Abstract— Software effort estimation is one of the most important tasks in software engineering. Software developers conduct software estimation in the early stages of the software life cycle to derive the required cost and schedule for a project. In the requirements stage, where most software estimation is conducted, the available information is usually imprecise or incomplete. In this paper, a new regression model is created for software effort estimation based on use case point model. Furthermore, a Sugeno Fuzzy Inference System (FIS) approach is applied on this model to improve the estimation. Results show that an improvement of 11% can be achieved in MMRE after applying the Sugeno fuzzy logic approach.

Keywords- Sugeno fuzzy logic, early software estimation, software engineering, Use Case Points

I. INTRODUCTION

Artificial intelligence (AI) has been widely used in many domains such as medicine, economics, business, and others. Recently, the interest of applying AI in software engineering has been soaring. Conducting software estimation is essential in any project as it helps project managers plan, identify risks, and determine the effort and cost of projects. Several cost estimation techniques exist and they can be classified under three main categories [1]. These categories are:

1. Expert Judgment: In this category, a software project estimator tends to use his or her expertise which is based on historical data and similar projects to estimate software. This method is very subjective and it lacks standardizations and thus, cannot be reusable. Another drawback of this method is the lack of analytical argumentation because of the frequent use of phrases such as “I believe that ...” or “I feel that ...” [2].

2. Algorithmic Models: This is still the most popular category in the literature [3]. These models include COCOMO [4], SLIM [5] and SEER-SEM [6]. The main cost driver of these models is the software size, usually the Source Lines Of Code (SLOC). Algorithmic models either use a linear regression equation, the one used by Kok et al. [7] or non-linear regression equations, those which are used by Boehm [4].

3. Machine Learning: Recently, machine learning techniques are being used in conjunction or as alternatives to algorithmic models. These techniques include neural networks, fuzzy logic, neuro-fuzzy, genetic algorithm and regression trees.

II. BACKGROUND

The use case point method was proposed by Karner in 1993 [9] and it is used for size and effort estimation that is based on use case diagrams. The main components of the use case diagrams are actors and use cases. The use case point model (UCP) [9] is used for size and effort estimation that is based on use case diagrams in the UML documents. The UCP model has widely been used in the last decade [10], yet it has several limitations. One of the limitations is that the software effort equation is not well accepted by software estimators because it assumes the productivity factor of the regression model. This has led to an additional 5% improvement in the effort estimation.

None of the above techniques are perfect and can fit all situations [8]. In this paper, a machine learning technique (fuzzy logic) is used with an algorithmic model (use case point model) for better software estimation results.

Software estimation is important in the early stages of the software life cycle. As UML diagrams have become popular in the last decade, software developers have become more interested in conducting software estimation based on UML models and especially the use case diagrams. The use case diagram conveys the functional requirements of a system and it is usually included in the Software Requirements Specification (SRS) documents. One of the models that is based on the use case diagrams is the use case point model (UCP) [9]. The UCP model has widely been used in the last decade [10], yet it has several limitations. One of the limitations is that the software effort equation is not well accepted by software estimators because it assumes the relationship between software effort and size is linear. In this paper, a new model based on regression analysis is introduced to tackle the limitations of the UCP model and has enhanced the software effort estimation by 6%. A Sugeno FIS has been used to adjust the crisp values of the productivity factor of the regression model. This has led to an additional 5% improvement in the effort estimation.

The remainder of the paper is organized as follows: Section II presents the background of the work. Section III states the problem definition whereas Section IV lists some related work. Section V and Section VI demonstrate the proposed model. Section VII evaluates the proposed approach. Finally, Section VIII concludes the paper and proposes future work.
• Complex actor: This is described as a system interacting through a graphical user interface (GUI). Its weight is 3.

The UAW is calculated as:

$$UAW = \sum SA \times 1 + \sum AA \times 2 + \sum CA \times 3$$  \hspace{1cm} (1)

Where SA, AA, and CA correspond to Simple Actors, Average Actors and Complex Actors, respectively.

2. **Unadjusted Use Case Weight (UUCW):** Use cases are classified based on the number of transactions in the success and alternative scenarios. A weight is assigned to each category as follows:

- Simple Use Case: A use case is classified as Simple if the number of transactions is \( \leq 3 \). Its weight is 5.
- Average Use Case: A use case is classified as Average if the number of transactions is between 4 and 7. Its weight is 10.
- Complex Use Case: A use case is classified as Complex if the number of transactions is more than 7. Its weight is 15. The UUCW is calculated as:

$$UUCW = \sum SU \times 5 + \sum AU \times 10 + \sum CU \times 15$$  \hspace{1cm} (2)

Where SU, AU, and CU correspond to Simple Use Case, Average Use Case and Complex Use Case.

3. **Unadjusted Use Case Points (UUCP):** This is the summation of UAW with UUCW. This is described as:

$$UUCP = UAW + UUCW$$  \hspace{1cm} (3)

The Unadjusted Use Case Points are adjusted based on the technical and environmental factors.

4. **Technical Factor (TF):** These factors contribute to the complexity of the project. The technical factors are depicted in Table I.

The technical factor (TF) is calculated as follows:

$$TF = C1 + C2 \sum_{i=1}^{13} F_i \cdot W_i,$$  \hspace{1cm} (4)

<table>
<thead>
<tr>
<th>( F_i )</th>
<th>Factors contributing to complexity</th>
<th>( W_i )</th>
</tr>
</thead>
<tbody>
<tr>
<td>( F_1 )</td>
<td>Distributed systems</td>
<td>2</td>
</tr>
<tr>
<td>( F_2 )</td>
<td>Application performance objectives</td>
<td>1</td>
</tr>
<tr>
<td>( F_3 )</td>
<td>End user efficiency</td>
<td>1</td>
</tr>
<tr>
<td>( F_4 )</td>
<td>Complex internal processing</td>
<td>1</td>
</tr>
<tr>
<td>( F_5 )</td>
<td>Reusability</td>
<td>1</td>
</tr>
<tr>
<td>( F_6 )</td>
<td>Easy installation</td>
<td>0.5</td>
</tr>
<tr>
<td>( F_7 )</td>
<td>Usability</td>
<td>0.5</td>
</tr>
<tr>
<td>( F_8 )</td>
<td>Portability</td>
<td>2</td>
</tr>
<tr>
<td>( F_9 )</td>
<td>Changeability</td>
<td>1</td>
</tr>
<tr>
<td>( F_{10} )</td>
<td>Concurrency</td>
<td>1</td>
</tr>
<tr>
<td>( F_{11} )</td>
<td>Special security features</td>
<td>1</td>
</tr>
<tr>
<td>( F_{12} )</td>
<td>Provide direct access for third parties</td>
<td>1</td>
</tr>
<tr>
<td>( F_{13} )</td>
<td>Special user training facilities</td>
<td>1</td>
</tr>
</tbody>
</table>

5. **Environmental Factor (EF):** These factors contribute to the team efficiency and productivity. The environmental factors are presented in Table II.

The environmental factor (EF) is calculated as follows:

$$EF = C1 + C2 \sum_{i=1}^{8} F_i \cdot W_i,$$  \hspace{1cm} (5)

Where \( C1 = 1.4, C2 = -0.03 \) and \( F_i \) is a factor which is equivalent to the \( F_i \) of the technical factor (i.e. between 0 and 5).

6. **Adjusted Use Case Points (UCP):** The UCP is calculated by multiplying the UUCP by the technical and environmental factors as follows:

$$UCP = UUCP \times TF \times EF.$$  \hspace{1cm} (6)

The maximum value of TF is 1.3, if the value of all technical factors is 5. If the project complexity is high and the team efficiency is low, there will be a high risk that this project will fail [11]. Typically, by taking the technical and the environmental factors into consideration, the value of the adjusted use case points (UCP) will be 30% more or less than the unadjusted use case points (UUCP).

7. **Effort:** This is the final stage of the use case point model. Karner proposed 20 person-hours for each UCP. This can be represented as:

$$Effort = \text{Size} \times 20.$$  \hspace{1cm} (7)

where size is the calculated software size in adjusted use cases (UCP) and the effort is measured in person-hours.

III. **Problem Definition**

In Equation 7, Karner proposed a method to calculate the software effort from the software size. However, this method has several shortcomings, which include:

<table>
<thead>
<tr>
<th>( F_i )</th>
<th>Factors contributing to efficiency</th>
<th>( W_i )</th>
</tr>
</thead>
<tbody>
<tr>
<td>( F_1 )</td>
<td>Familiar with Objectory</td>
<td>1.5</td>
</tr>
<tr>
<td>( F_2 )</td>
<td>Stable requirements</td>
<td>2</td>
</tr>
<tr>
<td>( F_3 )</td>
<td>Analyst capability</td>
<td>0.5</td>
</tr>
<tr>
<td>( F_4 )</td>
<td>Application experience</td>
<td>0.5</td>
</tr>
<tr>
<td>( F_5 )</td>
<td>Object oriented experience</td>
<td>1</td>
</tr>
<tr>
<td>( F_6 )</td>
<td>Motivation</td>
<td>1</td>
</tr>
<tr>
<td>( F_7 )</td>
<td>Difficult programming language</td>
<td>-1</td>
</tr>
<tr>
<td>( F_8 )</td>
<td>Part-time workers</td>
<td>-1</td>
</tr>
</tbody>
</table>

394
1. Karner assumed that the relationship between the effort and size is linear. This assumption does not reflect the actual situation in the software industry. For instance, if the effort required to build a software project of size 250 UCP is 5,000 person-hours, the effort needed to build the same project type of size 500 UCP would be more than 10,000 person-hours.

2. In Equation 7, the effort is a function of the adjusted use case point (UCP) size. As explained in the previous section, the UCP encompasses the non-functional requirements of the system and it may increase the original unadjusted use case point (UUCP) by 30%. However, IBM states that non-functional requirements might represent more than 50% of the total effort [12]. This means that the non-functional requirements may virtually increase the unadjusted size by 100%.

This paper presents a novel regression model to tackle these shortcomings. Furthermore, a Sugeno fuzzy logic approach is used to adjust the productivity factor in the proposed model, and thus, enhances the estimation.

IV. RELATED WORK

Some work has been done to enhance the effort estimation of the use case point model and other work was done to build regression models based on Function Points.

Periyasamy et al. [13] extended the UCP by classifying actors into 7 groups. Moreover, the authors proposed new weights for use cases. The weight of a use case is determined based on the number of associations between actors and the use case.

Wang et al. [14] extended the UCP by integrating a fuzzy set theory with Bayesian Belief Networks.

Schneider et al. [11] mentioned that environmental factors should be considered while calculating the effort. The authors suggested counting the number of factor ratings of F1-F6 in Table II (Environmental Factors) that are below 3 and the number of factor ratings of F7-F8 that are above 3. If the total is less than 3, then 20 person-hours per UCP should be used. If the total is 3 or 4, then 28 person-hours per UCP should be used. If the total is 5 or more, then the project team should be reconstructed so that the numbers fall at least below 5. A value of 5 indicates that the project is at significant risk of failure with this team. The main limitation of this method is that the effort required to develop one UCP is either 20 or 28 person-hours.

Jiang et al. [15] and Xia et al. [16] built regression models based on function points using ISBSG data. The main concern of these models is that they ignore the influence of the non-functional requirements on estimation.

As well, machine learning models exist and are used to improve the accuracy of software estimation. Examples of these models include [17], [18], [19], [20], and [21].

None of the existing work deals with creating regression models based on the use case points.

V. UCP REGRESSION MODEL

In this section, a regression model based on the UCP model is introduced. This is an extension to the model proposed in [22]. Section VI presents a Sugeno fuzzy logic approach to improve the proposed regression model.

The general equation of software effort can be represented as [6]:

\[
\text{Effort} = \frac{\text{Complexity}}{\text{Productivity}} \times \text{Size}. \tag{8}
\]

where Complexity is the complexity factor of a project and Productivity is the productivity factor of the team that is developing this project. The first step of the regression model is to discover the non-linear relationship between software size and software effort. For this purpose, regression analysis was applied on several projects that have similar projects complexity and team productivity. Regression analysis assumes that data should be normally distributed [23]. If the histograms of software size and effort were normally distributed, the regression equation would be:

\[
\text{Effort} = a \times \text{Size} + b. \tag{9}
\]

where a and b are constants.

Several experiments were conducted using Minitab to determine how data were distributed. The histograms of software size (Figure 1) and effort (Figure 2) showed that data were not normally distributed. After normalizing data using logarithmic transformation, data (ln size and ln effort) became normally distributed (Figures 3 and 4). The regression equation after logarithmic transformation is:

\[
\ln \text{Effort} = c \times \ln \text{Size} + d. \tag{10}
\]

Where c and d are constants. Equation 10 can be rewritten as:

\[
\text{Effort} = A \times \text{Size}^B. \tag{11}
\]

Using Minitab version 16, the values of A and B are 8.16 and 1.17 respectively. The proposed regression equation is:

\[
\text{Effort} = 8.16 \times \text{Size}^{1.17}. \tag{12}
\]
In our experiments, the Spearman and Pearson coefficients are 0.98 and 0.97 respectively. This shows that the two variables Effort and Size have a strong positive relationship.

The main equation for software effort in the proposed model is expressed as:

$$\text{Effort} = 8.16 \times \frac{\text{Project Complexity}}{\text{Productivity}} \times (\text{Size})^{-1.17}. \quad (13)$$

The second step of the proposed model is to calculate the values of Project complexity and Productivity. Although the technical factors in Table I do not include all complexity factors, yet, Karner’s technical factor TF can represent the Project Complexity factor during the estimation of UCP and consequently, the Project Complexity factor in Equation 13 can be ignored.

With respect to productivity, Table II lists some productivity attributes. A better approach than Schneider has been taken to calculate the Productivity factor. Based on Table II, the highest Productivity factor is achieved when the value of the factors F1 to F6 is 5 and the value of the factors F7 and F8 is 0. This implies that the value of $(\sum_{i=1}^{8} F_i \times W_i)$ is 32.5. On the other hand, the lowest productivity factor is achieved when the value of F1 to F6 is set to 0 and the value of F7 and F8 is set to 5. This implies that the value of $(\sum_{i=1}^{8} F_i \times W_i)$ is -10. The productivity factor is determined based on the value of $(\sum_{i=1}^{8} F_i \times W_i)$. Table III shows the proposed values of the Productivity.

The final equation of effort estimation in the proposed model becomes:

$$\text{Effort} = \frac{8.16}{\text{Productivity}} \times (\text{Size})^{-1.17}. \quad (14)$$

Where the values of the Productivity are shown in Table III. For instance, Equation 14 addresses the second limitation illustrated in Section III.

The main drawback of the productivity factor shown in Table III is that the values are crisp and there is no graduation of the productivity factor values as the value of $(\sum_{i=1}^{8} F_i \times W_i)$ increases. For instance, if the value of $(\sum_{i=1}^{8} F_i \times W_i)$ is 10, the productivity factor is 0.7, however, if the value of $(\sum_{i=1}^{8} F_i \times W_i)$ is 11, the value of the productivity factor is 1. To tackle this drawback, a fuzzy logic approach based on Sugeno FIS has been used.

VI. FUZZY LOGIC APPROACH

Fuzzy logic is a branch of Artificial Intelligence that has been widely used in many areas such as control and expert systems. This is because fuzzy logic can handle uncertain inputs and fuzzy rules can be subjective.

<table>
<thead>
<tr>
<th>$(\sum_{i=1}^{8} F_i \times W_i)$</th>
<th>Productivity Description</th>
<th>Productivity Factor</th>
</tr>
</thead>
<tbody>
<tr>
<td>Less than 0</td>
<td>Very Low</td>
<td>0.4</td>
</tr>
<tr>
<td>Between 1 and 10</td>
<td>Low</td>
<td>0.7</td>
</tr>
<tr>
<td>Between 11 and 20</td>
<td>Average</td>
<td>1</td>
</tr>
<tr>
<td>Greater than 20</td>
<td>High</td>
<td>1.3</td>
</tr>
</tbody>
</table>

Where $\text{Size}$ is the software size in UCP and $\text{Effort}$ is the software effort in person-hours. For instance, Equation 12 shows the non-linear relationship between Effort and Size and ignores the Complexity and Productivity factors. The main equation of software effort is expressed in Equation 13. For instance, Equation 12 tackles the first shortcoming presented in Section III.

To test the accuracy of the regression equation (see Equation 12), we measured the value of the coefficient of determination $R^2$. $R^2$ is the percentage of variation in Effort explained by the variable Size. An acceptable value of $R^2$ is $\geq 0.5$ [24]. The value $R^2$ reported for the regression model in Equation 9 is 0.972. Approximately 97% of the variation in Effort can be explained by the variable Size. This shows a strong correlation between Size and Effort.

To thoroughly test the regression model, Spearman [25] and Pearson [26] coefficients were determined to measure the correlation strength between the Effort and Size. The coefficients range of both Spearman and Pearson is between -1 and 1. The value 0 means that these two variables are not correlated. A positive value represents a positive correlation. Larger coefficient values correspond to stronger correlations. On the contrast, negative values mean negative correlations.

Table III shows the proposed values of the Productivity.

Figure 2. Histogram of Effort

Figure 3. Histogram of ln(Size)

Figure 4. Histogram of ln(Effort)
In fuzzy logic, input and output memberships should be defined as well as fuzzy rules. A membership function is a curve that defines the mapping between an input and an output. Membership functions include Triangular, Trapezoidal, Gaussian Bell and Singleton. There are two main types of fuzzy inference system; Mamdani [27] and Sugeno [28]. Both Mamdani and Sugeno can have the same input (membership functions). However, the main difference between these two models is that the output of Mamdani can take any membership function like the input but the output of Sugeno can be either constant or a straight line. In this paper, Sugeno FIS has been used. Based on the proposed values in Table III, the input membership chosen for the Sugeno FIS is Trapezoidal where the output is constant. Figure 5 shows the input membership function of Sugeno FIS.

There are two main approaches to elicit fuzzy rules [29]. These include:
1. The expert knowledge is translated into if-then rules. A structured model can be used to incorporate these rules. Membership functions and weights of rules can be calibrated using input and output data.
2. No prior knowledge about the system is initially used. A fuzzy model is constructed based on a certain algorithm. Fuzzy rules and membership functions are expected to describe the system behavior. An expert can modify the rules and the membership functions.

In this paper, the first approach will be used. There are four fuzzy rules in the proposed approach. These include:
1. If \((\sum_{i=1}^{n} F_i \times W_i)\) is less than 0, then productivity factor = 0.4.
2. If \((\sum_{i=1}^{n} F_i \times W_i)\) is between 0 and 10, then productivity factor = 0.7.
3. If \((\sum_{i=1}^{n} F_i \times W_i)\) is between 10 and 20, then productivity factor = 1.
4. If \((\sum_{i=1}^{n} F_i \times W_i)\) is greater than 20, then productivity factor = 1.3.

After applying the fuzzy logic approach, the productivity factor has a specific value for each value of \((\sum_{i=1}^{n} F_i \times W_i)\). Table IV shows some of the new values of the productivity factors. The labels IN, PO and PN correspond to \((\sum_{i=1}^{n} F_i \times W_i)\), old productivity factor and new productivity factor respectively. As seen in Table IV, the values of the new productivity factor (PN) are not crisp as the values of the old productivity factor (PO). This leads to better estimation values.

<table>
<thead>
<tr>
<th>IN</th>
<th>PO</th>
<th>PN</th>
<th>IN</th>
<th>PO</th>
<th>PN</th>
</tr>
</thead>
<tbody>
<tr>
<td>-10</td>
<td>0.4</td>
<td>0.4</td>
<td>8</td>
<td>0.7</td>
<td>0.8</td>
</tr>
<tr>
<td>-9</td>
<td>0.4</td>
<td>0.42</td>
<td>9</td>
<td>0.7</td>
<td>0.83</td>
</tr>
<tr>
<td>-8</td>
<td>0.4</td>
<td>0.45</td>
<td>10</td>
<td>0.7</td>
<td>0.86</td>
</tr>
<tr>
<td>-7</td>
<td>0.4</td>
<td>0.46</td>
<td>11</td>
<td>1</td>
<td>0.89</td>
</tr>
<tr>
<td>-6</td>
<td>0.4</td>
<td>0.48</td>
<td>12</td>
<td>1</td>
<td>0.9</td>
</tr>
<tr>
<td>0</td>
<td>0.4</td>
<td>0.55</td>
<td>20</td>
<td>1</td>
<td>1.15</td>
</tr>
<tr>
<td>1</td>
<td>0.7</td>
<td>0.58</td>
<td>21</td>
<td>1.3</td>
<td>1.17</td>
</tr>
</tbody>
</table>

For instance, a complete list of the productivity factor values can be obtained by implementing the proposed Sugeno FIS.

VII. Evaluation

The evaluation of the proposed model was performed using 24 projects that were not included among the projects used in the regression analysis. Software estimation was conducted using Karner’s model, Schneider’s model and the proposed model. In software estimation, most practitioners use MMRE and PRED(x) to calculate the error percentage. MMRE is the Mean of the Magnitude of Relative Error and it is a very common criterion used to evaluate software cost estimation models [30]. The Magnitude of Relative Error (MRE) for each observation \(i\) can be obtained as:

\[
MRE_i = \frac{|Actual \text{ Effort}_i - Predicted \text{ Effort}_i|}{Actual \text{ Effort}_i}.
\]

MMRE can be achieved through averaging the summation of MRE over \(N\) observations:

\[
MMRE = \frac{1}{N} \sum_{i=1}^{N} MRE_i.
\]

On the other hand, PRED(\(x\)) is the percentage of projects for which the estimate falls within \(x\)% of the actual value. For instance, if PRED(30) = 60, this indicates that 60% of the projects fall within 30% error range. Table V shows the evaluation results. The columns Kar, Sch, Pro_B and Pro_A represent Karner, Schneider, the proposed model before applying Sugeno FIS and the proposed model after applying the Sugeno FIS approach.

In table VI, the results show that the proposed model before applying the fuzzy logic approach improves the MMRE of Karner’s model by 6% and Schneider’s method by 2%. After applying the Sugeno FIS approach, 11% improvement over Karner’s model was obtained and 7%.
Secondly, the productivity factors proposed in Table III should be tested with projects of larger sizes by applying the proposed model. Moreover, a Sugeno FIS is applied on the regression model to enhance the accuracy of estimation. Experiments show that the accuracy of software effort can be enhanced by 11% over Schneider’s model. This means that an additional 5% improvement was obtained after applying the fuzzy logic approach. With respect to PRED(x), the proposed model has an adverse effect on PRED(25), but PRED(35) and higher were improved. For instance, the columns IProBK, IProBS, IProAK and IProAS represent the improvement of the proposed model before fuzzy logic over Karner’s model, improvement of the proposed model before fuzzy logic over Schneider’s model, improvement of the proposed model after fuzzy logic over Schneider’s model, and improvement of the proposed model after fuzzy logic over Schneider’s model respectively.

### VIII. Conclusion and Future Work

The use case point (UCP) model has been widely used to estimate software size and effort. The main advantage of the UCP model is that it can be used in the early stages of the software life cycle when the use case diagram is available. The UCP model has some limitations since it assumes that the relationship between software effort and size is linear. Furthermore, the UCP model ignores the influence of non-functional requirements on software effort. To tackle these limitations, a novel regression model has been introduced. Moreover, a Sugeno FIS is applied on the regression model to increase the accuracy of estimation. Experiments show that the accuracy of software effort can be enhanced by 11% by applying the proposed model.

Future work will focus on two main concerns. First, the proposed model should be tested with projects of larger sizes (greater than 5,000 person-hours) when data are available. Secondly, the productivity factors proposed in Table III should be calibrated using a neuro-fuzzy approach.

### REFERENCES


