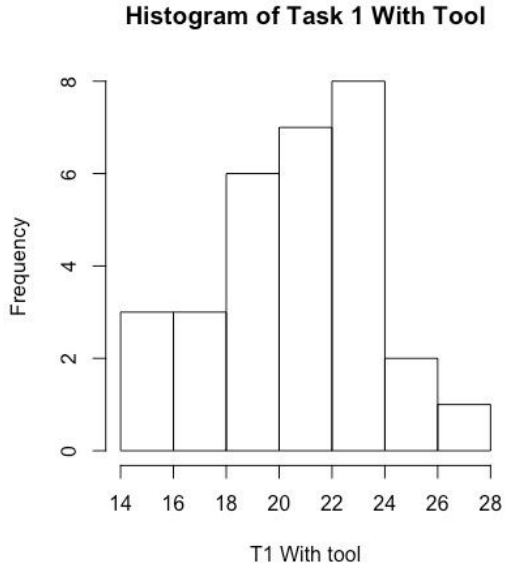


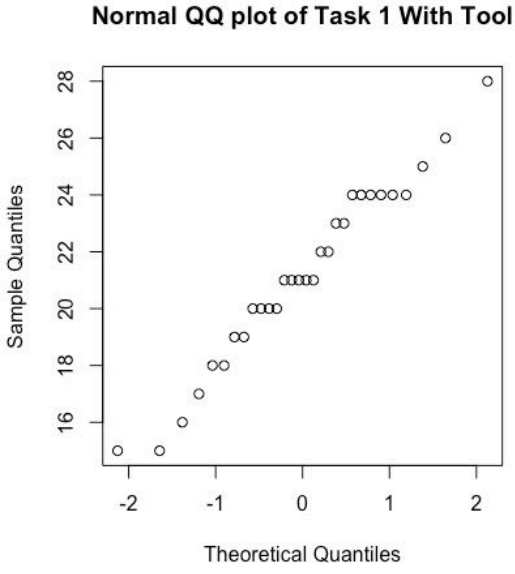


Shapiro-Wilk normality test

data: Time$T1.without.tool
W = 0.96747, p-value = 0.4725

Normality in Task 1: With the tool





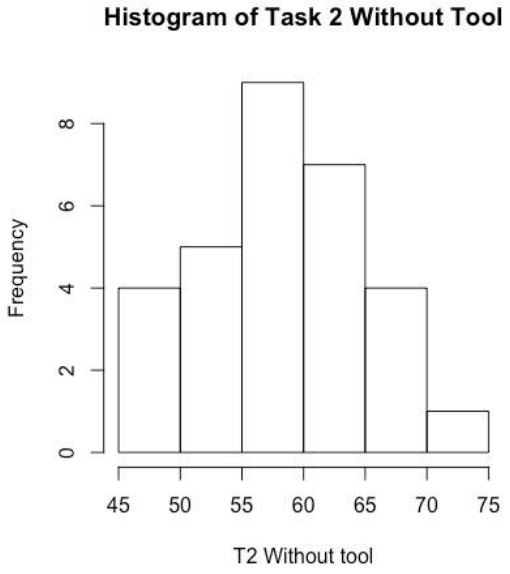
Shapiro-Wilk normality test

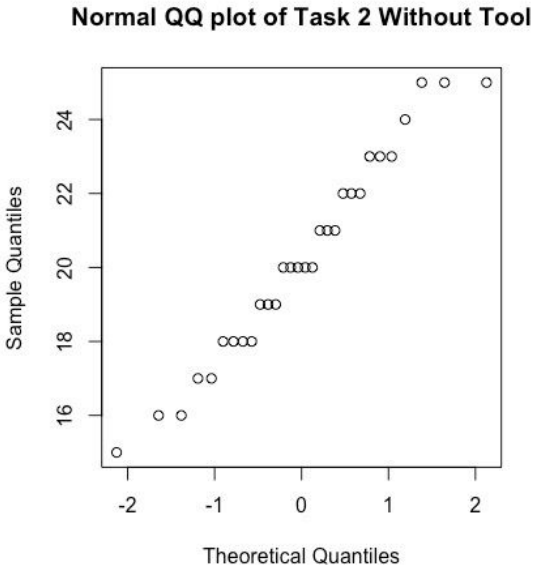
data: Time$T1.with.tool
W = 0.97286, p-value = 0.6201

Test Results : Paired t-test(Task 1)

data: Time$T1.without.tool and Time$T1.with.tool
t = -1.0377, df = 29, p-value = 0.308
alternative hypothesis: true difference in means is not equal to 0 95 percent confidence interval: -2.7729513 0.9062847
sample estimates: mean of the differences: -0.9333333

Normality in Task 2: Without the tool

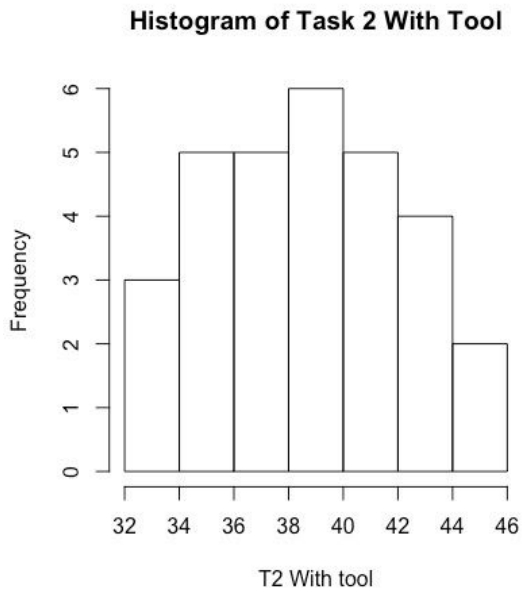


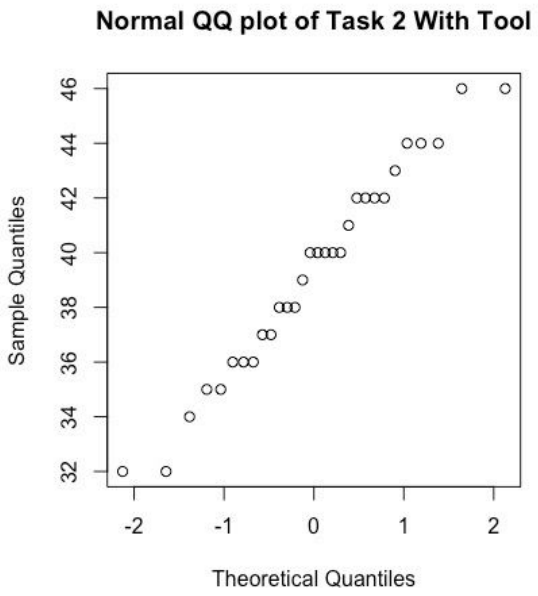


Shapiro-Wilk normality test

data: Time$T2.without.tool
W = 0.98223, p-value = 0.8813

Normality in Task 2: With the tool





Shapiro-Wilk normality test

data: Time$T2.with.tool
W = 0.97231, p-value = 0.6041

Test Result: Paired t-test(Task 2)

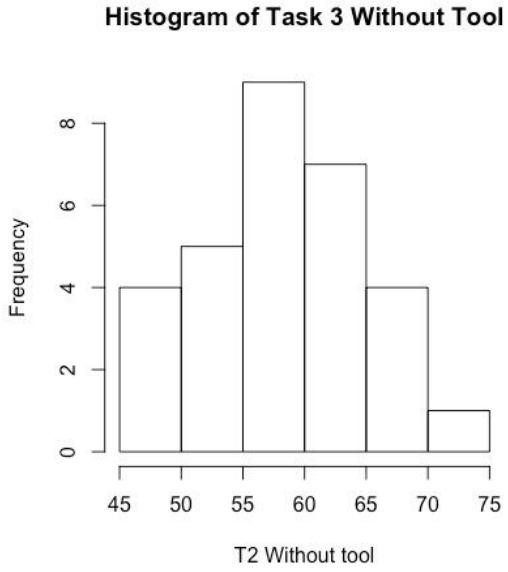
data: Time$T2.without.tool and Time$T2.with.tool

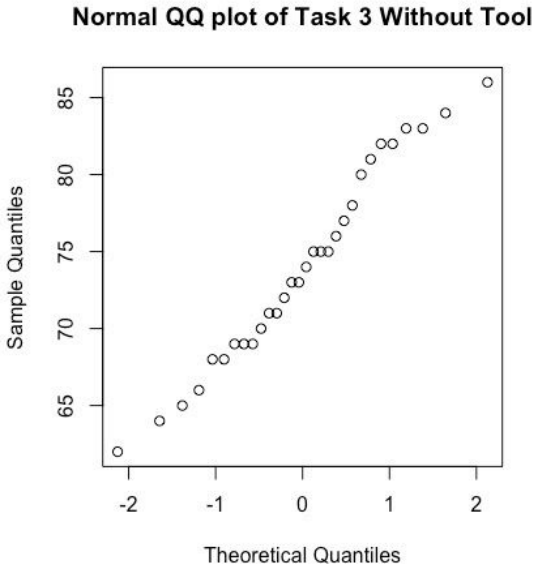
t = 13.211, df = 29, p-value = 8.473e-14

alternative hypothesis: true difference in means is not equal to 0 95 percent confidence interval: 17.15722 23.44278

sample estimates: mean of the differences 20.3

Normality in Task 3: Without the tool

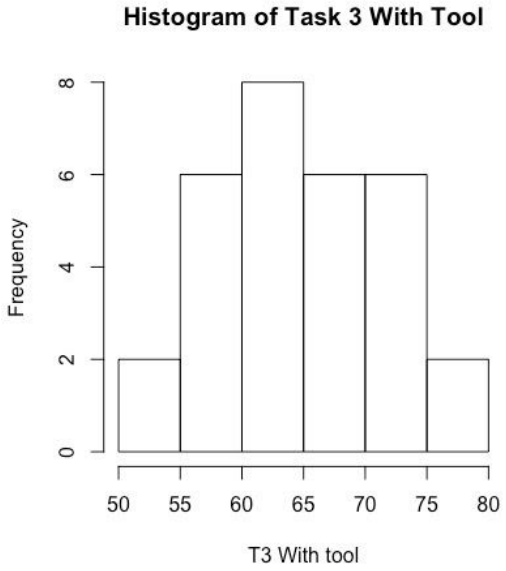


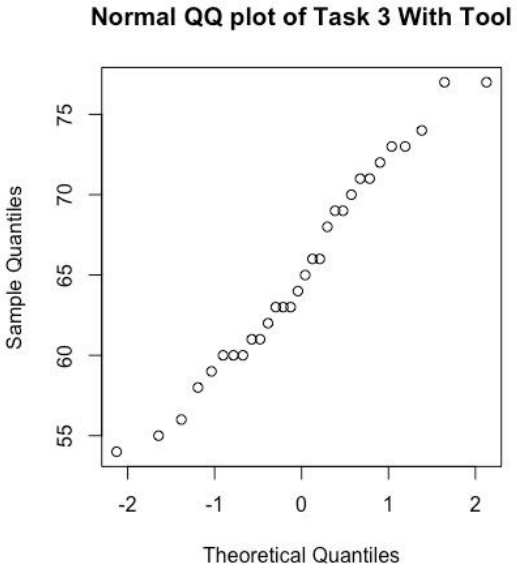


Shapiro-Wilk normality test

data: Time$T3.without.tool
W = 0.96963, p-value = 0.529

Normality in Task 3: With the tool





Shapiro-Wilk normality test

data: Time$T3.with.tool
W = 0.96873, p-value = 0.505

Test Result: Paired t-test(Task 3)

data: Time$T3.without.tool and Time$T3.with.tool
t = 10.562, df = 29, p-value = 1.882e-11
alternative hypothesis: true difference in means is not equal to 0
95 percent confidence interval: 7.015314 10.384686
sample estimates: mean of the differences 8.7

APPENDIX B :R STUDIO Code

```
#Task 1

mean(Time$T1.without.tool)
par(mfrow=c(1,2), pty = "s")
hist(Time$T1.without.tool,main="",xlab="")
title(main = "Histogram of Task 1 Without Tool", xlab = "T1 Without tool")
qqnorm(Time$T1.without.tool,main = "")
title(main = "Normal QQ plot of Task 1 Without Tool")
shapiro.test(Time$T1.without.tool)

mean(Time$T1.with.tool)
par(mfrow=c(1,2), pty = "s")
hist(Time$T1.with.tool,main="",xlab="")
title(main = "Histogram of Task 1 With Tool", xlab = "T1 With tool")
qqnorm(Time$T1.with.tool,main = "")
title(main = "Normal QQ plot of Task 1 With Tool")
shapiro.test(Time$T1.with.tool)

t.test(Time$T1.without.tool,Time$T1.with.tool,paired=TRUE)


#Task 2
mean(Time$T2.without.tool)
par(mfrow=c(1,2), pty = "s")
hist(Time$T2.without.tool,main="",xlab="")
title(main = "Histogram of Task 2 Without Tool", xlab = "T2 Without tool")
qqnorm(Time$T1.without.tool,main = "")
title(main = "Normal QQ plot of Task 2 Without Tool")
shapiro.test(Time$T2.without.tool)

mean(Time$T2.with.tool)
par(mfrow=c(1,2), pty = "s")
hist(Time$T2.with.tool,main="",xlab="")
title(main = "Histogram of Task 2 With Tool", xlab = "T2 With tool")
qqnorm(Time$T2.with.tool,main = "")
title(main = "Normal QQ plot of Task 2 With Tool")
shapiro.test(Time$T2.with.tool)

t.test(Time$T2.without.tool,Time$T2.with.tool,paired=TRUE)
```

```
#Task 3
mean(Time$T3.without.tool)
par(mfrow=c(1,2), pty = "s")
hist(Time$T2.without.tool,main="",xlab="")
title(main = "Histogram of Task 3 Without Tool", xlab = "T3 Without tool")
qqnorm(Time$T3.without.tool,main = "")
title(main = "Normal QQ plot of Task 3 Without Tool")
shapiro.test(Time$T3.without.tool)

mean(Time$T3.with.tool)
par(mfrow=c(1,2), pty = "s")
hist(Time$T3.with.tool,main="",xlab="")
title(main = "Histogram of Task 3 With Tool", xlab = "T3 With tool")
qqnorm(Time$T3.with.tool,main = "")
title(main = "Normal QQ plot of Task 3 With Tool")
shapiro.test(Time$T3.with.tool)

t.test(Time$T3.without.tool,Time$T3.with.tool,paired=TRUE)
```