Fuzzy and Mathematical Effort Estimation Models for Web Applications Development

Jabar H. Yousif*1 and Dinesh Kumar Saini2

1 Faculty of Computing & Information Technology, University of Sohar, Oman
2 Manipal University Jaipur, Department of Computer & Communication Engineering, India.

*Corresponding author: Jabar H. Yousif; jyousif@su.edu.om.

Abstract

This paper proposed an Effort Estimation Model for optimizing the deployment of Web Applications Based Fuzzy and Practical Models. This paper presented the effort estimation model that involves two levels—the first level estimates by Project Managers, and the second level estimates by Project Leaders or Developers. The model considers the classification of each task as either Low or Medium or High complexity. Efforts are estimated to design, code, and test tasks and take a new requirement as a case study, compared with the practical efforts model using historical data for the existing functionalities. The fuzzy logic model verifies the claims made in effort estimation, which proposed a new relation between data and effort value membership for actual data. It converts it into a crisp function in the range [0…1], which helps classify the task's complexity and subtask in the design, coding, and testing phases quickly, low cost, and high accuracy. The proposed effort estimation model would allow the project managers to efficiently control the project, manage the resources effectively, improve the software development process, and trade-off analyses time.

Keywords: Effort Estimation; Fuzzy model; Software Development; Web Applications; COCOMO Models
1. Introduction

The effort estimation is considered as one of the important activities in software development project management. Several researchers have been discussed and modeled the association between software development's main factors, like size and efforts (Al Asheeri & Hammad, 2019). Nevertheless, all these efforts, but still many problems need to solve and still. These problems usually happened in the early phases of a project, such as inconsistent, uncertain, and unclear data. The enhancement of the effort estimation phase is considered a vital tool for software development planning and forecasting the time and cost of developing a software project system (Huang, Ho, Ren, & Capretz, 2007). Creating low-price software development is regarded as an essential feature to produce competitive software. The relation between the need for reliable and accurate software in one direction and the prediction software cost is a challenging matter (Abhishek, Kumar, Vitta, Srivastava, & Applications, 2010). Typically, software effort estimation models can be classified into algorithmic and non-algorithmic models. The main computation method in algorithmic models is based on the statistical analysis of historical data and Software Life Cycle Management (SLIM). Besides, the COCOMO and Albrecht's Function Point models (Bardsiri, Jawawi, Hashim, & Khatibi, 2013). They focused on creating a precise estimation method for determining the values of main factors like the size of software lines of code (LOC), the complexity of each code, the number of interfaces, and user screens (Aloka, Singh, Rakshit, & Srivastava, 2011). The estimation of software project effort or cost has always been challenging despite the incredible amount of research (Araújo, Soares, & Oliveira, 2012). These challenges include the lack of understanding of the software project scope, the time pressure to complete the project, tight budgets, and quantitative results (Anandhi & Chezian, 2014).

The administrators of the web application set these privileges. The application provides the number of allocations made by users to enable billing of the clients for the service provider (Malathi, Sridhar, & Technology, 2012). Several methods are suggested for enhancing the efficiency of effort estimation. One of the most generally applied algorithmic models for estimating effort in the industry is COCOMO (Nguyen, Boehm, Huang, & Software, 2019). Kaushik's (Kaushik, Verma, Singh, Chhabra, & Management, 2017) proposed method (CUCKOO-FIS) combines two optimization models (Cuckoo optimization, Fuzzy Inference System). This method is employed in the software cost estimation for effort optimization, which is trained using the tera-PROMISE datasets. The results show that the proposed model has improved accuracy in cost estimation. Rijwani (Rijwani & Jain, 2016) proposes an Artificial Neural Network (ANN) model using Multi-Layered Feed Forward Neural Network (MLFF) with Back Propagation learning method. The proposed model shows better results and accurately predicted the software estimation effort.
Thamarai (Thamarai, Murugavalli, & Technology, 2016) introduces a software effort estimation based on a differential evolution algorithm called DEAPS, which is examined on the Desharnais dataset. The results prove that the proposed method develops the ability of exploration. Bardsiri (Bardsiri et al., 2013) introduces a hybrid estimation model based on a particle swarm optimization (PSO) algorithm and an Analogy-based estimation (ABE). This study obtained accurate estimates of results based on categorical and non-categorical datasets. Also, it helps to improve the performance of existing estimation models.

This paper's main focus is to build a fuzzy effort estimation model for design, development, and the unit test of a web application, either new or enhancement. This model will estimate based on the effort data available for past releases.

2. Existing Effort Estimation Model

The existing effort estimation model for design and development establishes a pattern that resembles features to be implemented in the new enhancement (Kumar, Behera, Kumari, Nayak, & Naik, 2020). The model compares the practices implemented in earlier releases and finds out a close match. The effort required to implement the matched pattern(s) in the previous releases is used to predict the future effort (Saini & Yousif, 2018). The patterns are matched based on specific metrics like (the Number of controls added, Number of files accessed, and Nature of database access).

The web application is classified under various modules like:

- Graphical User Interface Design like adding, modifying and deleting controls
- Graphical User Interface Functionality
- Database schema
- Generation of Reports
- Data Import
- Less/more data centric changes

Based on the requirement analysis, details are worked out in existing patterns, and the estimation is derived for every new enhancement. A data repository is already available, which has the efforts of design and development for various patterns. The model's drawbacks are that it is difficult to measure the similarity levels between the current requirements and those implemented in the past. When the requirements for a feature consists of a small set of tasks like modifying a data element based on a configuration parameter, the estimates were pretty accurate. On the other hand, if the requirements are called for many tasks, then the estimation is made with this model were highly inaccurate (Pandey, Litoriya, & Pandey, 2020). Also, the effort required to arrive at an estimate was very high, and hence it was
considered unproductive. The model provides the estimation in terms of person days, which does not usually reflect the correct effort in person-hours.

3. Proposed Effort Estimation Model

This paper's main focus is to build a fuzzy effort estimation model for design, develop, and the unit test of a web application, either new or enhancement. This model will estimate based on the effort data available for past releases. The model is based on each task's complexity, which assumes that the task complexity is either low or medium or high. Each level of complexity also has three sub-levels of complexity. So, there are nine levels of complexity for any task (Du, Ho, & Capretz, 2015). Each group of complexity is assigned the effort in terms of person hours for design, coding and unit testing.

The effort involves a bottom-up approach where a task is broken into several subtasks. The subtask could be developed using SQL Server programs or VB programs or ASP program, or any combination (Arnuphaptrairong, Circuits, & Systems, 2013). The efforts of all the subtasks are added to get the total effort required for a task. The total effort for design, coding, and unit testing of each pattern is calculated for any repeated units.

4. Data Collection

The design corresponding to each of the features implemented in a major release for the repository holds the data of programs written in Active Server Pages. The commands executed in the branding example are illustrated in Appendix 1. The estimation process aims to measure historical projects' attributes to arrive at a bottom-up effort estimation model. Bottom-up estimation begins with the lowest level parts of products or tasks to provide estimates for each. Then these estimates are combined to arrive at the higher-level estimates. The effort spent on each subtask is summed in each developer's timesheet that implemented the feature. So, the subtasks were interpreted for a given feature, and complexity was assigned to each subtask. For example, let us consider a feature that was implemented to add search criteria to the web application. The requirement was to search all the corporate cardholders based on their First Name and Last Name. The subtasks to implement this feature are:

1. Add form elements are corresponding to the first name and last name in an ASP page. (Technically, this means adding two text boxes – one for the First Name and the other for Last Name)

2. Add a JavaScript function to validate the user entries into these form elements (Technically, this means a new function needs to be written in JavaScript in the ASP page to validate for entries in mandatory fields and presence of special characters)
3. Modify a JavaScript function submit the form (In the existing web application, a JavaScript function already exists to submit forms to the server. This function needs to be modified to include this new ASP page)

4. Add new stored procedure in SQL Server to return the set of cardholders based on the input (i.e., First Name and/or Last Name) – Corresponds to writing a query to retrieve results from two tables

5. Modify a VB method to call the above-stored procedure by passing the parameters viz— first Name and Last Name, using a Command Object.

6. Adding a server-side function in ASP to display the cardholders (Server-side functions are written in VB script in the web application discussed)

Each of these subtasks is assigned a complexity as shown below.

- SubTask 1 – Low Complexity.
- SubTask 2 – Low Complexity.
- SubTask 3 – Medium Complexity.
- SubTask 4 – Low Complexity.
- SubTask 5 – Low Complexity.
- SubTask 6 – Medium Complexity.

These complexities are assigned based on the actual effort (Mishra, Tripathy, Mishra, & Communication, 2010). The author's organization's set of guidelines is followed wherever the time sheets did not reflect the subtasks correctly after identifying the subtasks belonging to each category. Low, medium, and high, the effort required for each subtask is computed from the timesheet. Appendix 1 shows that to add processes such as "add a new search criterion", the effort taken works out to 20.5 person hours as detailed below.

- SubTask 1 – 1-person hour x (2 form elements) = 2 person hours
- SubTask 2 – 4 person hours
- SubTask 3 – 3 person hours
- SubTask 4 – 1.5 person hours
- SubTask 5 – 4 person hours
- SubTask 6 – 6 person hours

Total Person Hours = 20.5 (for design, coding and unit testing)

This work addresses the non-availability of the subtask details for every new requirement during a high-level design stage by building a list of all possible generic features that could be implemented in the web application and recommending the level of complexity for every feature in the list. This does not limit the estimators to define the complexity perceived based on specific requirement characteristics. The estimator can rate a feature as “Medium” complexity even if the recommended complexity is “Low” and vice versa. The list of all possible generic features and the recommended complexity for SQL, VB, and ASP programs as appropriate is given in Appendix 2. The complexity for each of these features has been arrived based on the complexity of subtasks. Assigning the overall complexity for
each of these features has been decided after consultation with designers and developers (Minku, Yao, & Technology, 2013).

5. Soft Computing & Fuzzy Systems

Soft Computing (SC) is a new computing technique for utilizing real-world problems and provides lower-cost solutions. It mainly consists of the following methods: neural networks, fuzzy systems, and evolutionary computation. Fuzzy systems are implemented in a linguistic framework used to handle linguistic information and then perform approximate reasoning. The significant directions of soft computing applications are executed and conducted into knowledge representation, learning methods, path planning, control, coordination, and decision making (J. H. J. I. J. o. C. A. Yousif, 2015). Furthermore, SC is performed successfully in many applications such as character recognition, data mining, Natural Language Processing (NLP) (J. H. J. I. J. o. C. A. Yousif, 2013), Image processing, Machine control, Software engineering (Saini, Yousif, & Omar, 2009), Information management, etc.

Despite all efforts in using the computation in algorithmic models, it is not reached to produce efficient and reliable software models. As a result, there was a need to explore new methods for solving algorithmic models' limitations. The traditional techniques need to be replaced with non-traditional methods of calculation, such as Parkinson's, and experts estimate and Judgment, Price-to-Win and lastly, the machine learning methodologies (J. H. Yousif, & Saini, Challenges, 2020). The uses of techniques such as neural networks and logic fuzzy are more modern techniques for calculating the estimation models, which work on a few and inaccurate data and produced accurate and reliable results.

Fuzzy logic is an expert knowledge-based approach with powerful linguistic representation for epitomizing imprecision in input and output data sets for model building. Fuzzy systems are suitable for the uncertain or approximate reasoning function based on fuzzy rules, and it can be attuned by tuning the rules. The fuzzy set methods are appropriate for linguistic reasoning modes of nature to humans. It is used the concept of crisp sets. Data's robustness and flexibility are implemented by removing the sharp boundary between members and non-members of a group (Nassif, Azzeh, Idri, Abran, & neuroscience, 2019). The fuzzy set of an input variable's membership function is mapping a universe of discourse in the interval $[0, 1]$. Let $X$ is a non-empty set, and then the membership or containment of $X$ in a fuzzy set is "$A$" which decides an attribute membership function $\mu A(x) \in [0, 1]$. It is mathematically expressed as in Eq. 1:

$$ A = \sum_{i=1}^{n} \mathbb{I}(\frac{H_a(x_i)}{x_i}) $$

(1)
6. Implementation of the Estimation Model (Case Study)

This effort estimation model is used to estimate the effort based on new requirements in vogue. The estimated effort has been compared with the actual effort for three of the features implemented. The description of these three features is described below.

1. Branding Changes – corresponds to Graphical User Interface Design changes
2. Modification to Reports – corresponds to more data centric utility
3. Handling of transaction disputes – corresponds to Graphical User Interface Functionality.

As a case study, let us consider the estimation for branding changes.

The requirements for the branding changes are the following:

1. Add 8 new menu items.
2. Add 12 new 1st level sub menu items.
3. Add 9 new 2nd level sub menu items.
4. Add functionalities for Mouse_Over and Mouse_Out events for each menu and sub menu item.
5. Associate a path with every menu and sub menu item.
6. Add graphics to each menu and sub menu item.

Based on Appendix 2, the estimation for the design effort is as given in Table 1. The branding changes involved development is worked only in ASP pages, the estimation was done by taking the values listed for ASP pages in each sub task. The Table 3 illustrates a case study – effort estimation for coding based on practical calculations.

The total effort estimated for coding works out to 204.6 person hours. The actual effort was 189 person hours. The variance works out to about 8.2%. The Table 3 depicts the use case of the calculation effort estimation unit in testing phase. The estimated total effort for unit testing works out to 69.5 person hours. The actual effort was 88 hours. The variance works out to about 26.6%. Based on the above comparison between estimated effort and actual effort, it can be concluded that the effort estimation model can be used to estimate with a variance of about 25%. This variance is very much acceptable for first level estimation.

The practical and estimation calculation shows that the complexity of branding changes is only medium and high. But this is not reflecting the real case of the efforts spend in different stages. For example, if the user wants to add 8 new menu items, then the estimated effort is 5.5 (the first row in Table 1), and complexity is high. While if the user wants to add 12 new sub-menu items first, then the estimated effort is 7.5 (the second row in Table 1), and complexity is medium. This will make a vague understanding of the meaning of complexity values. Now, we will calculate the estimation of efforts based on the Fuzzy model.
Table 1: Case Studies – Effort Estimation for Design

| SN. No. | Description                        | Complexity | Basic Effort | Multiplication Factor | Calculation                   | Estimated Effort, Person Hours |
|---------|------------------------------------|------------|--------------|-----------------------|-------------------------------|--------------------------------|
| 1       | Add 8 new menu items               | High       | 2.0          | 0.25                  | 2.0+(0.25x2x7)                | 5.5                            |
| 2       | Add 12 new 1st level sub menu items| Medium     | 2.0          | 0.25                  | 2.0+(0.25x2x11)               | 7.5                            |
| 3       | Add 9 new 2nd level sub menu items | Medium     | 2.0          | 0.25                  | 2.0+(0.25x2x8)                | 6.0                            |
| 4       | Add Mouse Over Event for each item | Medium     | 2.0          | 0.6                   | 2.0+(0.6x2x28)                | 35.6                           |
| 5       | Add Mouse Out Event for each item  | Medium     | 2.0          | 0.6                   | 2.0+(0.6x2x28)                | 35.6                           |
| 6       | Add Graphic to each item           | Medium     | 1.0          | 0.3                   | 1.0+(0.3x1x28)                | 9.4                            |
| 7       | Associate path with each menu       | Medium     | 2.0          | 0.5                   | 2.0+(0.5x2x28)                | 30.0                           |

Table 2: Case Studies – Effort Estimation for Coding

| SN. No. | Description                        | Complexity | Basic Effort | Multiplication Factor | Calculation                   | Estimated Effort, Person Hours |
|---------|------------------------------------|------------|--------------|-----------------------|-------------------------------|--------------------------------|
| 1       | Add 8 new menu items               | High       | 8.0          | 0.25                  | 8.0+(0.25x8x7)                | 22.0                           |
| 2       | Add 12 new 1st level sub menu items| Medium     | 4.0          | 0.25                  | 4.0+(0.25x4x11)               | 15.0                           |
| 3       | Add 9 new 2nd level sub menu items | Medium     | 4.0          | 0.25                  | 4.0+(0.25x4x8)                | 12.0                           |
| 4       | Add Mouse Over Event for each item | Medium     | 3.0          | 0.6                   | 3.0+(0.6x3x28)                | 53.4                           |
| 5       | Add Mouse Out Event for each item  | Medium     | 3.0          | 0.6                   | 3.0+(0.6x3x28)                | 53.4                           |
| 6       | Add Graphic to each item           | Medium     | 2.0          | 0.3                   | 2.0+(0.3x2x28)                | 18.8                           |
| 7       | Associate path with each menu       | Medium     | 2.0          | 0.5                   | 2.0+(0.5x2x28)                | 30.0                           |

Table 3: Case Studies – Effort Estimation for unit Testing

| SN. No. | Description                        | Complexity | Basic Effort | Multiplication Factor | Calculation                   | Estimated Effort, Person Hours |
|---------|------------------------------------|------------|--------------|-----------------------|-------------------------------|--------------------------------|
| 1       | Add 8 new menu items               | High       | 1.0          | 0.25                  | 1.0+(0.25x1x7)                | 2.75                           |
| 2       | Add 12 new 1st level sub menu items| Medium     | 1.0          | 0.25                  | 1.0+(0.25x1x11)               | 3.75                           |
| 3       | Add 9 new 2nd level sub menu items | Medium     | 1.0          | 0.25                  | 1.0+(0.25x1x8)                | 3.0                            |
| 4       | Add Mouse Over Event for each item | Medium     | 1.0          | 0.6                   | 1.0+(0.6x1x28)                | 17.8                           |
| 5       | Add Mouse Out Event for each item  | Medium     | 1.0          | 0.6                   | 1.0+(0.6x1x28)                | 17.8                           |
| 6       | Add Graphic to each item           | Medium     | 1.0          | 0.3                   | 1.0+(0.3x1x28)                | 9.4                            |
| 7       | Associate path with each menu       | Medium     | 1.0          | 0.5                   | 1.0+(0.5x1x28)                | 15.0                           |
The fuzzy model's implementation is discussed, and the case study of the effort estimation for design in Table 1 is implemented as a fuzzy model as depicted in Figure 1. Table 4 presents the relationship between the amount of data in the application and the basic estimation efforts parameters.

Table 4: Complexity matrix for external input (EI)

| FTR (File Type References) | 1-5 | 5-15 | >15 |
|-----------------------------|-----|------|-----|
| 0-1                         | Low | Low  | Medium |
| 2                           | Low | Medium | High |
| 3 or more                   | Medium | High | High |

Usually, any fuzzy system consists of three main phases, commonly referred to as, Fuzzification, rule Evaluation, and defuzzification. However, Fuzzifier converts the crisp input to a linguistic variable using the membership functions stored in the fuzzy knowledge base. Then using If-Then type fuzzy rules converts the fuzzy input to the fuzzy output. Therefore, can translate the values with (0/1) in the file type references in the meaning of rules for external input (EI) with 0 or 1 file type references as follows:

R1: If the number of data in the range [0…4], then complexity is a low.
R2: If the number of data in the range [5…15], then complexity is a low.
R3: If the number of data is greater 15, then complexity is a medium.

Also, the values with 2 file type references can be translated in the meaning of rules for external input (EI) as follows:

R4: If the number of data in the range [0…4], then complexity is a low.
R5: If the number of data in the range [5…15], then complexity is a medium.
R6: If the number of data is greater 15, then complexity is a high.

Lastly, the values with 3 and more file type references can be translated in the meaning of rules for external input (EI) as follows:

R7: If the number of data in the range [0…4], then complexity is a medium.
R8: If the number of data in the range [5…15], then complexity is a high.
R9: If the number of data is greater 15, then complexity is a high. Defuzzifier converts the fuzzy output of the inference engine to crisp using membership functions analogous to the ones used by the fuzzifier.

The membership function must be computed and transferred as a crisp function in the range of [0…1] as depicted in Table 5.
Table 5: The crisp values of each estimated effort

| Complexity | Basic estimated effort | Crisp function |
|------------|------------------------|----------------|
| 3          | 5.5                    | 0.1544         |
| 2          | 7.5                    | 0.2106         |
| 2          | 6.0                    | 0.1685         |
| 2          | 35.6                   | 1              |
| 2          | 35.6                   | 1              |
| 4          | 9.4                    | 0.2640         |
| 2          | 30                     | 0.8426         |

Figure 2 shows that it can gain a high complexity only if any data number reaches a 1 in crisp function. And the complexity medium is cached if the value of a crisp function in the range of [0.2640 …0.8426]. Finally, the complexity low is acquired if the value of a crisp function in the range of [0… 0.2640].

Figure 1: the relations between the number of data & efforts value

Figure 2: the relations between the complexity & crisp values
Figure 3 shows the MSE for training and cross-validation for the Fuzzy logic model. It is giving evidence that the output of the fuzzy model closely fits the desired data. The data sets are divided into three categories (60% as training data sets, 20% as cross-validation data sets, and 20% as testing data sets). The model achieved a final MSE of (0.0647) in the training phases, and it is reached a minimum MSE of (0.0017) as summarized in Table 6. The fuzzy model will be exchanged the rules [R1 to R9], into a new form based on the crisp function values. Therefore, the rules will be as follows:

R1: If the crisp value is 1, then complexity is a high.
R2: If the crisp value in the range [0.2640 …0.8426], then complexity is a medium.
R3: If the crisp value in the range of [0… 0.2640], then complexity is a low.

The same transformation will be implemented for the other rules for coding, design and testing.

![MSE versus Epoch](image)

**Figure 3:** The MSE for training and cross validation for Fuzzy logic model.

Therefore, the fuzzy logic model is reducing the computation time by replacing the old model (with 9 rules), with new model with only (three rules). Besides, the fuzzy logic model determines that there no need to paid effort for testing the medium stage. The new module is

| Table 6: The results of fuzzy model and Final MSE |
|-----------------|-----|-----|
| **Best Networks** | Training | Cross Validation |
| Epoch #         | 200  | 478  |
| Minimum MSE     | 0.0544 | 0.0017 |
| Final MSE       | 0.0647 | 0.1148 |
7. Conclusion and Recommendations

The effort estimation model discussed in this paper focuses on estimating the effort of design, coding, and unit testing, and the first level estimates are within a variance of about 25%. The variance would improve for second level estimates and is expected to be about 15%. Estimation must be as closer as possible.

The results can also be compared with estimates calculated using function points and other methods that use regression. The first level estimates computed using the effort estimation model have a variance of about 25% compared with the actual effort. This variance is very much acceptable considering that the first level estimates can be tolerable by up to 35%. The proposed effort estimation tool would help the project managers efficiently control the project, manage the resources effectively, and improve the software development process and trade-off analyses among schedule, performance, quality, and functionality. A Soft Computing approach is implemented, and results are verified using the Fuzzy Logic models, which compute the complexity accurately and quickly. A serious problem in the effort estimation models that some subtasks did not use before hasn't any complexity yet. Using the fuzzy models can quickly compute these tasks' complexity and then estimate the effort needed to build the application. Using a fuzzy model helps calculate the complexity of different tasks like design, coding, and testing.

Acknowledgment

The research leading to these results has no Grant Funding.

References

[1]. Abhishek, C., Kumar, V. P., Vitta, H., Srivastava, P. R. J. o. S. E., & Applications. (2010). Test effort estimation using neural network. 3(04), 331.
[2]. Al Asheeri, M. M., & Hammad, M. (2019). Machine Learning Models for Software Cost Estimation. Paper presented at the 2019 International Conference on Innovation and Intelligence for Informatics, Computing, and Technologies (3ICT).
[3]. Aloka, S., Singh, P., Rakshit, G., & Srivastava, P. R. (2011). Test effort estimation-particle swarm optimization based approach. Paper presented at the International Conference on Contemporary Computing.
[4]. Anandhi, V., & Chezian, R. M. (2014). Regression techniques in software effort estimation using cocomo dataset. Paper presented at the 2014 international conference on intelligent computing applications.
[5]. Araújo, R. d. A., Soares, S., & Oliveira, A. L. J. E. S. w. A. (2012). Hybrid morphological methodology for software development cost estimation. 39(6), 6129-6139.
[6]. Arnuphattrairong, T. J. I. J. o. D., Analysis, Circuits, T. f. I., & Systems. (2013). Early stage software effort estimation using function point analysis: An empirical validation. 4(1), 15.
[7]. Bardsiri, V. K., Jawawi, D. N. A., Hashim, S. Z. M., & Khatibi, E. J. S. Q. J. (2013). A PSO-based model to increase the accuracy of software development effort estimation. 21(3), 501-526.
[8]. Du, W. L., Ho, D., & Capretz, L. F. J. a. p. a. (2015). Improving software effort estimation using neuro-fuzzy model with SEER-SEM.

[9]. Huang, X., Ho, D., Ren, J., & Capretz, L. F. J. A. S. C. (2007). Improving the COCOMO model using a neuro-fuzzy approach. 7(1), 29-40.

[10]. Kaushik, A., Verma, S., Singh, H. J., Chhabra, G. J. I. o. S. A. E., & Management. (2017). Software cost optimization integrating fuzzy system and COA-Cuckoo optimization algorithm. 8(2), 1461-1471.

[11]. Kumar, P. S., Behera, H., Kumari, A., Nayak, J., & Naik, B. J. C. S. R. (2020). Advancement from neural networks to deep learning in software effort estimation: Perspective of two decades. 38, 100288.

[12]. Malathi, S., Sridhar, S. J. I. J. o. S., & Technology. (2012). Efficient estimation of effort using machine-learning technique for software cost. 5(8), 3194-3196.

[13]. Minku, L. L., Yao, X. J. I., & Technology, S. (2013). Ensembles and locality: Insight on improving software effort estimation. 55(8), 1512-1528.

[14]. Mishra, S., Tripathy, K. C., Mishra, M. K. J. I. J. o. C. S., & Communication. (2010). Effort estimation based on complexity and size of relational database system. 1(2), 419-422.

[15]. Nassif, A. B., Azzeh, M., Idri, A., Abran, A. J. C. i., & neuroscience. (2019). Software development effort estimation using regression fuzzy models. 2019.

[16]. Nguyen, V., Boehm, B., Huang, L. J. o. S., & Software. (2019). Determining relevant training data for effort estimation using Window-based COCOMO calibration. 147, 124-146.

[17]. Pandey, M., Litoriya, R., & Pandey, P. J. W. P. C. (2020). Validation of existing software effort estimation techniques in context with mobile software applications. 110(4), 1659-1677.

[18]. Rajwani, P., & Jain, S. J. P. C. (2016). Enhanced software effort estimation using multi layered feed forward artificial neural network technique. 89, 307-312.

[19]. Saini, D. K., & Yousif, J. H. (2018). Effort Estimation Model for Developing Web Applications Based Fuzzy and Practical Models.

[20]. Saini, D. K., Yousef, J. H., & Omar, W. M. J. A. S. E. N. (2009). Enhanced inquiry method for malicious object identification. 34(3), 1-5.

[21]. Thamarai, I., Murugavalli, S. J. I. o. S., & Technology. (2016). An Evolutionary Computation Approach for Project Selection in Analogy based Software Effort Estimation. 9(21).

[22]. Yousef, J. H., Saini, D. K. (2020). Big Data Analysis on Smart Tools and Techniques, Cyber Defense Mechanisms: Security, Privacy, Challenges, Taylor and Francis Ltd, CRC Press, 111.

[23]. Yousef, J. H. I. J. o. C. A. (2013). Natural language processing based soft computing techniques. 77(8).

[24]. Yousef, J. H. J. I. J. o. C. A. (2015). Classification of Mental Disorders Figures based on Soft Computing Methods. 117(2), 5-11.

---

**Appendix 1: List of Sub Tasks in ASP Program**

| ASP Items | Sub Task Description                        | Complexity | Effort Person Hours |
|-----------|---------------------------------------------|------------|---------------------|
| 1         | Adding a new menu to Branding               | Medium     | 6                   |
| 2         | Adding a sub-menu to Branding               | Medium     | 4                   |
| 3         | Adding a mouse event to a menu              | Medium     | 3                   |
| 4         | Adding a graphic to a menu/submenu         | Medium     | 1.5                 |
| 5         | Modifying a picture in Branding Header      | Medium     | 1.5                 |
| 6         | Modifying a link in Branding Header         | Low        | 1.5                 |
| 7         | Modifying a link in Branding Footer         | Low        | 1.5                 |
| 8         | Navigating the user to a specific page based on the user action | Low | 1.5 |
| 9         | Adding a java script function to validate user entries | Medium | 4                   |
| 10        | Modifying a java script function to validate user entries | Low | 2                   |
| 11        | Modifying display of GUI content            | Low        | 2.5                 |
### Appendix 2: List of Features with Recommended Complexity

| S.No. | Description                                                                 | SQL  | VB  | ASP |
|-------|-----------------------------------------------------------------------------|------|-----|-----|
| 1     | Check for the existence of a configuration parameter                        | Low  | Low | Low |
| 2     | Identify values in the database based on input                              | Low  |     |     |
| 3     | Update values in the database based on input                                | Medium |    |     |
| 4     | Insert new values in the database based on input                            | Medium |    |     |
| 5     | Update related values in the database based on the master value inserted/updated | Medium |    |     |
| 6     | Loop through a set of values and perform updates                            | High |     |     |
| 7     | Add filters                                                                 | Medium |    |     |
| 8     | Establish referential integrity among entities                               | Medium |    |     |
| 9     | Delete value(s) based on input                                              | Low  |     |     |
| 10    | Delete value(s) based on input and also establish referential integrity      | Medium |    |     |
| 11    | Write a new field into the file                                             | Medium |    |     |
| 12    | Update a field in the file                                                  | Medium |    |     |
| 13    | Select output based on input/filters                                         | Low  |     |     |
| 14    | Select output based on complex logic                                        | High |     |     |
| 15    | Format the output                                                           | Medium |    | Medium |
| 16    | Add a new field to the entity                                               | Low  |     |     |
| 17    | Validate a file                                                             | Medium |    |     |
| 18    | Map source field and destination field                                       | Medium |    |     |
| 19    | Identify the group to which a set of values belong                           | Medium |    |     |
| 20    | Identify entities dynamically based on input                                | High |     |     |
| 21    | Generate Reports                                                            | Medium |    |     |
| 22    | Validate business rules                                                     | Medium |    | Medium |
| 23    | Validate Data Types                                                         | Medium |    |     |
| 24    | Menu Design - Add a new menu                                               | High |     |     |
| 25    | Add a new sub-menu                                                          | Medium |    |     |
| 26    | Add a mouse event to a menu                                                 | Medium |    |     |
|   | Task Description                                                                 | Priority | Effort  |
|---|-------------------------------------------------------------------------------|----------|---------|
| 27 | Add a graphic to a menu/sub-menu                                             | Medium   |         |
| 28 | Modify Branding Header                                                         | Medium   |         |
| 29 | Modify Branding Footer                                                          | Medium   |         |
| 30 | Read a value from registry                                                    | Low      | Low     |
| 31 | Write a value into registry                                                   | Low      | Low     |
| 32 | Add a new search criterion                                                    | Medium   | Medium  |
| 33 | Navigate the user to a specific page based on the user action                 | Medium   |         |
| 34 | Validate user entries                                                         | Medium   |         |
| 35 | Formatting GUI displays                                                       | Medium   |         |
| 36 | Add a new user input field                                                    | Medium   |         |
| 37 | Implement Pagination feature                                                  | Low      | Medium  |
| 38 | Removing a user input field                                                   | Low      |         |
| 39 | Displaying results in the form                                                | Medium   | Medium  |
| 40 | Maintain user state across pages                                              | Medium   | Medium  |
| 41 | Save user's search criteria                                                   | Medium   | Medium  |
| 42 | Adding privileges to user groups                                              | Medium   | Low     |
| 43 | Computing values based on user input                                          | Medium   | Low     |
| 44 | Generate Reports in Complex Formats                                           | Medium   | High    |
| 45 | Tune Performance                                                              | High     | Medium  |
| 46 | Login Process                                                                 | High     | Medium  |
| 47 | Validate Login Privilege                                                      | High     | Medium  |
| 48 | Purge Obsolete Data                                                           | High     | Medium  |
| 49 | Handle encrypted customer data                                                | Medium   | High    |
| 50 | Add a new entity                                                              | Medium   |         |
| 51 | Import Data - design a utility                                                | High     | High    |
| 52 | Print Reports                                                                 | Medium   | Medium  |
| 53 | Add a new menu based on user's privilege                                     | Medium   | Medium  |
| 54 | Add a sub-menu based on user's privilege                                      | Medium   | Medium  |
| 55 | Validate user entries based on business rules                                 | Medium   | Medium  |