FIT3161
Project Proposal with Literature Review

Designing software effort estimation model using machine learning techniques

Jovan Ong Shung Jiet
29274311
jong0016@student.monash.edu

Benjamin Marc Wijayaratne
30955246
bwij0006@student.monash.edu

Wong Zi Qi
30897548
zwon0015@student.monash.edu

Team: MCS17
Supervisor: Dr Golnoush Abaei
Word count: 7276

Table of contents

1. Introduction

Being able to estimate the effort required to develop a new software project has been a long time problem in the early stages of software project planning. In the early stages, software managers tend to not be able to plan and monitor software projects well enough and normally leads to overrunning costs, required effort, staffing and more. Large IT projects typically run 45 percent over budget for cost and 7 percent beyond the project deadline, as well as delivering 56 percent less value than anticipated (Bloch, 2020). Therefore, software effort estimation played a major role towards project planning to help complete projects under budget and by deadline. Furthermore, accurate estimation is a critical point in reducing the risks of completing projects which in turn also decreases the chances of projects failing. Since then, software effort estimation models have been evaluated for decades (Carvalho, 2021) to help project planning and effort estimation in the software industry. One of the widely used traditional techniques for software estimation is the Expert Judgment, which requires a lot more effort such as documenting activities, making the estimation process far more time-consuming and complex, and is also subject to potential human errors (Carvalho, 2021). Another problem that comes with traditional methods of software effort estimation is also the accuracy, because different project development lifecycle models necessitate varying degrees of effort at each stage of the lifecycle. Hence, Machine Learning techniques have been proposed to help tackle the software effort estimation issue, as it is an efficient alternative that is not subject to human error, as well as having a significantly efficient learning capacity and its ability in modifying its behaviour autonomously.

The goal of our project is to develop a working machine learning software effort estimation model that is more effective than current state-of-the-art software estimation models. Furthermore, we plan to be able to build our model to accurately predict effort required for both Agile and Traditional projects. Prior to training the machine learning algorithm for effort estimation, we intend to better extract data from repositories with data understanding, and then pre-processing the data by quantifying and processing missing data. Furthermore, the aim is to scale and partition the dataset accordingly for effective use of the dataset to meet time and space performance requirements. Finally, we still select the best performing model from testing various state-of-the-art models as well as our personal model using various measures for an accurate model for effort prediction.

For our project, the development of an intuitive GUI is just as important. We aim to firstly develop a web-based effort estimation tool with a user interface that focuses on ease of use for users, and to also have a customizable template to reflect on project management approaches. Afterwards, we intend to research and test further on more machine learning and deep learning techniques to help improve the accuracy of the effort estimation model. We have also associated the potential risks of the project and discussed methods to monitor and

mitigate the risks, as well as assigning major roles to each member in the team based on areas of expertise and specialisation.

In the following semester, datasets widely used for software effort estimation will be collected. We intend to continue developing the project code and eventually train and test our machine learning based software estimation model and compare the results with other conventional state-of-the-art models.

2. Literature Review

2.1 Introduction

In the past decades, software estimation in projects has been alluring numerous research to improve the accuracy of effort prediction. Extensive studies have been put into applying various machine learning (ML) approaches to effort prediction utilizing historical data at the onset of the project life cycle. As software projects become more complex, traditional parametric models and statistical methods often fail to depict the correlation between the project features and the software effort (Suresh Kumar et al., 2021). Thus, as software effort estimation practices progress immensely, techniques should be available to compute the effort required to build abruptly changing software. ML is preferable in this case because it has the ability to access historical data from which it can learn from it (Das et al., 2015). Also, it can adapt to a wide range of variations that a software project is subject to (Das et al., 2015). Palaniswamy and Venkatesan (2021) agree with this position, by suggesting that ML techniques are effective in addressing bias and subjectivity concerns if suitable data pre-processing and feature extraction approaches are utilized.

Periodically, defining which effort estimators can achieve the best results would be challenging in the ML aspect. Any effort estimator may move up or down in the ranking since comparisons of estimators are conducted based on modified conditions. To resolve this limitation, the resulting approach from merging the estimates of various estimators, alias Ensemble Learning, is introduced. With ensemble learning, which integrates multiple learning algorithms, the software effort estimation process can be improved to yield better accuracy in the effort predicted (Kocaguneli et al., 2012).

In this literature review, we hope to foster deeper understanding in the research with the purpose of developing a more advanced approach that provides estimation in software effort using ensemble learning.

2.2 Content

2.2.1 A Pragmatic Ensemble Learning Approach

Suresh Kumar et al. (2021) compared various ML algorithms, such as Stochastic Gradient Descent (SGD), K-nearest neighbor (KNN), Decision Tree (DT), Bagging regressor (BR), Random Forest regressor (RFR), AdaBoost regressor (ABR) with their proposed Gradient Boosting regressor (GBR) by applying COCOMO'81 and CHINA datasets which are publicly available in the PROMISE repository. COCOMO'81 dataset contains 63 software projects, while the CHINA dataset comprises 499 projects. The performance was evaluated using mean absolute error (MAE), mean squared error (MSE), root mean squared error (RMSE), and coefficient of determination (R^2). The statistical results concluded that the proposed GBR outperformed the other models in terms of lower MAE, MSE, and RMSE, and higher R^2 overall. Future work comprises adopting other ensemble learning models in software effort estimation and attaining exclusive consideration towards large-sized projects.

2.2.2 Extreme Learning Machine

De Carvalho et. al. (2021) presented a comparative study of their model built using the Extreme Learning Machine (ELM) technique, with the models in the paper's literature, including Support Vector Machine (SVM), Linear Regression (LR), Multi-Layer Perceptron (MLP), and KNN. The dataset, Desharnais dataset, and Carvalho's work dataset were used to estimate and compare the effort of a software project. They found that the ELM model applied to the Desharnais dataset had outperformed KNN, LR, SVM, and MLP models in terms of lesser Mean Magnitude of Relative Error (MMRE), MAE, MSE, and RMSE. Observations showed that the applied ELM model obtained the best results due to its simplicity, rapid learning, and better performance in generalization. Also, the applied ELM model exhibited lesser distortion than those literature models. A statistical test was used to validate the results through a normality test and hypothesis test. A further investigation of performance was carried out through the application of the ELM model to Carvalho's Work dataset. The result revealed that the generalization performance of the applied ELM model with 2 and 5 hidden layers was more stable and reliable. The limitations to this study were the choice of ELM algorithm parameters, data availability, and the MMRE metric, which does not provide good accuracy in the effort prediction, as stated by Shepperd and MacDonell (2012). For future work, the authors suggested employing Particle Swarm Optimization (PSO) to optimize the model parameters. A study that utilizes PSO proposed by Palaniswamy and Venkatesan (2021) will be discussed in the following.

2.2.3 Hyperparameters Tuning of Ensemble Model

Palaniswamy and Venkatesan (2021) proposed a methodology that employed a stacking ensemble. Stacking ensemble uses a meta learner to integrate diverse fundamental machine learning algorithms. The hyperparameter tuning of base learners and a meta learner was then conducted using PSO and Genetic Algorithms (GA). These algorithms search through the immense hyperparameter space to determine the optimal hyperparameter values for the ensemble model. After preprocessing, the ISBSG dataset with 13 independent attributes, a target variable, and a summary work effort was used in this research. The authors had restricted the number of base learners to four since the accuracy is not enhanced even if we increase the base learners from 4 to 5 and more. The selected base learners were LR, MLP, RFR, and ABR. The authors also employed SVR as the meta learner with radial basis function (RBF) kernel. The evaluation metrics used were MAE, MMRE, and PRED(x). If MMRE is less than 0.25 and PRED (25) is greater than 0.75, the model would then be considered satisfactory in terms of accuracy. Besides, two sets of experiments were carried out to forecast the accuracy of software effort estimation, namely without hyperparameters tuning, and with hyperparameters tuning using PSO and GA individually. They found that the stacking ensemble with hyperparameters tuning outperformed the ensemble without tuning in terms of lower MAE, lower MMRE, and higher PRED(x), and the PSO method yielded slightly better performance than GA in terms of accuracy. For future works, the authors proposed that more data from IT industries of relating fields can be gathered and used for software effort estimation to enhance accuracy. Other recent optimization techniques, which produce the near best hyperparameter configurations, can be utilized to explore the immense hyperparameter space and yield better outcomes.

2.2.4 Agile Story Point Estimation Approach

Prior studies in software effort estimation, but in an Agile context, have mainly focused on estimating the effort of stories that constitute a project. Story points were utilized to evaluate these stories. Both studies proposed by Choetkiertikul et al. (2019) and Fu and Tantithamthavorn (2022) used Deep Learning architectures to estimate the story points of a given user story on JIRA datasets. During the word-embedding process, Choetkiertikul et al. (2019) utilized word-level tokenization, while Fu and Tantithamthavorn (2022) employed byte pair-encoding subword tokenization. Story points are related to the corresponding effort value in the following literature.

2.2.4 Effort Estimation for Agile Software using Story Points

Satapathy and Rath (2017) applied the story point approach along with ML algorithms to estimate the effort required for building software projects using Agile methodology. The datasets used contains twenty-one software projects developed by six software houses. An attempt had been made to improve the results of the story point approach using DT, SGB, and

RF algorithms over the story point dataset. Validation and comparison of the outcomes were conducted with the existing models obtained from Zia et al. (2012). They found that the SGB technique outperformed other ML techniques based on three metrics, MMRE, MdMRE, and PRED (x) for the considered dataset. Future research in this study would be applying other ML techniques such as ELM and BN to the story-point-approach-related dataset. The employment of BN on the stories-related dataset was proposed by Malgonde and Chari (2019) in the following.

2.2.5 Ensemble-based Model for Predicting Agile Software

In 2019, Malgonde and Chari proposed a predictive model and an ensemble-based model to estimate the effort required to develop stories for agile software development projects using a dataset of 503 stories obtained from the information technology (IT) department of a university. The predictive model employed seven ML algorithms, such as Bayesian network (BN), Ridge Regression (RR), Artificial Neural Networks (ANN), SVM, DT, KNN, and Ordinary Least Squares Regression (OLS). The authors suggested that the training of predictive models should follow temporal precedence since agile projects are completed in a sequence of sprints. Therefore, the predictive models were trained by utilizing the blocked cross-validation technique instead of using basic cross-validation. This technique was then used along with MAE as the score function for determining optimal hyperparameter values throughout the hyperparameter tuning process. The three measures used in performance evaluation were MAE, Mean Balanced Error (MBE), and RMSE. The result suggested that neither of the predictive algorithms uniformly outperformed others across sprint categories on any of the metrics. In addition, a statistical test, Friedman's test was conducted to compare the performance of candidate algorithms by identifying if there were considerable differences between the errors. BN and OLS models were found to have statistically higher MAE errors than other algorithms. The authors also developed an ensemble-based model which retained SVM, ANN, KNN, DT, and RR as its components based on the effect size analysis (Friedman's test). This ensemble-based model could reliably deliver good estimations on an aggregate basis, as evidenced by the performance evaluation. The results revealed the ensemble-based model outperformed other approaches in terms of MAE and MBE. The limitations to this study entail replications of bias or distortion by ML algorithms since archival datasets were used, impacted generalizability of findings since experiments were limited to the dataset from a particular organization, and error-prone process of estimating effort since the models used were calibrated utilizing project data relating to earlier technologies. Lastly, the authors suggested including human experts in ensembles and providing efficient optimization methodologies at the project portfolio level in future research. Efficient optimization of hyperparameter tuning is examined in the earlier literature proposed by Palaniswamy and Venkatesan (2021).

2.3 Synthesis Matrix

Author	Dataset	Methodology	Evaluation Metric
Suresh Kumar et al. (2021)	1) COCOMO'81 2) CHINA	SGD, KNN, DT, BR, RFR, ABR, GBR	1) MAE 2) MSE 3) RMSE 4) R^2
De Carvalho et. al. (2021)	1) Desharnais 2) Carvalho's Work	ELM, SVM, LR, MLP, KNN	1) MMRE 2) MAE 3) MSE 4) RMSE
Palaniswamy and Venkatesan (2021)	ISBSG	- Stacking ensemble - Base learners: LR, MLP, RFR, and ABR - Meta learner: SVR - Hyperparameter tuning: PSO, GA	1) MAE 2) MMRE 3) PRED(x).
Choetkiertikul et al. (2019)	JIRA	Deep learning, word-level tokenization	-
Fu and Tantithamthavorn (2022)	JIRA	Deep learning, byte pair-encoding subword tokenization	-
Satapathy and Rath (2017)	21 software projects developed by 6 software houses	DT, SGB, RF	1) MMRE 2) MdMRE 3) PRED (x)
Malgonde and Chari (2019)	503 stories obtained from the IT department of a university	Predictive moel: BN, RR, ANN, SVM, DT, KNN, OLS Ensemble model: SVM, ANN, KNN, DT, RR	1) MAE 2) MBE 3) RMSE

Table 2.1 Synthesis Matrix

2.4 Conclusion

Based on our findings, it is observed that there are various successful and diverse approaches to our topic prevail. We have identified multiple ensemble algorithms which can be employed to produce software effort estimators. The selection of integrated ML techniques in the ensemble may include the ELM model proposed by De Carvalho et. al. (2021) since it yielded better performance. Furthermore, we may perform automated hyperparameters' tuning on the ensemble model using the PSO approach to find the near best hyperparameter configurations for optimization purposes. For Agile software development, we may make use of the existing story-point estimator tool to process the user stories. The collected story points can then be taken as input arguments for ML models to compute normalized effort.

3. Project Management Plan

3.1 Project Overview

This project involves developing and implementing a software effort estimation system using machine learning techniques to produce accurate effort estimation that prevents slippage to timelines and over budgeting issues, thus increasing the chances of success in the projects.

3.2 Project Scope

3.2.1 Scope

In-scope

Additional scope for Agile projects:

- A story-point estimator system that integrates word embeddings, context network, and context transformation network to process natural language.
- Collection of story-point, project velocity and actual effort values.
- Data analysis to verify if the collected dataset is normally distributed.
- Data transformation if the collected dataset does not follow normal distribution.

Scope applicable for both Agile and traditional projects:

- Data understanding to better extract data from repositories.
- Quantifying and processing of missing data.
- Scaling of dataset.
- Partitioning of dataset.
- Performance evaluations using different measures.
- Model selection.
- Effort prediction.

Development of intuitive GUI:

- A web-based effort estimation tool.
- A customizable template to reflect on the project types.
- A solution panel that provides summary of our estimates.

Testing:

- Unit testing to ensure each segment operates properly.
- Integration testing to verify if the selected models for Agile, traditional projects, and GUI operate together as expected from user perspectives.
- System testing to evaluate if the system successfully estimates effort and displays it at the user end.

Out of scope

- Effort estimation in mobile application development.
- Incorporation of our effort estimation model with other systems that provide roadmap, backlog, Gantt chart, etc. to create an inclusive project management tool.
- Training and assistance in estimation tool utilization for company involvement.

Constraints

- The system should be applicable for all types of project development approaches, including Agile.
- Availability of dataset.
- Availability of computational resources.
- Budget.
- Time limitations.

Assumptions

- Estimation accuracy for Agile projects will increase if task descriptions have more text.
- User input involves selection of project development approach. For Agile approach, the input will comprise task summary, and task description written in natural language. For traditional approach, the input will comprise project name, project sizing, scale drivers, and cost drivers.
- The users are expected to have decent knowledge of using computers.
- The full project team will be available to create the WBS.

Project deliverables

Project management-related deliverables:

- Weighted-scoring model
- Business case
- Project scope statement
- Requirements traceability matrix
- Work breakdown structure
- Gantt chart
- Meeting minutes
- Final project presentation
- Final project proposal

Product-related deliverables:

- A web-based software effort estimation tool
- Design representations of the system
- Source code of algorithm implementation

3.2.2 Product requirements

Characteristics requirements

- The tool is designed to accurately estimate the software effort required to develop a project, with intuitive GUI provided. The tool's functional requirements should contain all technical details and testing for delivering its uninterrupted functionality.
- The tool should work properly and return desired output, which is software effort, in this case, within an amount of time in the specified environment.
- The tool should be publicly available online.
- The website should be visually pleasant and error tolerant.
- The user's personal data, for instance, some project details which are recognized as confidential information of a company, should be protected.

Functional and non-functional requirements

ID	***Requirements (Functional or Non Functional)***	***Assumption(s) and/or Customer Need(s)***	***Category***	***Source***	**Status**
001	Successful modelling	ML techniques used are examined and verified in the industry	Functional	Project proposal	In Progress
002	Correctly conducts experimental evaluation using different measures	Evaluation measures used are examined and verified in the industry	Functional	Project proposal	In Progress
003	Produces appropriate visualization for comparison purposes	Visualization is appropriate for different types of results	Functional	Project proposal	In Progress
004	Selects optimal and suitable model as final model	Final model is appropriately chosen from the built models	Functional	Project proposal	In Progress
005	Final model correctly computes software effort estimation	Effort estimation computed is validated by industry benchmark	Functional	Project proposal	In Progress
006	Produces a customizable template to reflect on the type of projects, development method, resources need, etc.	The template takes user input	Functional	Project proposal	In Progress
007	Displays a solution panel that presents a summary of our estimates	The unit measurements for effort and cost depend on the dataset and algorithm implementation	Functional	Project proposal	In Progress

008	The system is robust enough to handle invalid inputs	Feedback to user is displayed instead of triggering crash	Non-Functional	Project proposal	In Progress
009	The system is user-friendly	Proper rounding for result of estimation	Non-Functional	Project proposal	In Progress

Table 3.1 Requirements Traceability Matrix

3.2.3 Product user acceptance criteria

1) Successful implementation of software effort estimation system using optimal and suitable model.
2) Effort prediction validated by industry benchmark and projects history.
3) Successful development of intuitive GUI.
4) High-level configurability of the system.

(refer to case study 3)

3.3 Project Organisation

3.3.1 Process Model

Agile methodology, which embraces adaptive planning, would be more suitable for our product initiative due to its flexibility and learning-embraced nature. The principal objective of developing the product for our project team is to adapt our knowledge to real-world applications and provide insight into a new aspect of study through self-research. Employing Agile encourages and embraces learning as part of the product development process - the product is defined when the team iterates, allowing the team to learn, regulate and improve on delivering a product with better quality more quickly from the gathered feedback. There are no well-defined priorities and requirements at the onset of the project. They can be easily adjusted to any changes and to satisfy the needs of stakeholders, exhibiting the team's ability to take advantage of the opportunities and eventually deliver a better solution during the development process. Each improved work on partial systems can then be successively built and delivered throughout the project to produce a final completed system. Moreover, the team is not directed by a professional project manager, which corresponds to the Agile methodology when it comes to project management roles. We are given shared responsibility to deliver the project and are expected to be self-organized. On contrary, the Waterfall methodology, which embraces predictive planning, would bring drawbacks to our team. Its structured and straightforward nature makes the development costly as the requirements and the needs of stakeholders can vary throughout the research process. We may find one implementation from a research paper more effective than the others. Adjusting the previous framework to match the latest implementation after defining every requirement in the beginning is difficult and often entails expensive rework. Furthermore, the Waterfall methodology follows neither an iterative nor incremental approach. Before any development can proceed, the requirements for Waterfall projects must be entirely documented and permitted, leading to excessive effort and focus spent on building documentation instead of the product.

3.3.2 Project Responsibilities

Team member	High-Level Responsibilities
Technical Lead	Jovan Ong Shung Jiet
Scrum Leader	Benjamin Marc Wijayaratne
Quality Assurance	Wong Zi Qi

Table 3.2 High-level responsibilities

High-level responsibilities can be seen described in the table above; Table 3.1. Table 3.2 will describe each major project function and the individuals that are responsible using a responsibility assignment matrix based on the RACI (Responsible, Accountable, Consulted,

Informed) model. The assignment matrix may not be complete as some responsibilities for the development stage have not been defined.

Project function	Benjamin	Jovan	Zi Qi	Supervisor
Literature review	A	A	R	I
Dataset Collection	C	A	R	C
GUI Design	C	R	A	I
Software and Hardware	R	A	C	C
Libraries and Frameworks	C	R	A	C
Project Management Plan	I	I	A	C
Scope statement	I	I	A	C
Gantt Chart	A	C	C	I
Meeting Minutes	A	C	C	I
Risk Register	R	A	A	C
Work Breakdown Structure	R	A	A	I
Requirements Traceability Matrix	I	I	A	C
Methodology	R	A	A	C
Dataset preprocessing	I	R	A	C
Dataset partitioning	I	R	A	C
Ensemble ML	I	R	A	C
Evaluation and Analysis	I	R	A	C
Code Documentation	I	R	A	I
Unit Testing	I	A	R	I
User Manual	I	A	R	I
Design Documents	A	R	R	C

Table 3.3 Responsibility Assignment Matrix

3.4 Management Process

3.4.1 Risk Management

Overall, our project did not have to worry about stakeholder management (as we really only had to worry about our client) or handling material resources (such as money). Because of that, we had very few risks that had any actual probability of happening. They also had very little impact on us if they did happen to occur.

Our risk management process overall first began with identifying possible risks. This would be done as a team as our biggest resource at stake was time, this was especially important as we were doing this as a project while in university, thus every team member had their own individual classes to attend and their own varying share of time and effort that they could contribute.

From there we would move on to creating and going over our risk register. This would allow us to put potential risks on a proper list and allow us to rank them accordingly. Using the risk register we would then go over each item on the register and discuss how to either mitigate the risk entirely or develop protocols in order to lessen the impact of the risk at hand if avoiding was completely impossible.

3.4.2 Stakeholder Analysis and Communication Plan

Our primary reporting mechanism used is our Microsoft Team. Through this we can report any issues to our client and bring up any topics of concern that may need to be addressed within the day itself and cannot wait for a scheduled meeting.

Other than that, our main method of communication or reporting to our client would be through our weekly meetings. Usually these meetings last anywhere from 15 to 30 minutes and occur every Friday through a scheduled Zoom meeting. During these meetings, we typically begin with a report on the progress we have made or the research we have conducted followed by displaying our findings or research to our client. Following this, our client will give us any feedback regarding our work and we decided on tasks for the coming week to be done before our next meeting.

3.4.3 Monitoring and Controlling Mechanisms

Communication Plan and Task Allocation

Our main form of communication is through Discord, a messaging application that has recently gained popularity. Upon us being given a new task (usually either in the form of from our client during weekly meetings or through the form of an assignment), we would then break off into our group and split up the larger overall task into smaller, more manageable tasks that each person would then take on themselves. The allocation is usually decided according to the specialisation or previous experience of each group member. This allows for every group member to contribute to their split of the group work, allows for the group members to work on the tasks in which they are specialised in and also prevents oversaturation from occurring on a certain task.

Monitoring of Project Progress Against Planned Milestones etc

Monitoring of project progress internally within the team occurred every other day or everyday depending on the urgency of the project at hand. This would typically involve the group leader going over and reviewing the progress of the other group members. This would ensure that no group member was going off course or progressing in a different direction.

It was paramount that every member of the group was on pace to produce results on time. This is due to the fact that although the tasks are usually split into smaller tasks for each member to undertake, usually all of the tasks would still contribute to the same deliverable, hence if one member is late, the other members also become late.

Review and Audit Mechanisms

Currently, in the research stage of the project, we are currently using Google Drive to audit and review each other's work. This allows us to both view the contributions of each member, but also to contribute to each other’s work should any of us require it.

In the implementation stage of the project, we plan to make use of GitHub to share and collaborate on any code work. This allows for code to either be ‘pushed’ to the repository to make changes to the code, or ‘pulled’ from the repository to update the code existing on our local systems.

We will most likely be making use of stage 1’s methods in stage 2, should we need to increase our knowledge concerning the topics at hand or if any interesting or concerning research needs to be taken into account.

3.5 Schedule and Resource Requirements

3.5.1 Schedule

1.0 Initiation - 7/4/2022
- 1.1 Develop Business Case (Finish By 14/4/2022)
- 1.2 Assignment Project Leader (Finish By 8/4/2022)

2.0 Planning and Resource Delegation
- 2.1 Primary Planning
  - 2.1.1 Develop WBS (Finish By 19/4/2022)
  - 2.1.2 Create Schedule (Finish By 28/4/2022)
  - 2.1.3 Allocate Resources (Finish By 9/5/2022)
- 2.2 Research
  - 2.2.1 Search for Information (Finish By 20/5/2022)
  - 2.2.2 Condense and Analyse Data (Finish By 20/5/2022)
  - 2.2.3 Analyse Information (Finish By 20/5/2022)

3.0 Implementation
- 3.1 Acquire Data (Finish By 8/8/2022)
- 3.2 Preprocess Data (Finish By 17/8/2022)
- 3.3 Partition Data (Finish By 25/8/2022)
- 3.4 Evaluate Models (6/9/2022)
- 3.5 Select Model (15/9/2022)
- 3.6 Evaluate Selected Model (25/9/2022)

4.0 Closeout
- 4.1 Document Closeout (5/10/2022)

Note: This schedule operates on the assumption that the semester will begin on the 8th of August 2022, hence why part 3 begins in August.

3.5.2 Resource Requirements

No	Type	Details	Quantity	Source	Assumptions
1	Person	Project Leader	1	Staff	
2	Person	Team Member	2	Staff	Adequately trained in required softwares
3	Hardware	PC	3	Staff	Hardware specs good enough to run required software
4	Software	Discord	N/A	Freely Available	
5	Software	Microsoft Teams	3	University	
6	Software	Google Drive	3	University	Large enough to hold all required files (10GB++)
7	Software	GitHub	N/A	Freely Available	
8	Software	Python	N/A	Freely Available	Includes all required libraries (Ensemble ML included)

Table 3.4 Resource requirements

4. External Design

4.1 User interface

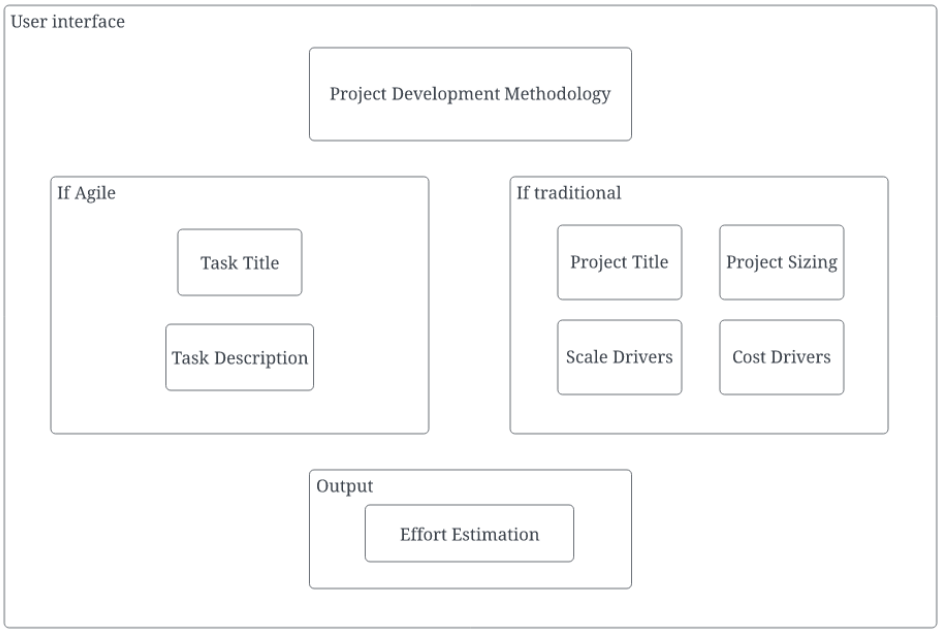


Figure 4.1. Overview of the user interface

The application's user interface will allow the user to:

- Enter a software development project for effort estimation
- Choose project management approach (agile or traditional)
- Enter variables of projects that can affect effort
- Visualise and evaluate effort costs for each project

In addition to these fundamental requirements, we are aiming to keep the user interface simple yet meaningful and effective, providing the users a pleasant experience while using the web application. Furthermore, we intend to ensure that all elements in the user interface are positioned and sized proportionally in regards to the size of the display presenting them to further increase user satisfaction.

4.2 External contents used

Libraries/Frameworks

Front-end

Framework	Justification
Streamlit	● Open-source app framework for constructing machine learning data scripts into web apps
Flask	● Micro web framework that provides tools for easy development of web applications

Back-end

Library	Justification
Matplotlib	● Library for plotting and visualising data ● Extension of NumPy
NumPy	● Python library for multidimensional arrays, matrices, large collection of array functions
Pandas	● Open-source library for data analysis and manipulation
Scikit-Learn	● Free and open-source general purpose machine learning library
Tensorflow	● Provides tools for developing 2D and 3D convolutional neural networks (CNN)

Datasets

1) Atlassian JIRA repositories
The JIRA repository is available online and contains both Agile and traditional projects data. It is the dataset used to train the model for effort estimation.

2) COCOMO Dataset
The COCOMO dataset is available online and is a widely used dataset for software effort estimation. The dataset is mainly used for traditional approaches only.

5. Methodology

5.1 Research

This stage consists of the team conducting the research that will be the building blocks of our project moving forward.

5.1.1 Searching for Research

To begin with we began our research by simply looking for any and all papers that consisted of the keywords ‘machine learning’ and ‘software effort estimation’. This allowed us to filter papers that related to our research topic while also maintaining a wide enough net that would allow for us to be able to pick up any papers that were not necessarily directly related to our research but would be helpful in building our knowledge in the field.

5.1.2 Research Analysis

From there we would then split the research into parts for each member to read. Each member would then read their split of the research papers and analyse each paper individually.

Each member’s analysis would be compiled into a list which would include multiple different items:

1. Data - Information about the data set used in the research as well as (if publicly available) where to obtain said data set.
2. Publishing Year - Year in which the paper was published (Used mainly to check for relevancy of the paper).
3. Methodology - Information about the methodology used in the research. Summary of steps.
4. Evaluation Metrics - A list of the types of measurements used in the research to evaluate the results.
5. Futureworks - How the research could be improved or changed to cover more

This allows us to have a comprehensive list of the most important information concerning each and every research paper that we researched without having to have each member read every paper, and so giving us an extra boost in efficiency.

From these processes would arise the foundation needed for us to progress to the next stage of our project.

5.2 Implementation

Our stage 2 is based upon the following chart:

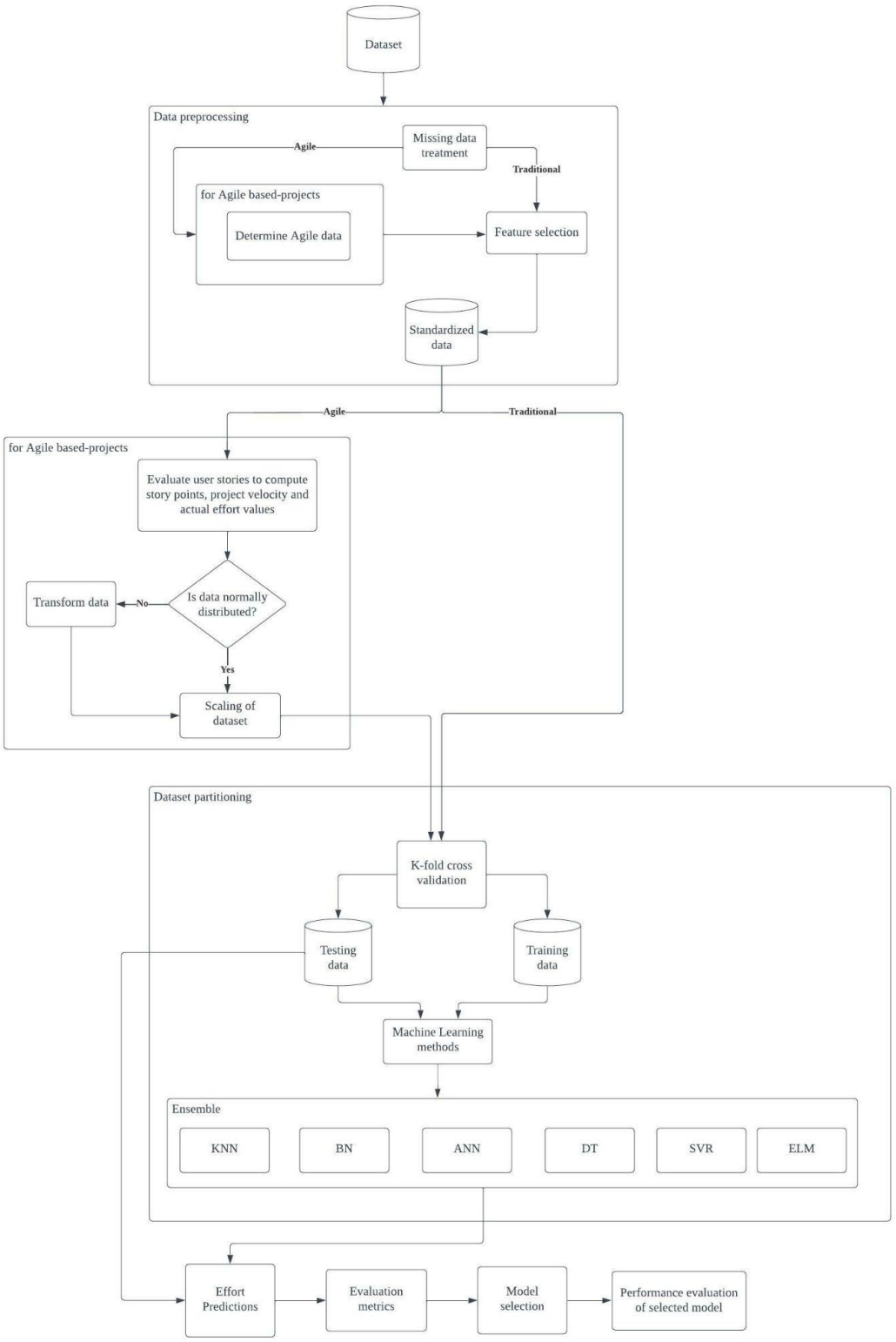


Figure 5.1 Backend Flow chart

5.2.1 Data Preprocessing

In this study, the ISBSG dataset or Atalassian JIRA repositories that contains both Agile and traditional projects can be employed. The data extracted is in heterogeneous form. The extracted heterogeneous data is not usable for training purposes due to non-uniform data from real-world-based projects, hence it is required to be filtered especially for Agile projects. The datasets COCOMO'81 can be used for traditional projects.

Process missing data

Missing values may affect software effort estimation accuracy. The dataset's missing values are dealt with Missing Data Treatment via dropping columns and rows with more null values

Determine Agile data (optional)

Agile data is extracted from the data pool by analysing the attributes and unique characteristics of Agile based-projects, forming the Agile sample set.

Feature selection

Removal of irrelevant independent features helps in improving the performance of machine learning techniques. Through the computation of correlation coefficient matrix, the irrelevant features are eliminated while the principal features are retained. The dataset is now standardised.

5.2.2 Further data processing for Agile based-projects (optional)

Further data processing is required for Agile based-projects since the standardised data from the previous step does not take the principle parameters an Agile project requires for effort prediction into account. There are three main parameters, including story points, project velocity and actual effort values.

Evaluation of user stories

User stories refer to the smallest unit of work in the Agile framework. They are evaluated to determine the principal parameters. These collected story points and the final velocity value of the agile projects are taken as input arguments for diverse machine learning models to calculate normalised efforts.

Analysis of processed dataset

The processed dataset based on the story point approach is analysed statistically to remove outliers that may potentially affect the estimation accuracy by identifying if the dataset is normally distributed based on the values of skewness and kurtosis. A non-normally

distributed dataset is required to go through the transformation process to ensure that it follows the normal distribution.

Transformation of data

The dataset is transformed logarithmically into a normally distributed dataset. Visualisations can be provided to verify if the transformation aids by analysing the before-and-after distribution.

Scaling of dataset

The dataset is normalised to achieve uniform and comprehensible results.

5.2.3 Data partitioning

The processed dataset is segregated into training data set and testing data set using K-fold cross-validation to build models using ensemble ML. Ensemble ML, which is studied from the literature review is employed. The base classifiers incorporate various ML techniques, including KNN, BN, ANN, SVR, DT, and ELM. The selection of ML techniques may vary depending on the future implementation. Testing on the built models is carried out using the testing data set.

5.2.4 Performance evaluation and Model selection

In this section, different evaluation metrics are employed to select the best model that produces the most accurate estimation based on the simulated dataset and factors. There will be two best models selected for traditional and agile based-projects respectively.

5.2.5 Performance evaluation on selected model

The selected model is evaluated once again with a different dataset or different seed to ensure that the evaluation results of the selected model are not misleading. The reevaluation methods may vary depending on the future implementation.

(refer to Project Initial Concept & Design)

5.3 Software and Hardware Specifications

5.3.1 Specification Sheet

No.	Category	Suggested	Potential Substitute
1	Software Library	Ensemble ML	AdaBoost
2	Programming Language Environment	Python v3.7 running on either VSCode or PyCharm	Any newer version of Python after v3.7
3	Project Management	Microsoft Teams	
4	Project Management	Google Drive	
5	Project Management	GitHub	Google Drive
6	Project Management	Project Libre	Microsoft Project
7	Hardware Requirements	- Intel Core i5 6th Gen - 16 GB RAM - NVIDIA GeForce GTX 960	- Higher Gen accepted - Anything higher than 8GB accepted - Higher Gen Accepted
8	Data Storage	2GB	Higher accepted

Table 5.2 Software and Hardware specifications

5.3.2 Justification

There was no specialised software included as all software to be used are widely available to the public and free to be used.

Software libraries had been boiled down to ‘Ensemble ML’ since it was our primary ML technique.

For now, the system we have decided to use in order to store our data is Google Drive as it is freely available to people to be used and access is granted and managed easily through the use of Google’s share system.

We have decided to run with Python 3.7 or above as our chosen programming language. This is mainly due to our experience with Python collectively as well as version 3.7 being the version chosen to be the standard Python version used in the university.

Three project management tools were selected:

ProjectLibre as a reliable Project Management Tool.

Microsoft Teams to enable communication between supervisor and the team as well as enable the team to communicate with each other.

Google Drive to enable quick access to files and documents (both for editing and for viewing).

GitHub to enable easy sharing of code between team members.

No networking requirements were listed as this project does not require a network connection aside from potentially the initial download of the required software.

Hardware requirements provided are the basic requirements typically expected of a computer when working with machine learning software. CPU could be switched out for another CPU of the same power, but GPU must be NVIDIA if it is to assist in the machine learning process as AMD GPUs do not allow for assisting in the machine learning process.

2GB of data storage was selected in order to allow for space for the required libraries. This may change in the future as the software we make use of and as the libraries we make use of change.

6. Test Planning

As a professional software developer team, we should aim to reliably deliver software that meets its quality goals. This can be achieved through effective test planning, which outlines the test scope, test strategy that can be employed to verify if our system is developed according to its requirements and specifications, test objectives, and the resource requirements essential to carry out the testing process. The test scope refers to the project scope (*section 3.2*). The detailed in-scope functionalities will be tested, while out-of-scope functionalities will be excluded from testing.

6.1 Test Strategy

Each module that aims to meet certain software quality attributes based on the requirements listed under Project Scope (*section 3.2*) will be verified or validated through testing.

6.1.1 Unit Testing

Unit testing for both backend and frontend strives to assure software functional correctness. In the backend system that performs model training, the modules may comprise data processing, data analysis, data transformation, data scaling, data partitioning, performance evaluation, and model selection. The backend system may also include a module to validate incoming input, which can be deployed during input processing in the frontend. This module strives to assure robustness by dealing with exceptional inputs. For the frontend system, modules such as displaying a customizable template for inputs, input processing, and presenting a solution panel will be tested. Usability will be assured during the testing for the frontend system.

6.1.2 Integration Testing

Next, integration testing will be carried out after passing unit tests to verify if system components correctly interact with each other. For instance, it helps verify if the backend and frontend operate together as expected, which involves successfully taking in input, using the best model to compute software effort estimation based on the input, and displaying the results at the user end. Integration testing strives to assure all software quality attributes that are satisfied in unit testing. With the Agile approach employed for developing our project, we continuously build, test, and deploy iterative code. Through this iterative process, we can reduce the likelihood of developing new code based on the buggy or failed code from the past. Taking the above assertion into consideration, Continuous Integration (CI), a technological approach to performing integration testing, will be deployed to ensure that each submitted modification of code will pass all test cases, guidelines, and code compliance standards we have ascertained for our system. Test coverage will be measured to gauge the effectiveness of our tests. In addition, we will be implementing Benchmark testing to validate our system based on industry standards. We may conduct simulations of models under the performance evaluation section to evaluate time consumption and compare the performance

results of the selected model against metrics that have been agreed upon in the industry. This helps in assuring performance efficiency. Moreover, we will be implementing API testing to verify if the system properly receives and sends requests between frontend and backend using Streamlit API, connects to an existing story-point estimator system under the input processing module to specifically estimate effort for Agile-based projects, as well as connects to the database under data processing module. Mocking can be used to handle these external dependencies since it provides flexibility to promptly execute efficient testing that is well suited to the CI environment. It works by simulating the modules that interact with the component. Its flexibility comes from returning pre-configured values, which is faster than accessing the real external dependency. Also, our system will be fault-tolerance by using Mocking because we can simply make the simulation of a dependency crash, for example, a database, if we want to ensure that our system will keep functioning even if one of its dependencies crashes. These features help assure performance efficiency and reliability.

6.2 Test Objectives

Following is a traceability matrix that relates requirements with test objectives:

Requirement		Functional Correctness	Usability	Reliability	Robustness	Performance Efficiency
Unit Testing	Backend	✔			✔	
	Frontend	✔	✔			
Integration Testing	Functionalities from unit testing	✔	✔		✔	
	Benchmark	✔				✔
	API	✔		✔		✔
Overall		✔	✔	✔	✔	✔

Table 6.1 Traceability matrix for test objectives

6.3 Resources

We will deploy the following libraries and tools to aid in testing our system.

PyUnit

By using the Python unit testing framework, unit tests can be automated, set up and shut down code can be shared between tests, tests can be grouped into collections, and tests can be independent of the reporting framework. The classes that support these qualities for a set of tests are provided by the ‘unittest’ module. We will break down each module of our system into various test cases. Each module represents a test suite. The test suite is utilized to group tests that should be executed altogether. Also, mocking can be implemented using ‘unittest.mocking’.

GitLab

We will employ a GitLab as our server to automate the builds and test for CI. The server provides features that assist in implementing the testing, including triggering an automated build of our system when a new revision of code is registered, running tests and analyzing the test coverage on the new build, and providing easily accessible results of running the tests. Moreover, GitLab fully integrates our version control system and CI system, making it easy to set up for testing.

6.4 Sample Test Cases

Following is a sample of test cases for our system:

Requirement	Test Suite	Test Case	Verified Date
Backend	Data processing	- Checks if the retrieval of datasets from database is successful - Checks if the processed data contains no missing data - Checks on the quality of preprocessed data	Incomplete
	Data analysis	- Checks on the selected feature based on some criteria - Checks if non-normally distributed dataset goes to data transformation step, otherwise proceed to the data scaling step	Incomplete
	Data transformation	- Checks if the transformed data follows normal distribution	Incomplete
	Data scaling	- Checks if the data is properly scaled between the range of 0 to 1	Incomplete
Frontend	Input validation	- Checks if input is correctly validated - Checks if message is displayed properly when invalid input is entered	Incomplete
	Output verification	- Checks if the solution panel correctly displays results	Incomplete

Table 6.2 Sample test cases

7. Conclusion

Software effort estimation tools have been proven to be an essential tool for planning large IT projects to meet requirements of budget, time and more. This has led to research for decades to more accurately estimate the efforts required to complete an IT project and albeit challenging, estimating the effort required during the initial phase would greatly benefit the people involved (Carvalho et al. , 2021). Our team hopes that our proposed software effort estimation model would greatly benefit all software managers alike and help them more accurately predict effort requirements of projects when implemented. By implementing the agile methodology to manage the project, we intend to manage tasks among team members much more efficiently and timely, and to manage resources such as existing literature efficiently during the project lifecycle and finally to produce the final effort estimation model on time by the end of the next semester.

8. Abbreviations

SGD: Stochastic gradient descent

KNN: K-nearest neighbor

DT: Decision trees

BR: Bagging regressor

RFR: Random forest regressor

ABR: Ada-Boost regressor

GBR: Gradient boosting regressor

ELM: Extreme Learning Machine

SVM: Support Vector Machine

LR: Linear Regression

MLP: Multi-Layer Perceptron

BN: Bayesian network

RR: Ridge Regression

ANN: Artificial Neural Networks

OLS: Ordinary Least Squares Regression

PSO: Particle Swarm Optimization

GA: Genetic Algorithms

MRE: Magnitude relative error

MMRE: Mean magnitude of relative error

RMSE: Root mean square error

MAE: Mean absolute error

MdMRE: Median magnitude of relative error

MSE: Mean square error

R^2: Coefficient of Determination

PRED(x): Percentage of estimates that are within x% of the actual efforts

MBE: Mean Balanced Error

9. References

Choetkiertikul, M., Dam, H. K., Tran, T., Pham, T., Ghose, A., & Menzies, T. (2019). A Deep Learning Model for Estimating Story Points. *IEEE Transactions on Software Engineering*, *45*(7), 637–656. https://doi.org/10.1109/tse.2018.2792473

Das, S., Dey, A., Pal, A., & Roy, N. (2015). Applications of artificial intelligence in machine learning: review and prospect. *Int J Comput Appl*, *115*(9). https://www.researchgate.net/publication/276178017_Applications_of_Artificial_Intelligence_in_Machine_Learning_Review_and_Prospect

de Carvalho, H. D. P., Fagundes, R., & Santos, W. (2021). Extreme Learning Machine Applied to Software Development Effort Estimation. *IEEE Access*, *9*, 92676–92687. https://doi.org/10.1109/access.2021.3091313

Fu, M., & Tantithamthavorn, C. (2022). GPT2SP: A Transformer-Based Agile Story Point Estimation Approach. *IEEE Transactions on Software Engineering*, 1. https://doi.org/10.1109/tse.2022.3158252

Kocaguneli, E., Tosun, A., & Bener, A. (2010). AI-Based Models for Software Effort Estimation. *2010 36th EUROMICRO Conference on Software Engineering and Advanced Applications*, 323–326. https://doi.org/10.1109/SEAA.2010.19

Malgonde, O., & Chari, K. (2018). An ensemble-based model for predicting agile software development effort. *Empirical Software Engineering*, *24*(2), 1017–1055. https://doi.org/10.1007/s10664-018-9647-0

Palaniswamy, S. K., & Venkatesan, R. (2020). RETRACTED ARTICLE: Hyperparameters tuning of ensemble model for software effort estimation. *Journal of Ambient Intelligence and Humanized Computing*, *12*(6), 6579–6589. https://doi.org/10.1007/s12652-020-02277-4

Satapathy, S. M., & Rath, S. K. (2017). Empirical assessment of machine learning models for agile software development effort estimation using story points. *Innovations in Systems and Software Engineering*, *13*(2–3), 191–200. https://doi.org/10.1007/s11334-017-0288-z

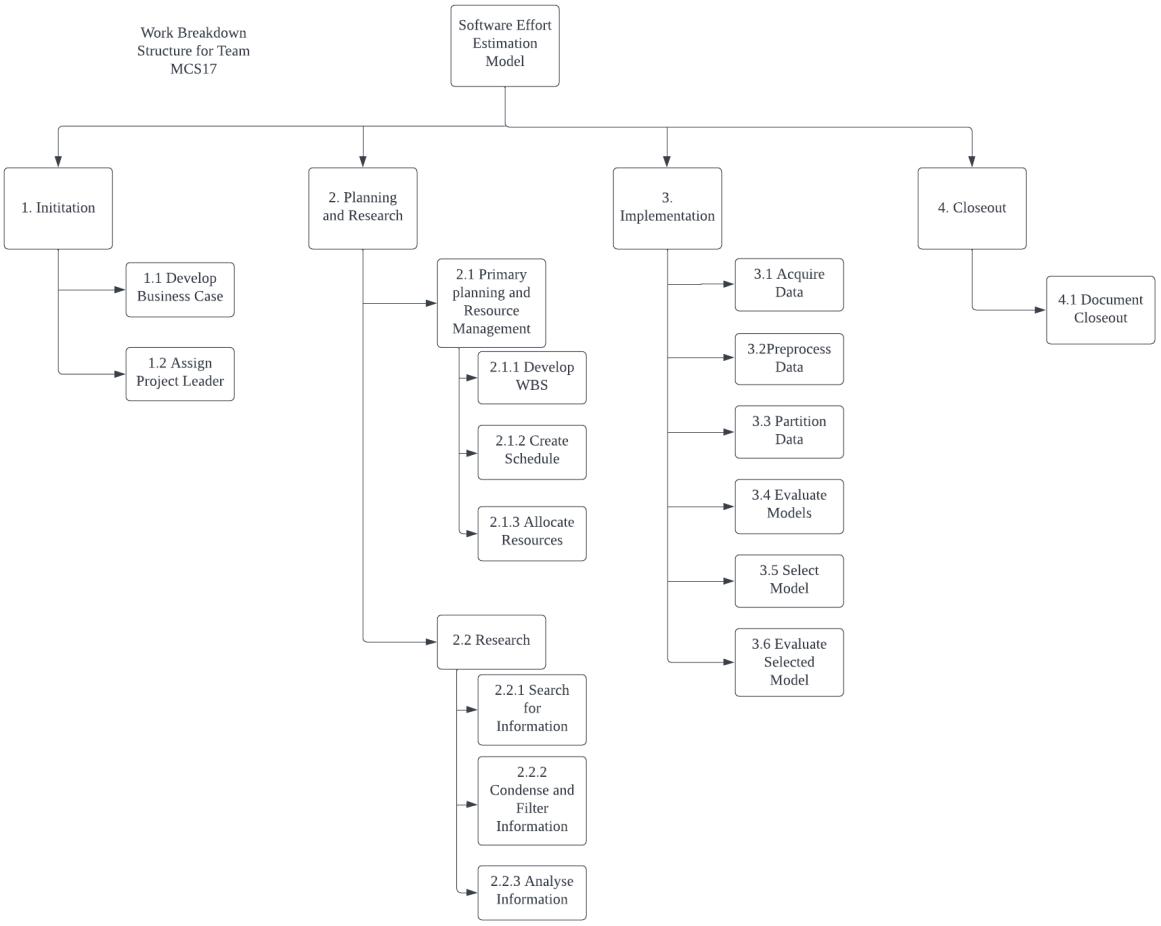
Shepperd, M., & MacDonell, S. (2012). Evaluating prediction systems in software project estimation. *Information and Software Technology*, *54*(8), 820–827. https://doi.org/10.1016/j.infsof.2011.12.008

Suresh Kumar, P., Behera, H. S., Nayak, J., & Naik, B. (2021). A pragmatic ensemble learning approach for effective software effort estimation. *Innovations in Systems and Software Engineering*. https://doi.org/10.1007/s11334-020-00379-y

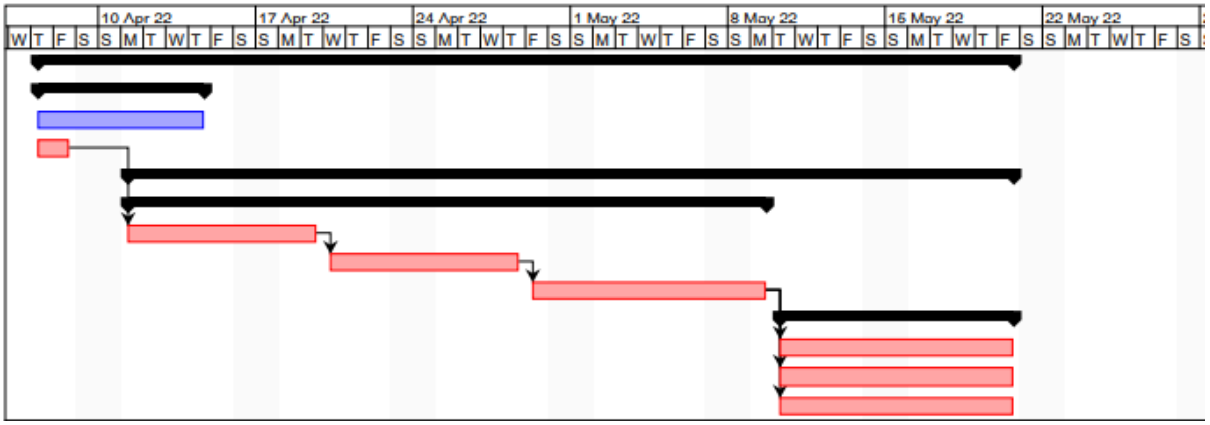
Zia, Z. K., Tipu, S. K., & Zia, S. K. (2012). An Effort Estimation Model for Agile Software Development. *Adv Comput Sci Appl*, *2*(1), 314–324. https://www.researchgate.net/publication/268186219_An_Effort_Estimation_Model_for_Agile_Software_Development

10. Appendix

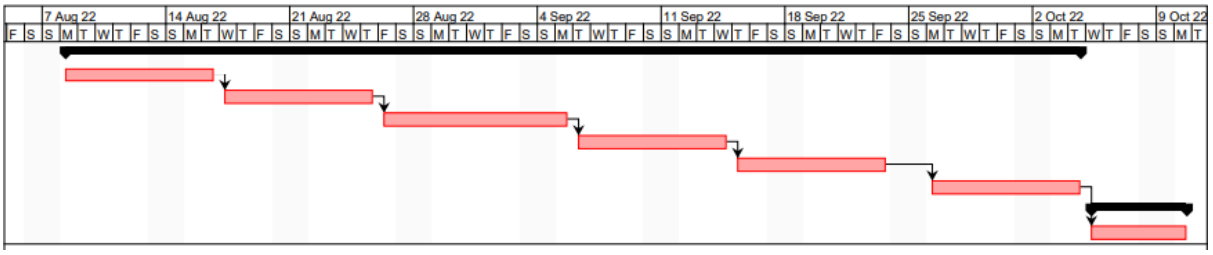
10.1 Appendix A: Work Breakdown Structure



10.2 Appendix B: Gantt Chart



Gantt Chart Part 1 (Apr 2022 - May 2022)



Gantt Chart Part 2 (Aug 2022 - Oct 2022)

10.3 Appendix C: Team Member Contributions

Front cover - Jovan Ong

Table of content - Jovan Ong

Introduction - Jovan Ong

Literature Review - Wong Zi Qi

Scope and Requirements - Wong Zi Qi

Project Organisation (Process Model) - Wong Zi Qi

Project Organisation (Responsibilities) - Jovan Ong

Schedule and Resource Requirements - Benjamin Marc Wijayaratne

Management Process - Benjamin Marc Wijayaratne

External Design - Jovan Ong

Methodology (Stage 1) - Benjamin Marc Wijayaratne

Methodology (Stage 2) - Wong Zi Qi

Hardware and Software Requirements - Benjamin Marc Wijayaratne

Test Planning - Wong Zi Qi

Conclusion - Jovan Ong

Appendix - Benjamin Marc Wijayaratne, Jovan Ong

Style and presentation - Jovan Ong