International Journal on New Computer Architectures and Their Applications (IJNCAA) 1(4): 953-976
The Society of Digital Information and Wireless Communications, 2011 (ISSN: 2220-9085)

A Proposed Framework for Use Case based Effort Estimation using Fuzzy Logic: Building upon the outcomes of a Systematic Literature Review

Mohammed Wajahat Kamal and Moataz A. Ahmed
Information and Computer Science Department,
King Fahd University of Petroleum and Minerals,
Dhahran 31261, Saudi Arabia
{wajahat, moataz}@kfupm.edu.sa

ABSTRACT

Reliable and accurate software development effort estimation has always been a daunting task for project managers. More recently, the use of Use Cases for software effort estimation has gained wide popularity. Researchers from academia as well as industry have shown interest in the Use Case based approaches because of the promising results obtained along with their early applicability. There has been a number of approaches proposed in the literature. However, there is no criteria that could be used to aid practitioners in selecting appropriate approaches suitable for their particular development efforts. In this paper we present a set of attribute-based criteria to classify and compare these approaches and provide such aid to practitioners. The set is also meant to guide researchers interested in proposing new use case-based approaches. The paper conducts a systematic review of a number of representative Use Case-based effort estimation approaches against the criteria. Analysis of the discussion highlights some open issues for future research. Addressing some of the issues, we present and discuss a framework for developing use case-based effort estimation models.

KEYWORDS

Effort Estimation, Use Case, Comparison Criteria, UML and Fuzzy Logic System.

1 INTRODUCTION

Effort is delineated as the amount of labor required to complete a certain work. Software effort estimation is the process of predicting the effort required to develop a software system based on incomplete, crude, uncertain or ambiguous inputs [1], [2]. It deals with the prediction of the most probable cost and time to actualize the required development task. Software effort estimation spawned some of the first attempts at rigorous software measurement, so it is the oldest, most mature aspect of software metrics. Researchers have proposed so many models to be used for effort estimation. One of the main inputs to any effort estimation model is the estimated or the actual software size, e.g., lines of code (LOC). As such, measuring/estimating the software size accurately and also as early as possible is of prime importance [3], [4]. A good size estimate can lead to a good effort estimate. This is a challenging task though, since on one hand, early effort estimates play a vital role when bidding for a contract or determining whether a project is feasible in terms of a cost-benefit analysis [5], [6], [7], [8]. On the other hand, however, early estimates of size, for example

based on requirements specification, are the most difficult to obtain, and they are often the least accurate, because very little detail is known about the project and the product at its start [9]. Furthermore, available details are characterized as being imprecise and uncertain.

Use cases, as being available relatively early during the software development lifecycle, are expected to offer a good estimate of the size of the corresponding future system. Consequently, effort estimation using *use cases* has been gaining popularity and the response to the approach has been received quite well by the research community. Some metrics along with corresponding techniques have been proposed to estimate effort using use case information. Majority of them utilize the basic Use Case Points [10] size metric as a predictor of effort. In our bid to carry out a critical survey of the literature on using *use cases* for software development effort prediction, we discovered that a common ground for assessing and comparing these prediction techniques is not available. Though a few related works are available, there is no significant contribution which explicitly offers an evaluation framework for comparison and evaluates the proposed Use Case based metrics on a common platform [11], [12], [13], [14]. Boehm [15] presented a set of useful criteria (attributes) for evaluating the utility of software cost models. The attributes targeted model-based estimation methods. Similarly, Saliu and Ahmed [16] proposed a set of attributes; theirs targeted soft computing-based effort estimation models though. As such, no criteria were developed to target use case-based models. The primary goal of this work is to fill the void caused by the unavailability of such literature which can help practitioners in selecting appropriate metrics for their respective development efforts and also guide researchers interested in developing new metrics in this domain. Accordingly and based on a comprehensive survey, we identified some set of attributes to be used in assessing and comparing various use case-based approaches for effort prediction. This set of attributes is presented in Section 4.

The rest of paper is organized as follows: Section 2 gives a brief insight about the paradigms and problems associated with using Use Cases. A brief summary of the metrics included in the study is presented in Section 3. Section 4 presents the comparison framework and definitions of the attributes. Section 5 consists of the comparison tables and the actual comparison of the available Use case metrics. Section 6 is the analysis of the comparison findings. The fuzzy logic based framework for effort estimation has been discussed in section 7. Section 8 concludes the paper and presents plans for future work.

2 USE CASE BASED EFFORT ESTIMATION

The history of using use cases for effort estimation started with the development of the Unified Modeling Language (UML) by Jim Rumbaugh, Grady Booch, and Ivar Jacobson of Rational Software Corporation in mid-nineties [17]. Sometime later, UML was incorporated into the Rational Unified Process RUP by Rational Software. Meanwhile, Gustav Karner also of

Rational Software Corporation developed an estimating technique to predict the effort required based on Use Cases, much the same way as Function Points. Karner's technique is known as Use Case Point Method [10] and is incorporated into RUP. It is the basic estimating technique for predicting effort based on use cases.

Use cases are used to capture and describe the functional requirements of a software system. Use Case Models define the functional scope of the system. The Use Case model is relevant and valuable for early size measurement and estimating effort as it employs use cases as input. According to a survey conducted by Neil and Laplante [18], 50% of the software projects have their requirements presented as Use Cases. Based on these facts, the approach to estimate effort using Use Cases has gained popularity and subsequently the basic technique proposed by Karner, UCP has gained more recognition. The idea is more or less same as the Function Points developed by Albrecht [19]. Based on UCP, many techniques have been proposed since then, like Use Case Size Points [20], Extended Use Case Points [21], UCP modified [22], Adapted Use Case Points [23], Transactions [24] and Paths [24] to mention a few. A more detailed description of the aforementioned techniques will be presented in the later sections.

Along with the advantages of using these methods, several issues and concerns about these approaches have also been raised. Few of the problems are as follows; varying complexities in the use case models, adjusting the technical complexity factors and experience factors, classification of use cases and the overall construction of the UCP method. Additionally, there are few problems associated with using Use Cases as well [25], [26]. First, there is no standardized style of writing a Use Case. The variations in the style and formality of writing a Use Case brings about many issues like how to measure the size of the Use Case, and how to classify the Use Case. Second, an important issue with Use cases is the assessment of complexity of the Use Case. In a typical CRUD (Create, Replace, Update, Delete), is it correct to consider the Use Case (UC) as one UC with four scenarios or one UC with one scenario, as all the other scenarios are so similar.

Third, a Use Case represents an external actor's view. In case the system has states, it becomes necessary to define another model to represent this behavior which is quite complex. Fourth, granularity of Use Cases is another big issue. What is the optimum length and what are the details that should be mentioned while describing a Use Case. Fifth, most of the researchers complain about the non-consideration of non-functional requirements in the Use Case models.

This raises the question that, are Use Cases a good choice to depend on for estimating effort? The answer lies with the proper evaluation and investigation of these approaches. Many proposed approaches have addressed these issues satisfactorily and many of them have ameliorated many problems as well. We discuss these approaches and compare them for analysis in the following sections.

3 USE CASE BASED METRICS

In this section, we present a summary discussion of the effort estimation techniques we have selected for comparison. The summary has been presented to help the reader understand the basic idea of each effort estimation technique. The metrics to be compared are as follows:

- Use Case Points (UCP) [10]
- Transactions [24]
- Paths [24]
- Extended Use Case Points (EUCP) [21]
- UCPm [22]
- Adapted Use Case Points (AUCP) [23]
- Use Case Size Points (USP) [20]
- Fuzzy Use Case Size Points (FUSP) [20]
- Simplified Use Case Points (SUCP) [27]
- Industrial use of Use Case Points (IUCP) [3]

Use Case Points: The basic technique proposed by Gustav Karner [10] for estimating effort based on Use Cases. The method assigns quantitative weights to actors based on actor classification as simple, average and complex. The sum of all the weighted actors in the system gives the Unadjusted Actor Weight UAW. Similarly, Use Cases are classified according to their complexity and are assigned quantitative weights. The sum of all the Use Cases in the system gives the Unadjusted Use Case Weight UUCW. The sum of UAW and UUCW gives the Unadjusted Use Case Points UUCP. Then, a number of technical complexity factors and experience factors are weighted and are multiplied to the UUCP to yield Use Case Points UCP. Finally, the obtained Use Case Points are multiplied by the Productivity Factor PF to give the final Effort Estimate. Critics claim Karner's method to be decent with the exception of the non-flexibility in adjusting the Productivity Factor which was later proved to be a major variable affecting the estimation process.

Transactions: A metric proposed by Gabriela Robiolo *et al* [24] for estimating size of software based on the size of Use Cases. It depends on the textual description of a Use Case. A Transaction is defined by a stimulus by the Actor and response by the system. The sum of all the stimuli is the number of Transactions in a particular Use Case. Summing up the transactions for all the use cases in the entire system, the number of Transactions is calculated. In order to estimate the final effort, the Historical Mean Productivity technique was used by the authors [24]. Three major objectives using this metric and the following metric 'Paths' were highlighted by the method which are simplifying the counting method, to obtain different units of measurement that individually may capture a single key aspect of software applications and reducing the estimation error.

Paths: Another metric proposed by [24] which pursues similar objectives as the 'Transaction' metric. It is based on the concept of Cyclomatic complexity which identifies binary and multiple decisions in code. The same idea has been applied in terms of textual descriptions of Use Cases. The method is as follows; obtaining the complexity of each transaction. For obtaining the complexity of each transaction, first count the number of binary decisions, then identify the multiple decisions by

counting the different pathways and subtract one from the number obtained. In the final step, for computing the complexity of each uses case, sum up the complexity value for each transaction.

Extended Use Case Points: The EUCP method proposed by Wang *et al* [21] contains three parts; first, refining the Use Case classification with fuzzy set theory. Second, using a learning Bayesian Belief Network BBN for getting the Unadjusted Use Case Points UUCP probability distribution. Third, using a BBN for generating the effort probability distribution which is derived from UCP. The contribution of this approach is a probabilistic cost estimation model obtained by integrating fuzzy set theory and Bayesian belief networks with the generic UCP method.

UCPm: A slight modification of the Use Case Points method proposed by Sergey Diev [22]. The method stresses more on defining Actors and Use Cases comprehensively. The slight change from the basic UCP method is the calculation of the size of the software product. The 'UUCP' obtained is multiplied with the technical complexity factor 'TCF' to give the size of the software product. To this, environmental factor 'EF', base system complexity factor 'BSC' and pre-defined number of person-hours per use case point 'R' are multiplied. Finally, supplementary effort factor is added to yield the final effort estimate of the software product. The supplementary effort may include activities like writing configuration management scripts or performing regression testing.

Adapted Use Case Points: The basic objective of this method proposed by Mohagheghi *et al* [23] is to develop a technique which fits the incremental model of software development and in situations where requirements specifications are frequently changed. The method follows the structure of the UCP method but with major differences. All actors are assumed to be average without differences in classification. All the Use Cases are assumed to be complex and then later on they are decomposed to smaller use cases and classified as simple or average. The method includes the extended use cases as well and counts them as base use cases. Exceptional flows are also counted as average use cases. The method has very promising results and the major contributions are the adaptation of the UCP method for incremental development and identifying the impact of effort distribution profile on effort estimation results.

Use Case Size Points: Proposed by Braz and Vergilio [20]. The metric focuses on the internal structures of the Use Cases in depth and hence better captures the functionality. The primary factors considered in this metric are the Actors classification, pre-condition classification and post-condition classification, main scenarios, alternate scenarios, exception classification and the Adjustment Factor. The sum of all these factors gives the Unadjusted Use Case Size Points UUSP which is subsequently multiplied by the difference of the technical complexity factor and the experience factor. The results are compared with Function Points and UCP metrics.

Fuzzy Use Case Size Points: Another metric proposed by Braz and Vergilio [20]. The primary factors considered in this metric are the Actors classification, pre-condition classification and post-condition classification, main scenarios,

alternate scenarios, exception classification and the Adjustment Factor. The sum of all these factors gives the Unadjusted Use Case Size Points UUSP which is subsequently multiplied by the difference of the technical complexity factor and the experience factor. The difference between USP and FUSP is in the use of the concept of Fuzzification and Defuzzification. This creates gradual classifications that better deal with uncertainty. Also, it reduces the human influence on the classification of the Use Case elements. The results obtained using this metric are slightly better than the Use Case Size Points metric.

Simplified Use Case Points: The main aim of this method proposed by M. Ochodek *et al* [27] is to simplify the UCP method and the process of Effort Estimation in general. This is not a completely defined metric. The approach used for realizing the objective is the cross validation procedure, which compares different variants of UCP with and without certain factors. Factor Analysis was also performed to investigate the possibility of reducing the adjustment factors. The results from this study include recommending a metric based on rejection of actor weights and rejection of 9 Technical Complexity Factors and 6 Experience Factors.

Industrial Use Case Points: The IUCP method proposed by Edward Caroll [3] is not a defined metric but an amalgamation of different industrial practices used in association with the UCP method to increase the accuracy and reliability of the estimation procedure. The main contribution of this method is the inclusion of the Risk Factor and additional effort for activities other than the development of the software product. Also, in depth analysis of few factors like Performance Analysis, Deliverable Analysis, Schedule Analysis, Defect Analysis, Causal Analysis and Quantitative Management Analysis is mentioned. The importance of using a Process Improvement Cycle is also highlighted.

4 COMPARISON CRITERIA

To compare the proposed metrics, we developed a criterion set consisting of ten attributes, which were chosen carefully to accommodate all the pros and cons of using those metrics. Unfortunately, there is no literature survey available in the specific domain of effort estimation based on Use Cases. As such, there are no previous evaluation attributes available. Nevertheless, few attributes have been borrowed from Saliu's and Ahmed's [16] work as well as Irfan's [25] work which was aimed at evaluating various size metrics. The qualified evaluation attributes and their descriptions are as follows:

Accuracy: The degree of precision or correctness obtained in estimating the effort with reference to a particular approach is termed as Accuracy. It is basically obtained by comparing the effort estimated with the actual effort and checking for deviations. A higher accuracy of an approach validates the efficiency of that approach. Better accuracy implies better reliability [25]. It should be noted that comparing estimation accuracy of various approaches is not easy pertaining to reasons such as different datasets, different definitions of similar terms and

different goals of estimation accuracy [28].

Ease of Use: This attribute implies simplicity of use. How easy it is to use a particular technique/approach? A fact that should be understood is that, the effort required in estimating effort for software development should be minimal. What is the use of a technique which itself requires a lot of time and effort? [26]. Preferably, the approach used should be simple enough to be implemented in a reasonable time frame as Bente Anda [11] states that the UCP method requires little technical insight and effort and hence makes it easy to use in early stages.

Use Case detail considerations: The level of detail considered in evaluating a particular Use Case before using it in the estimation process is important for various reasons. Issues like the granularity of Use Cases, number of scenarios in a Use Case, inclusion of Extended Use Cases with the Base Use Cases, classification of Use Cases as simple and complex are commonly debated among various researchers for the Use Case based estimation methods [14], [22], [23]. This is a valuable attribute for comparing the different approaches related to Use Case based methods.

Factor Inclusion: The effort estimation calculated using the basic UCP method considers various Experience factors and Technical Complexity factors [10]. The variety of other Use Case based approaches we have considered, discard few of these factors and consider them unrequired for the estimation process, whereas few of the approaches consider some additional factors [23], [27]. The attribute will help in analyzing the approaches and contribute in specifying the optimal factors to be considered in the estimation process.

Adaptability: The capability of the model or method to adjust according to new environments and fit the incremental style of development practices is termed as Adaptability of the model. "Incremental or evolutionary development approaches have become dominant. Requirements are changed in successive releases, working environments are shifted and this has been accepted as a core factor in software development" [23]. A method or a model should be adaptive to these changes and if it is otherwise, then the model will have limited usability value.

Handling Imprecision and Uncertainty: Quite a common aspect in all the software development practices is to take account of the imprecisions and uncertainty associated with the processes. We know that there is a reasonable imprecision in estimating the size of software and a lot of uncertainty in predicting various factors associated with developing software [29]. A model which considers these factors is better than a model which doesn't.

Sensitivity: The receptiveness or responsiveness to an input stimulus is called sensitivity. In terms of software development, a model in which the change in estimated effort with respect to a small change in the input values is large or significant is termed as a sensitive model. In Effort Estimation, it is desirable to have low sensitivity models.

Transparency: The visibility of the underlying effort prediction model is termed as transparency. It is desirable to have transparent models as it would provide the experts the ability to

incorporate their opinions based on their knowledge and experience. Empirical research studies have shown prediction models coupled with expert opinions to be better than the prediction systems or the expert alone.

Appropriate use of Productivity Factor: The conversion of estimated points based on Use Cases to Effort requires the multiplication of a factor called productivity factor whose units are person-hours. Initially, Karner [10] proposed a productivity factor value of 20 person-hours, which later turned out to be variable for different projects. An appropriate use of the productivity factor results in close to accurate estimations and reduces the deviations. This is a valuable attribute to distinguish between the available approaches.

Artifacts Considered: This attribute reflects the artifacts that are considered in the implementation of a particular technique or metric. Effort Estimation using Use Cases considers all the functional requirements in a satisfactory way, but a major complaint against the use of this method is that the non-functional requirements are not considered extensively. But, if the artifacts pertaining to non-functional requirements like estimating for reports, schedule spreadsheets, staffing concerns are considered [3], then the method could have a valid defense. The use of artifacts considered by different models is helpful in comparing them.

Empirical Validations: The evaluation and validation of a metric or a model in general is essential. If the model is validated, then the validation criteria and the dataset on which it is validated are considered. Datasets from the industry are considered more reliable than student datasets or datasets from open sources [25]. The empirical validation of a model adds to its credibility as well.

5 COMPARISON BETWEEN THE METRICS

This section presents the actual comparison and evaluation of the qualified metrics. It is worth noting here that we used subjective ratings in evaluating the different approaches. Future work will investigate applying more quantitative objective ratings. A point worth mentioning here is that, all the afore-mentioned metrics have been validated by using real time projects of large companies. The comparisons have been presented in tabulated form for sake of simplicity and ease of understanding. All the tables are followed by a short discussion which summarizes the tabulated information and provides recommendations for the use of certain metrics with respect to the attributes.

1 Accuracy

Metric	Comments
UCP[10]	Relatively good accuracy and promising results. More accurate than expert estimates in few cases and almost equally accurate in some other cases.
Transactions[24]	Good accuracy, close to UCP, lower variability of prediction error, high correlation with actual effort.
Paths[24]	Better accuracy than Transactions and UCP, lower deviation from actual effort, high correlation with actual effort.
EUCP[21]	Better accuracy than UCP as they use Fuzzification and a Bayesian Belief Network to train the system.
UCPm[22]	Relatively good accuracy, less

	calculations required in the method.
AUCP[23]	Very good accuracy, effort calculated using AUCP for release 1 and release 2 were 21% and 17% lower than actual Effort.
USP[20]	Competent accuracy compared to others, but lower error rates.
FUSP[20]	Competent accuracy results with lower error rates, a fuzzified form of USP with minor changes in results.
SUCP[27]	Slight improvement in accuracy. Discarding TCF and EF doesn't cause a negative effect in predicting effort.
IUCP[3]	Perhaps the most efficient and accurate results. Using the process improvement loop, the deviation in prediction has been cut down to 9%, which is a very significant contribution.

Discussion: Even after evaluating all metrics based on their respective results, terming a certain metric better than others is not justified because of many reasons such as different data sets used, differences in the nature of the software projects, environmental and expertise differences, etc. Nevertheless, it is recommendable to use metrics which use machine learning techniques like FUSP. Additionally, the use of industrial practices in the estimation process improves the accuracy of the method. Hence, the use of IUCP is also recommendable.

2 Ease of Use

Metric	Comments
UCP[10]	Very easy to compute effort using UCP. It can be done at the early stages of the development of the life cycle. A rough estimate can also be made just by mental calculation.
Transactions[24]	An easy method involving counting the number of transactions in each Use Case and subsequently the total in a system.
Paths[24]	A relatively complex method to use, involving obtaining the complexity of a transaction by summing up the number of binary decisions and identification and summing up of multiple decisions.
EUCP[21]	A complex method involving fuzzifying the inputs and training the Bayesian Belief Network for estimating effort and consequently defuzzifying the output to obtain a crisp value.
UCPm[22]	An easy method, almost similar to UCP; the only difference being size is calculated as the product of Unadjusted Use Case Weights and the sum of Technical Complexity factors.
AUCP[23]	A complex method compared to other approaches. Involves computing modified Unadjusted Use Case Weights and uses many additional factors such as Adaptation Adjustment Factor (AAF), and Equivalent Modification Factor (EMF) which itself comprises of 6 other factors.
USP[20]	A fairly simple method to calculate the effort. Only lengthy part is to consider the details of use cases and classify them appropriately.
FUSP[20]	A simple method, slightly complex than USP because of the Fuzzification of inputs and Defuzzification of outputs respectively.
SUCP[27]	A method simpler than conventional UCP, this reduces the number of Technical Complexity Factors and Experience Factors by limiting them to 6 only.
IUCP[3]	A simple method similar to UCP, with the additional overhead of calculating for non-functional requirements like documenting reports, spread sheets, etc.

Discussion: Almost all the metrics are subjectively rated equally in terms of 'Ease of Use', with the exception of Paths and AUCP metrics. It is intuitive that since the basic UCP method is quite simple in terms of use, a metric or method which deviates from the norms and structure of the basic method is bound to be relatively complex. Though the EUCP method is mentioned as complex, the rational can be to consider the metrics which use soft computing methods as relatively more time consuming rather than terming them as complex to use. We

recommend SUCP as the metric easiest to use compared to the others with UCP coming a close second.

3 Use Case Detail Considerations

Metric	Comments
UCP[10]	Only considers the complexity classification of a Use Case by counting the number of transactions in a Use Case. Classified as simple, average and complex.
Transactions[24]	Considers only the stimulus by an actor and response by the system, by counting the number of transactions. No other details are considered.
Paths[24]	Identifies binary and multiple decisions in a Use Case. Sums up the number of binary and multiple decisions in a Use Case and consequently for the entire system. No other details are considered.
EUCP[21]	The Use Case classification is refined by considering detailed aspects of a Use Case such as User Interface Screens, pre-conditions, primary scenario, alternative scenario, exception scenario, post-conditions.
UCPm[22]	High level of detail is considered. Scoping of actors, classification of Use Cases as zero weight use cases, duplicated use cases, background process use cases, report use cases. Also considers the granularity of use cases.
AUCP[23]	Initially all Use Cases as considered complex, then are broken down to simple and average based on transactions. Include extended Use Cases as base Use Cases and exceptional flows in a Use Case are also assigned a weight factor of 2.
USP[20]	A detailed classification comprising of pre-conditions, post-conditions, main scenarios, alternate scenarios and exceptional scenarios.
FUSP[20]	The Use Case detailed classification comprises of pre-conditions, post-conditions, main scenarios, alternate scenarios and exceptional scenarios.
SUCP[27]	Considers the complexity classification of a Use Case by counting the number of transactions in a Use Case. Additionally, the cardinality of Use Cases is computed.
IUCP[3]	Similar to UCP, IUCP does not consider any extra Use Case details except the complexity classification.

Discussion: Majority of the metrics base their calculations of size on the number of transactions in a Use Case without considering other details related with use cases. If the metrics were to be ranked according to this attribute or recommended on this basis, Use Case Size Point 'USP' would win the evaluation followed by UCPm and AUCP. The reason for this ranking is quite visible in the tabulated information. USP considers almost all the details associated with a Use Case. UCPm takes it to a further level by classifying use cases by varying levels but misses including the pre-conditions and post-conditions.

4 Factor Inclusion

Metric	Comments
UCP[10]	Includes Actor weights and Use Case weights. Also includes 13 Technical Complexity Factors and 8 Experience Factors.
Transactions[24]	No use of Actor weights and Use Case weights. Does not include any Technical Complexity Factors and Experience Factors.
Paths[24]	No use of Actor weights and Use Case weights. Does not include any Technical Complexity Factors and Experience Factors.
EUCP[21]	Includes Actor weights, Use Case weights, 13 Technical Complexity Factors and 8 Experience Factors.
UCPm[22]	Includes Actor weights, Use Case weights, 13 Technical Complexity Factors, 8 Experience Factors. Additionally, UCPm includes Base System Complexity factor and Supplementary Effort Factor.
AUCP[23]	Actor Weights and Use Case weights are included. All the Technical Complexity Factors and Experience Factors are

	discarded. Includes new factors such as Adaptation Adjustment Factor (AAF), Equivalent Modification Factor (EMF), and Overhead Factor (OF).
USP[20]	Actor weights and Use Case weights are included as per the detailed Use Case classification. Additionally, 14 Technical Complexity factors and 5 Environmental Factors are included.
FUSP[20]	Actor weights and Use Case weights are included. 14 Technical Complexity Factors and 5 Environmental Factors are included.
SUCP[27]	Discards Actor weights and includes only Use Case weights. 9 out of 13 Technical Complexity factors and 6 out of 8 Experience Factors are discarded.
IUCP[3]	Includes Actor weights and Use Case weights. Also includes 13 Technical Complexity Factors and 8 Experience Factors.

Discussion: Perhaps the most debated attribute which can involve lot of future work. The issue is to find the optimum number of factors that are to be considered while estimating effort. Many metrics agree with the standardized thirteen technical complexity factors and the eight experience or environmental factors as proposed by the basic UCP method. SUCP discards nine technical complexity factors and six experience factors. UCPm keeps all the standard factors same but includes additional factors. Few metrics like Transactions, Paths and AUCP discard all the standardized factors but the latter makes up for the non-inclusion by using new factors such as AAF, EMF and OF. As such, we cannot recommend any metric to be the best in terms of this attribute.

5 Adaptability

Metric	Comments
UCP[10]	Very simple and adaptable method. Fits any Use Case modeling environment easily.
Transactions[24]	An adaptable method, worked well with 13 different projects under different environments. Fits the dynamic model of software development. Only needs counting the number of transactions.
Paths[24]	Fairly adaptable. Depends on calculating the complexity of Use cases. Slight difficulty expected in adapting to environments with less experienced teams.
EUCP[21]	Less adaptive as compared with other metrics because of the involvement of the training BBN.
UCPm[22]	Fairly adaptable to different environments. Difficulty with less experienced teams for estimating effort.
AUCP[23]	Perhaps the most adaptable metric. The aim of realizing this metric was to fit the incremental model of development and support environments where Extreme Programming is used.
USP[20]	Slightly less adaptable relatively. The adjustment factors need to be calibrated with each and every changing project and environment.
FUSP[20]	Same as the USP method. Less adaptable relatively.
SUCP[27]	Adaptable in many environments. Applied to 14 industrial and academia projects with relative ease and promising results were obtained. Removal of few factors supports adaptability.
IUCP[3]	A very adaptable metric, perhaps because of the feedback loop and its ability to fit into any mode of operation and environment. The metric has been custom designed to fit any model of development.

Discussion: Almost all metrics qualify well for this attribute. Few of them are more adaptable in terms of their structure, ease of use and lesser difficulty with new and inexperienced teams. An interesting observation is that, the use of soft computing methods like in the case of EUCP, where a learning Bayesian Belief Network is incorporated in the estimation process, it made the metric relatively less adaptable to different working environments. But the validity of this observation can be debatable. AUCP is the most recommended metric in terms of Adaptability.

6 Handling Imprecision and Uncertainty

Metric	Comments
UCP[10]	Doesn't handle imprecision, though it manages to deal with uncertainty up to some extent.
Transactions[24]	Doesn't handle imprecision nor uncertainty.
Paths[24]	It is not designed to handle imprecision and uncertainty.
EUCP[21]	Handles imprecision and uncertainty fairly because of the use of Fuzzy logic and additionally because of the learning Bayesian Belief Network.
UCPm[22]	Not capable of handling imprecision and uncertainty.
AUCP[23]	Does not handle imprecision, but the metric deals with uncertainty satisfactorily.
USP[20]	Is not capable of handling both imprecision and uncertainty.
FUSP[20]	The fuzzified version of USP, and hence it handles imprecision and uncertainty quite well.
SUCP[27]	Does not handle imprecision, nor does it handle uncertainty.
IUCP[3]	A metric tailored to deal with uncertainties but cannot handle imprecision.

Discussion: It is much desirable that in a process like estimation of effort and cost where loads of uncertainty is possible and imprecise estimates are quite common, a metric should account for both the afore-mentioned factors. Unfortunately, most of the metrics don't account for both imprecision and uncertainty. Few of them such as UCP, AUCP and IUCP are capable of dealing with uncertainties but not imprecision. EUCP and FUSP, since they use soft computing techniques account reasonably well for both imprecision and uncertainty and are recommended for use.

7 Sensitivity

Metric	Comments
UCP[10]	The metric is less sensitive to input changes. Can accommodate noise reasonably well.
Transactions[24]	Is less sensitive to changes. A small change to the input i.e. the increase or decrease in the number of transactions of a Use Case will not adversely impact the effort estimated.
Paths[24]	Is moderately sensitive when compared to Transactions metric. If the Use Case details are changed, the number of binary decisions and multiple decisions change considerably. This affects the final estimated effort.
EUCP[21]	Less sensitive because of the Fuzzification and Defuzzification process. Accommodates noise levels easily.
UCPm[22]	Less sensitive as the input factors don't impact the final estimated effort much.
AUCP[23]	A moderately sensitive metric. AUCP incorporates many factors because of which, a slight change in some factors may result in considerable changes to the final estimated effort.
USP[20]	Less sensitive to changes.
FUSP[20]	A slightly less sensitive metric than the USP. It accounts for varying levels of input changes.
SUCP[27]	A lesser sensitive metric. Almost similar to the conventional UCP metric.
IUCP[3]	Not sensitive to input changes. Works the dynamic way and hence accounts for changes anywhere in the process lifecycle.

Discussion: A much desirable attribute for comparison in many fields and not just effort estimation, Sensitivity like 'Use Case Details Consideration' can distinguish between metrics in a very proper way. Unfortunately, it is very difficult to distinguish between the available metrics because of lack of information related with the sensitiveness of the metric inputs and outputs. Nevertheless, few metrics have been classified as lowly sensitive and moderately sensitive. It is worth noting that, using soft computing approaches can minimize the sensitivity of a metric considerably. The IUCP can be recommended for use if Sensitivity is the main concern.

8 Transparency

Metric	Comments
UCP[10]	UCP is not transparent. The equations of the UCP method

	don't give any idea about the way UCP is calculated. As such experts cannot calibrate the factor values of UCP
Transactions[24]	Not transparent. The calculation of size is based on the number of transactions and the final effort is calculated based on Historical Mean Productivity.
Paths[24]	Not transparent. The calculation of size is based on the number of paths and the final effort is calculated based on Historical Mean Productivity.
EUCP[21]	Not transparent. Even though EUCP uses the Bayesian Belief Network for training the prediction system, the visibility of the underlying process is minimal.
UCPm[22]	Not transparent enough. Just allows the expert to calibrate few factors but as a whole the effect of calibrating those factors cannot be determined.
AUCP[23]	AUCP is not transparent, as it follows the UCP method and its associated equations with few modifications.
USP[20]	Not transparent. All the use cases are classified and size is calculated based on training from the historical data.
FUSP[20]	Not transparent. The size and effort are calculated based on historical data.
SUCP[27]	Not transparent. Doesn't allow for any calibrations within the process.
IUCP[3]	IUCP is not transparent. It has the basic equations of the UCP method and only adopts few additional industrial practices, which don't account for transparency.

Discussion: Transparency is a very important factor in effort prediction processes. A metric or a method can be termed as fully transparent if its underlying model is clear enough to be understood and allows the experts to calibrate the input values while knowing what the corresponding results will be obtained. But unfortunately, none of the metrics have taken into account this factor.

9 Appropriate Use of Productivity Factor

Metric	Comments
UCP[10]	Karner described the method and fixed the productivity factor at 20 man-hours per Use Case Point.
Transactions[24]	Effort calculation is based on Historical Mean productivity technique. No involvement of Productivity Factor.
Paths[24]	Effort Estimation is based on Historical Mean productivity technique. No involvement of Productivity Factor.
EUCP[21]	Not much use of the productivity factor. All the calculations are based on adjusting other factors.
UCPm[22]	Uses the productivity factor specified by the conventional UCP method.
AUCP[23]	Productivity factor of 36 man-hours per Use Case is used in addition to other adjustment factors such as AAF, EMF and OF. In case of the overhead factor (OF) not being used, the use of 72 man-hours as productivity factor has been prescribed.
USP[20]	A productivity factor of 26 man-hours is used as per the calculations.
FUSP[20]	Productivity factor of 26 man-hours has been used.
SUCP[27]	Productivity factor of 20 man-hours, 28 man-hours and 36 man-hours has been used as per the requirement of the project under consideration which is appropriate.
IUCP[3]	Productivity factor of 20 man-hours and 28 man-hours has been used as other adjustments are taken care of by the risk adjustment factor and factors like estimating for reports.

Discussion: This attribute has a vital contribution in the comparative analysis. In infant stages of estimating effort based on use cases, there were quite significant variations in estimated effort even though the technical complexity factors and experience factors were properly adjusted. The reason which came in the focus after many years was the inappropriate use of Productivity Factor. Since, Karner proposed a 20 person-hour per use case; it was not changed

for quite some time until variations with it resulted in more accurate effort estimates. SUCP can be recommended for use as it allows variable use of the Productivity Factor with respect to the project. The use of IUCP is also recommended as it provides freedom to the estimators for selecting the appropriate Productivity Factor.

10 Artifacts considered

Metric	Comments
UCP[10]	Does not take into account any additional artifacts.
Transactions[24]	Does not consider any additional artifacts. Deals with the functional requirements only.
Paths[24]	No consideration of additional artifacts.
EUCP[21]	No additional artifacts considered.
UCPm[22]	No additional artifacts are considered.
AUCP[23]	Considered artifacts related to non-functional requirements of the process lifecycle like availability, performance and security.
USP[20]	No consideration of additional artifacts.
FUSP[20]	No additional artifacts are considered.
SUCP[27]	Additional artifacts are not considered.
IUCP[3]	A lot many artifacts have been considered by the IUCP metric. Artifacts like estimating for reports, risk management artifacts, artifacts dealing with performance analysis, deliverable analysis, schedulable analysis and defect analysis are considered.

Discussion: In terms of this study, artifacts imply the inclusion of non-functional requirements in the effort estimation process. As tabulated in the above tables, most of the metrics do not consider any additional artifacts with the exception of the AUCP and the IUCP. AUCP considers important non-functional requirements such as performance and security. IUCP also considers non-functional requirements in addition to including lesser effect artifacts such as Reports documentation etc. As such, both AUCP and IUCP are recommended for use.

11 Empirical Validations

Metric	Comments
UCP[10]	Many empirical validations are available for the use of traditional UCP approach. Many authors have validated the UCP procedure empirically using both Industry datasets as well as Student datasets.
Transactions[24]	Empirically validated using datasets comprising of 13 small business projects distributed across 3 different contexts; an Undergraduate Academic Environment, System and Technology Department at Austral University and a level 4 CMM certified company. The projects are also distributed implementation wise as well.
Paths[24]	The same datasets used to validate the Transactions metric were used.
EUCP[21]	Validated using two industry projects in a Chinese company of 500 employees. Since results show some inconsistency, more evaluation needs to be done with the metric.
UCPm[22]	Not validated using any dataset. The proposed metric is a result of analysis carried out over 50 projects in a period of 2 years as reported.
AUCP[23]	The results of applying this metric were validated using a telecom project of Ericcson and across 2 releases. The authors report more case studies that validated the AUCP metric but information about them has not been specified explicitly.
USP[20]	A case study was done to validate the results of this metric using a real project database of a private company. The metric was validated against Function Points and traditional UCP.
FUSP[20]	Same case study as was used by the USP metric. FUSP was validated against Function Points, traditional UCP and USP itself. Differences between USP and FUSP were also highlighted. The use of these metric needs more validations and more experiments needs to be done.

SUCP[27]	Empirically validated against 7 industrial projects and 7 other projects from the Poznan University of Technology. The range of the actual effort was 277 man-hours to 3593 man-hours. Promising results were obtained. Additionally, a framework was built to evaluate the estimation accuracy of all the 14 projects using this metric.
IUCP[3]	The metric has been validated over a continuous period of 5 years, consisting of 200 projects in a CMM level 5 company. The results are astonishing as the feedback loop helped in reaching 9% deviation with reference to the Actual Effort for 95% of the company's projects.

Discussion: The attribute where in all the metrics are on par with each other. It is interesting to note that all the metrics have been extensively validated using Industrial data sets. As such, we cannot underestimate the evaluations of the proposed metrics in any manner.

6 ANALYSIS

Based on the critical survey and after drawing comparisons between the various Use Case based metrics on a common ground, several shortcomings arose which were anticipated. The comparison brought forth many weak links in the Use Case based estimation process and at the same time highlighted many advantages of using it. The comparison findings can be summarized in tabulated format as shown in table 1.

Nearly all the metrics have been validated extensively using industry datasets and student datasets. This is an onus for the validity of the efficiency and accuracy of the metrics. This is well complemented by the fact that most of them have competent and reliable effort estimates. Most of the proposed metrics are easy to use which makes them more liable to be favored over other techniques and metrics which provide similar results. Adaptability, in terms of usage of the metrics is noteworthy considering that almost all metrics qualify as being fairly adaptable and the case studies involving them verify the fact. Few metrics consider detail classification of the Use Cases with respect to complexity by considering all the aspects related to the implementation of Use Case. Metrics which capture the details are definitely more useful and efficient than metrics which do not consider detailed classification. Also, the inclusion and exclusion of the technical complexity factors and experience factors showed varied results. Mostly, it was generalized that the exclusion of few factors does not have negative impact on the estimation of effort. Many metrics considered the technical complexity factors to be overlapped and hence discarded many such factors.

Sensitivity is an attribute which could not be properly addressed in the comparison. It is due to the fact that enough information was not available to distinguish the metrics from being highly sensitive and lowly sensitive. It is desirable to have metrics and techniques which have low level of sensitivity. Based on our comparison, few metrics were found to be lowly sensitive and few moderately sensitive. Productivity factor is an important concern while estimating effort using Use Cases. It is an important contributor for the conversion of the metric in terms of size to effort. Appropriate use of this factor affects the final estimated results. The degree of correlation between estimated effort and Actual effort can be established satisfactorily if the productivity factor is

rightly used. Most of the proposed approaches don't consider the importance of this factor and focus more on other adjustment factors. Use of expert opinion or analogy can be used to at least appropriately select the Productivity factor.

The two most important and perhaps the negative factors in terms of using Use Case based metrics are the inability to deal with imprecision and uncertainty and little consideration of additional non-functional artifacts. These two attributes show the vulnerability of the Use Case based approach when compared with other approaches. Most of the compared metrics do not account for imprecision with the exception of the metrics using Fuzzy logic and other machine learning techniques. Uncertainty, however, did not seem to have caught enough attention; future research is needed to consider the uncertainty associated with measurements provided by the different metrics.

One of the most important weaknesses of Use Case based approaches was the non-consideration of the non-functional requirements associated with software development processes. Though few metrics attempted to incorporate the artifacts pertaining to non-functional requirements, it is not enough. Any software process depends on both functional and non-functional requirements. A metric or technique which does not consider additional artifacts will have varying levels of deviation in the estimated effort.

Despite few shortcomings and negative aspects, the detailed comparison and evaluations support the fact that estimating effort using Use Cases is justified and that they can be successfully used in the software effort estimation process. The important requirement is that the negative aspects which expose the vulnerability of Use Cases should be addressed. In the same context, if a standardized approach is established to write Use Cases, many issues would be minimized. Alternately, each organization can come up with their own standards of writing Use Cases and keep a check on the standards so that, the estimation process can be generalized using Use Cases. The incorporation of non-functional requirements is an essential paradigm that should be taken care. It would remove lot of pessimism about the reliability and efficiency of the use of Use Case metrics. Lastly, using the process improvement lifecycle as a feedback loop to learn and incorporate efficient techniques should be prescribed by organizations so as to reap the benefits of efficient and accurate effort estimation. Causal Analyses and Quantitative Management Analysis of the reports documented should be carried out on a periodic interval to ensure continuous improvement.

Attributes / Metrics	UCP	Transactions	Paths	EUCP	UCPm	AUCP	USP	FUSP	SUCP	IUCP
Accuracy	H	H	H	M	H	VH	M	H	H	VH
Ease of Use	VH	H	H	L	VH	L	M	M	VH	VH
Use case detail considerations	M	L	L	L	M	H	VH	VH	H	H

Factor Inclusion	M	VL	VL	M	M	H	H	H	H	H
Adaptability	H	H	H	H	H	VH	H	H	VH	VH
Handling Imprecision and Uncertainty	VL	VL	VL	VL	VL	L	VL	L	VL	L
Sensitivity	L	L	M	L	L	M	L	L	L	VL
Transparency	VL	VL	VL	VL	VL	VL	VL	VL	VL	VL
Appropriate use of Productivity Factor	M	VL	VL	L	M	H	M	M	H	H
Artifacts considered	VL	VL	VL	VL	VL	M	VL	VL	VL	H
Empirical Validations	VH	M	M	L	VL	M	M	M	H	VH

VL – Very Low L – Low M –Moderate H – High VH – Very High

Table 1: Subjective comparison of metrics with respect to the comparison criteria

7 A FRAMEWORK FOR USE CASE BASED EFFORT ESTIMATION USING FUZZY LOGIC

Based on the analysis of the comparison results between the various Use Case based metrics, we propose here a generic framework for estimating effort based on fuzzy logic. Fuzzy Logic offers two major properties that are the key to the desirable effort estimation models: ability to handle imprecise information and transparency. One of the limitations of effort estimation models is the dearth of historical data which can be used to develop efficient estimation models. In such a case, expert opinions are very important to accommodate in the estimation model. But this is possible when the model is transparent enough to allow the experts or end-users to understand the underlying model and incorporate the necessary changes. The changes or calibrations can be either to input or intermediate factors of the estimation model. Transparency allows the experts to tune the model and calibrate input or intermediate factors in building and using the effort estimation model. A model which allows coupling of expert opinions outperforms the standalone model or the expert himself. Hence, we propose to incorporate the concept of type-1 fuzzy logic systems in the estimation framework which is known to provide solutions for the aforementioned problems.

7.1 Factors used in Use Case-based Effort Estimation

For the purpose of describing the possible framework for effort estimation, the various factors considered in the literature are presented in the following subsection. This provides a brief insight about the major factors to be considered in the estimation framework. Subsequently, a generic estimation framework is presented in the second subsection to portray the idea of developing the fuzzy estimation inference system.

1. Weighted Actors [3][10][20][21][22][23][27] - Actors are classified as simple, average or complex based on the level of interaction with the system.
2. Weighted Use Cases [3][10][20][21][22][23][27] - Use Cases are classified as simple, average or complex based on the number of transactions within the Use Case.
3. Technical Complexity Factors [3][10][20][21][22][23][27] – Thirteen technical complexity factors are considered in majority of the metrics whereas few metrics discard few technical complexity factors.
4. Experience Factors [3][10][20][21][22][23][27] – Eight experience factors are considered in majority of the metrics whereas few metrics discard few experience factors. These are also called Environmental Factors.
5. Productivity Factor [3][10][20][21][22][23][27] – The factor which translates the number of UCP's or any other size metric into effort in terms man-hours. Initially, Karner proposed a 20 man-hours productivity factor per UCP. More recently, many others have productivity factors of 28 man-hours and 36 man-hours depending on the nature of the development project.
6. Supplementary Effort Factor [22] – The additional effort required to build the product which does not necessarily depend on the size of the product, e.g., effort to write configuration management scripts or to perform regression testing.
7. Equivalent Modification Factor [23] – This factor is used to estimate equivalent use case points for modification of software from a previous release or secondary changes, including perfection of quality attributes.
8. Overhead Factor [23] - OF is used to estimate the total effort based on the effort for Development before System Test.
9. Use Case elements [20] – These are the basic elements of a Use Case like pre-conditions, post- conditions, main scenarios and exception scenarios, etc.
10. Risk Factor [3] – This factor is used to accommodate the risk factors that are not considered elsewhere in the development lifecycle and in the effort prediction process. Few risk factors can be special system level integration requirements, special process specific training, etc.

7.2 Architectural Overview of the Fuzzy Logic System for Effort Estimation

The first step in building a fuzzy logic system hereafter referred to as FLS; is to define the fuzzy sets for each input/internal and output/external attribute. The second step is to formulate the rule base using the linguistic variables for each fuzzy set. The third step is training the FLS to refine the linguistic relationships in the rule base. The fourth step is to validate the performance of the FLS using test data.

The ability to handle the imprecise information available during the early stages of software development is important as discussed earlier. Instead of selecting crisp values for the actor weights and Use Case elements weights, fuzzy sets can be used to model the actor weights and the various Use Case

elements weights. Since, fuzzy sets have overlapping with the adjacent fuzzy sets, the expert can choose between two close values for a single factor based on experience. This would provide freedom to the experts to select from a range of values for all the attributes in the *framework.* Other factors such as technical complexity factors, experience factors, risk factors and supplementary effort factors are also modeled as fuzzy sets.

So all in all, fuzzy sets can be employed to handle imprecision in the estimation process. The objective here is to fuzzify all the factors in the estimation framework. For the sake of illustration, Figure 1 gives an example of a possible architecture of the effort estimation framework.

The architecture is multi-layered (two layers). The output of the first layer is the input for the second layer. The architecture has 4 components. The first component is the FLS which takes 13 technical complexity factors as input and gives a fuzzified value called TCF (Technical Complexity Factor).The second component is another FLS which takes 8 experience factors as input and gives a fuzzified value called EF (Experience Factor). Similarly, the third component is the FLS which takes the SE factor (Supplementary Effort) and Risk factor as input and gives a fuzzified value called AF (Additional Factors).

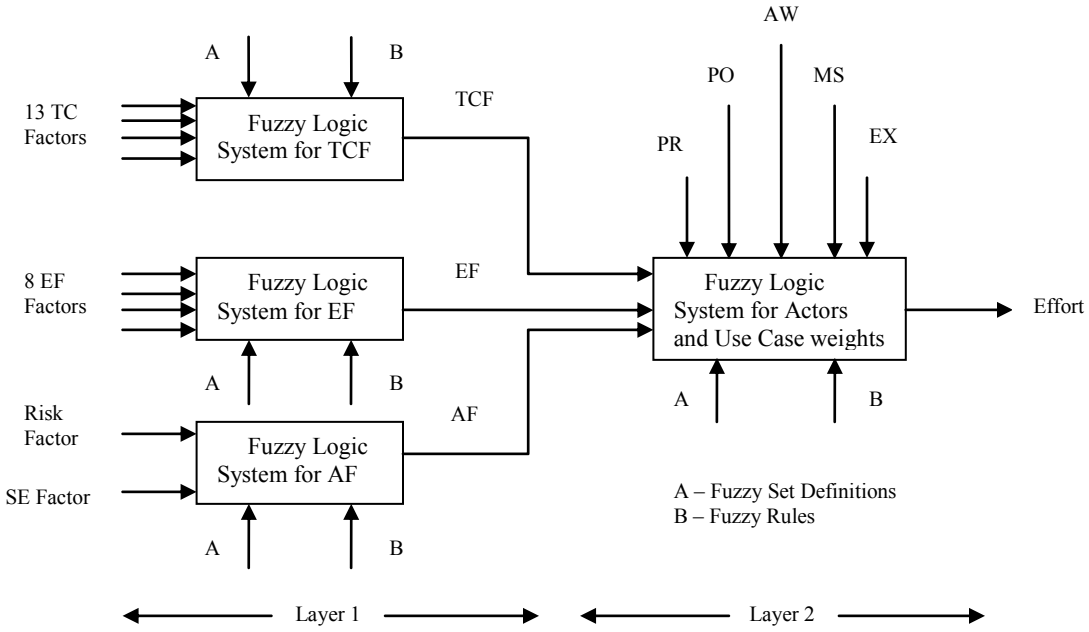

Figure 1: A Possible Fuzzy Logic based multi-layered Effort Estimation Architecture

Likewise, the fourth component is the FLS which takes the TCF, EF, AF, AW (actor weights), and Use Case elements as input and gives the fuzzified value of effort. The various Use Case elements are PR (pre-conditions), PO (post-conditions), MS (Main Scenarios) and EX (Exceptions).

For the purpose of developing the rule base, fuzzy rules need to be formulated. Rules are formulated based on the relationship between the various attributes of the framework. The rules can be as shown below.

- For layer 1;

- IF TCF1 is high AND TCF2 is v.high AND TCF3 is low AND…….. AND TCF13 is nominal THEN TCF is high
- IF TCF1 is nominal AND TCF2 is high AND TCF3 is v.low AND…….. AND TCF13 is low THEN TCF is nominal

…..

- IF EF1 is low AND EF2 is high AND EF3 is v.high AND…….. AND EF8 is nominal THEN EF is nominal
- IF EF1 is v.low AND EF2 is nominal AND EF3 is high AND…….. AND EF8 is low THEN EF is low

…..

- IF SE is high AND RISK is low THEN AF is nominal
- IF SE is nominal AND RISK is v.high THEN AF is high

- For layer 2;

- IF TCF is high AND EF is v.high AND AF is low ANDPR is nominal AND PO is v.low AND MS is high AND EX is nominal AND AW is low THEN EFFORT is nominal
- IF TCF is v.low AND EF is high AND AF is nominal AND PR is low

AND PO is v.high AND MS is low AND EX is v.high AND AW is nominal THEN EFFORT is nominal

After defining the fuzzy sets and formulating the fuzzy rules, the third step is to train the FLS. Training the FLS is required to refine the linguistic relationships in the rule base which is provided by the experts using either the historical data or their experience. The last part is to activate the FLS and validate its performance for predicting effort using test datasets, preferably industrial datasets. This completes the discussion about the estimation framework. As stated earlier, this is just a generic architecture for the estimation framework. The actual architectures may vary depending on the inclusion and exclusion of various attributes within the framework. Future work will involve testing various architectures for the purpose of coming up with a standardized model for estimating effort based on Use Cases.

8 CONCLUSION AND FUTURE WORK

Estimating effort in software development is a difficult and challenging activity. There is no metric or technique which can be preferred over other techniques in all cases and circumstances. Each technique has its corresponding advantages and disadvantages. Nevertheless the focus should be on developing metrics and techniques which complement the desired capabilities. Due to its early applicability during the development lifecycle, use case based metrics have gained wide acceptance recently and have been proven to yield promising results. More research should be dedicated to develop metrics to overcome the negative aspects discussed above though.

Moreover, the variety of use case-based size metrics, which has been proposed, suggests that there may be some inconsistencies among the measurements computed using these metrics. In turn, such inter-inconsistencies raise the concern that relying on measurements of one single metric might not lead to the same estimation of the effort. An obvious important candidate for future work is to study the uncertainty that should be considered when relying on a single metric. The uncertainty arises due to the inability of the metric designer to comprehensively consider all factors that would indeed contribute to the use case size as a predictor for effort; that is neglecting some factors due to the lack of a complete theory of the concept of size and effort, the impracticality to list all the factors that affect the size and effort, etc. In other words, future work should research a framework meant to facilitate portraying the probability distribution of the error associated with measurements computed using a given metric. This, in turn, allows associating a *degree of reliability* to the effort estimated by a given metric; that is a level of how dependable such estimate is.

Even though the evaluation attributes were carefully selected, there may certainly be some additional attributes which can help in better evaluation from a different perspective. Attributes like Sensitivity could not be properly addressed because of lack of insufficient information in the corresponding metrics description. It is much desirable to

distinguish between metrics as being lowly sensitive and highly sensitive. Moreover, as pointed out earlier, the use of subjective ratings in evaluating the different metrics need more clarity; future work will investigate applying more quantitative objective ratings. This would help in recommending a particular metric as the best metric in terms of practical use for software practitioners and software developers.

An important work would be to address the problem of use cases not accommodating non-functional requirements. This is very important in terms of effort estimation as the consideration of non-functional requirements can bring about reasonably large variances in the estimated effort. Typically, in the industry, the common practice to avoid this problem is by including supplementary effort which includes effort pertaining to the non-functional requirements. But this is quite vague. A detailed work like the concept of *misuse cases* for eliciting security requirements by Guttorm Sindre and Andreas Opdahl [30] should be carried out for including the non-functional requirements in a software project.

Another direction in which work needs to be done is to understand whether the inclusion or exclusion of the Technical Complexity factors and Experience Factors brings forth any significant differences in the estimated Effort. In the Adapted Use Case Points [23] approach, all the technical complexity factors and experience factors are excluded and factors like the Adaptation Adjustment factor, Equivalent Modification factor and the Overhead Factor are included. The change in effort as a result of this factor exchange is portrayed to be better compared to the previous approach. In [27], authors have discarded nine technical complexity factors and six experience factors terming those factors as 'not required' in the estimation process. No other techniques have recommended this factor minimization. Analysis needs to be done to understand the effect of factor inclusion in the estimation process.

An interesting observation is the dearth of usage of machine learning techniques in the estimation approaches based on Use Cases. With the exception of Use Case based estimation, many an approach in Effort Estimation utilizes the benefits of the machine learning techniques. Interestingly, there is no work in the literature which uses soft computing in the estimation process. An effort to incorporate fuzzy logic in the estimation process has been attempted by [20] and [21]. Future work in this area is of paramount importance especially given the benefits of using soft computing. In this regard, a general framework for effort estimation using fuzzy logic has been proposed and discussed earlier in section 7.

Based on the conclusions and the possible generic framework discussed earlier, future work can be aptly described by presenting the following 3 research questions which can form the basis for strong research work. The research questions are as follows;

1. How can fuzzy logic be employed to enable the development of transparent use case based effort prediction models capable of incorporating expert opinions?
2. Among the variety of factors considered by different researchers

in the various effort prediction models, which factors are the most influential on the accuracy and which factors can be ignored?

3. Does prediction model architecture impact the models accuracy; e.g., single layer vs. multi-layer?

The answers to these questions can pave way for better effort estimation techniques and practices to be implemented in software development houses and at the same time open more avenues for further research, both in academia and industry.

Acknowledgements. The authors wish to acknowledge King Fahd University of Petroleum and Minerals (KFUPM) for utilizing the various facilities in carrying out this research.

9 REFERENCES

1. Hareton, L., and Zhang F.: "Software Cost Estimation", Department of Computing, Hong Kong Polytechnic University. http://paginaspersonales.deusto.es/cortazar/doctorado/articulos/leung-andbook.pdf
2. Kirsopp, C., Shepperd, M.J., Hart, J.: Search heuristics, case-based reasoning and software project effort prediction, in: Genetic and Evolutionary Computing Conference (GECCO 2002), AAAI, New York, 2002
3. Caroll, E.R.: Estimating Software based on Use Case Points. OOPSLA'05, October 16-20, 2005. ACM 1-59593-193-7/05/0010
4. Lai, R., Huang, S.J.: "A Model for Estimating the Size of a Formal Communication Protocol Specification and Its Implementation", IEEE Transactions on Software Engineering, Volume 29, Issue 1, Jan. 2003 Page(s):46 – 62
5. Boehm, B., Abts, C., and Chulani, S.: "Software Development Cost Estimation Approaches: A Survey", University of Southern California Centre for Software Engineering, Technical Report, USC-CSE-2000-505, 2000
6. MacDonell, S.G., Gray A.R.: "A Comparison of Modeling Techniques for Software Development Effort Prediction", In Proceedings of the 1997 International Conference on Neural Information Processing and Intelligent Information Systems, Denedin, Newzealand, Springer-Verlag (1997), 869-872
7. Ribu, K.: "Estimating Object-Oriented Software Projects with Use Cases", Master of Science Thesis, Department of Informatics, University of Oslo, Oslo, Norway, November 7, 2001
8. Strike, K., El-Emam, K., Madhavji M.: "Software Cost Estimation with Incomplete Data", IEEE Transactions on Software Engineering, Vol. 27, No. 10, Oct. 2001
9. Costagliola, G., Ferrucci, F., Tortora, G., Vitiello, G.: "Class Point: An Approach for the Size Estimation of Object-Oriented Systems", IEEE Transactions on Software Engineering, Volume 31, Issue 1, January, 2005, pp. 52-74
10. Karner, G.: Metrics for Objectory. Diploma thesis, University of Linkoping, Sweden. No. LiTH-IDA-EX-9344:21, December 1993
11. Anda, B., Benestad, H.C., Hove, S.E.: "A multiple case study of Effort Estimation based on Use Case Points" Empirical Software Engineering, 2005
12. Forbes, M.: "Use Case Survey (2009): Towards Adopting Enterprise Standards for Use Cases"
13. Robiolo, G., Orosco, R..: "Employing use cases to early estimate effort with simpler metrics", Innovations System Software Engineering (2008) 4:31–43
14. Smith, J. (1999): "The estimation of effort based on use cases". IBM Rational Software White Paper
15. Boehm, B.: "Software Engineering Economics" Englewood Cliffs, NJ: Prentice-Hall, 1981, ISBN 0-13-822122-7

16. Saliu, M.O., Ahmed, M.A.: "Soft Computing Based Effort Prediction Systems – A Survey", A Chapter in E. Damiani, L. C. Jain, and M. Madravio (EDs), Soft Computing in Software Engineering, Springer-Verlag Publisher, July 2004, ISBN 3-540-22030-5
17. Jacobson, I., Booch, G., Rumbaugh, J. (1999): The Unified Software Development Process. Reading, MA: Addison Wesley
18. Neill, C.J., Laplante, P.A.: "Requirements Engineering: The State of the Practice". Software, vol. 20, no. 6, November 2003, pp. 40-46
19. Albrecht, A.J., Gaffney, J.E.: "Software function, source lines of codes, and development effort prediction: a software science validation", IEEE Trans Software Eng. SE-9, 1983, pp. 639-648
20. Braz, M.R., Vergilio, S.R.: Software Effort Estimation based on Use Cases. Proceedings of the 30th Annual International Computer Software and Applications Conference (COMPSAC'06), 2006 IEEE
21. Wang, F., Yang, X., Zhu, X., Chen, L.: Extended Use Case Points Method for Software Cost Estimation. 978-1-4244-4507-0/09, IEEE 2009
22. Diev, S.: Use Cases modeling and software estimation: Applying Use Case Points. ACM Software Engineering Notes, November 2006
23. Mohagheghi, P., Anda, B., Conradi, R.: Effort Estimation of Use Cases for Incremental Large-Scale Software Development. ICSE' 05 May 15-21, Copyright ACM 1-58113-963-2/05/0005
24. Robiolo, G., Badano, C., Orosco, R.: Transactions and Paths: two use case based metrics which improve early effort estimation. 978-1-4244-4841-8/09, IEEE 2009
25. Ahmad, I.: "A Probabilistic Size Proxy for Software Effort Estimation: A Framework", Master Thesis, Information and Computer Science Department, King Fahd University of Petroleum and Minerals, Dhahran, Saudi Arabia, April 2008
26. Probasco, L.: "Dear Dr. Use Case: What about Function Points and Use Cases?" www.ibm.com/developerworks/rational/library/2870.html
27. Ochodek, M., Nawrocki, J., Kwarciak, K.: Simplifying effort estimation based on Use Case Points. Journal of Information and Software Technology, 0950-5849 © 2010 Elsevier B.V
28. Grimstad, S., Jorgensen, M.: "A Framework for the Analysis of Software Cost Estimation Accuracy", Proceedings of the 2006 ACM/IEEE International Symposium on Empirical Software Engineering pages 58-65
29. Muzaffar, S.Z.: "Adaptive Fuzzy Logic based Framework for handling imprecision and uncertainty in software development Effort prediction models", Master Thesis, Information and Computer Science Department, King Fahd University of Petroleum and Minerals, Dhahran, Saudi Arabia, January 2006
30. Sindre, G., Opdahl, A: "Eliciting security requirements with misuse cases". Requirements Engineering (2005) 10: 34-44