IOSR Journal of Computer Engineering (IOSR-JCE)
e-ISSN: 2278-0661,p-ISSN: 2278-8727, Volume 22, Issue 1, Ser. II (Jan - Feb 2020), PP 10-19
www.iosrjournals.org

Software Development Effort Estimation Using a Fuzzy Logic-Based System within the Context of the Scaled Agile Framework

Hung Q. Tran
CAE USA, FL, USA

Abstract: *In the context of Scaled Agile Framework (SAFe), software development effort estimation represents a critical task. Proper estimation of the development effort enables the development team to plan the development tasks and predict the amount of time and cost required to deliver software products. Additionally, it allows the development team to determine if the staffing level is sufficient (i.e., team's capacity) during the Program Iteration (PI) planning.*

Background*: SAFe is a framework for scaling Agile across an enterprise. At the development team level, it provides guidance to define, build, test, and deploy software solutions in a short iteration timebox. Agile teams are responsible for delivering software results that meet customer needs and expectations. Part of that responsibility is to estimate the effort of the work they must perform.*

Agile teams decompose the required work into stories, which are short descriptions of small pieces (functionalities) of software-desired functionality. The effort to develop and deliver a story is estimated using story points, which is a numeric value that represents a combination of work complexity, amount of work, and uncertainty/risk to implement. The concept of story points is simple, yet often difficult to apply. The main difficulty resides in that a story point represents an abstract subjective value. Many software developers find it challenging to correlate story points to the degree of complexity, size, and uncertainty of work effort.

Objective*: In this paper, we will provide an overview of the software development process within the context of SAFe, specifically, how Agile teams typically estimate work effortby using story points. Then we will propose a novel approach to estimating work effort. This novel method makes essential use of a fuzzy logic-based system—an artificial intelligence technique that emulates the human reasoning process during the estimation of work effort phase by Agile software development teams.*

Conclusion*: Software effort estimation model incorporating fuzzy logic can help software development teams to overcome the issue of estimating multiple attributes conjointly.*

Key Word*:Scaled Agile Framework; Effort Estimation; Story Points; Fuzzy Logic.*

Date of Submission: 15-01-2020 Date of Acceptance: 03-02-2020

I. Introduction

Software development estimation represents the process to determine the required effort to develop, implement, and maintain software based on the customer requirements. Sinhal and Verma[1] published a comprehensive review of a variety of software estimation methods. In their paper, the authors outlined numerous accepted methods that are currently available. Generally, software estimation techniques can be classified into the following categories: algorithm, non-algorithm, and hybrid. Hybrid model is simply a combination of algorithm and non-algorithm estimation techniques [2]. The most popular algorithm estimation model is Boemh'sConstructive Cost Model (COCOMO) [3]. This algorithm model requires model inputs, such as an estimation of the number of lines of code (LOC) or the complexity of software implementation. Such attributes are often difficult to estimate in the early stages of the development process. Due to this limitation, many researchers explored the non-algorithm estimation techniques, which are based on soft computing including fuzzy logic technique [4][5][6].

Dean Leffingwell first introduced Scaled Agile Framework (SAFe) in 2011. SAFe is an industry-proven method that was designed to help manage some of the challenges larger organizations experience while practicing Agile methodology.

The purpose of this paper is to present a novel approach—a fuzzy logic-based system—that Agile development teams can use to estimate the work effort within the context of SAFe. First, the fuzzy logic technique and its basic rules will be introduced. Then, we will explain the development process and how this technique can be applied to estimate the software development effort.

II. Background

2.1. Scaled Agile Framework

The SAFe framework has three main levels that centralize the strategic themes of an organization: portfolio, program, and team [7].

- The SAFe portfolio involves the people and processes necessary to build a solution that enables an organization to meet its strategic objectives.
- The SAFe program involves development teams and other resources devoted to ongoing system development mission. Its specific role is to help people align to a shared mission.
- The SAFe team involves the roles, activities, and processes that Agile development teams use to build and deliver work products.

Agile teams are the foundation of the SAFe because they perform the majority of the implementation effort. In fact, all software development activities take place at the team level. Agile teams are composed of a cross-functional group of developers with the responsibility to define, build, test, and deliver work products within a short iteration timebox (i.e., Agile sprints). Their main function is to deliver results that meet the customer's needs and expectations. To do so, they perform the following activities [8]:

- Decompose work product "features" to define "stories." (Stories are short descriptions of features maintained within the team backlog.)
- Estimate work effort.
- Determine the corresponding technical design to meet customer's needs and expectations.
- Commit to the work effort that they can accomplish within an iteration timebox (i.e., Agile sprint). The duration of an iteration sprint is usually last for 2 weeks.
- Implement the required work.
- Test the expected functionality.
- Deploy work products.

These activities are all-important to an Agile team. However, the activity that proves to be the most difficult, yet most critical to the Agile teams is the estimation of work effort. A proper estimation of work effort enables the team to successfully plan and commit to the amount of work that the team believes it can accomplish during the program iterationplanning cycle.

2.2. User Story and Story Points

In the context of Agile project management, the team backlog represents all potential work items of the team. These work items may include user stories, technical items, and non-functional requirements. Items presented in team's backlog must be estimated to reflect the level of effort that the team believes is required to complete them according to the team definition of "done" and satisfy the acceptance criteria. The estimation of effort is critical because it provides guidance to the team on how to plan their workload according to the priorities and business values of the enterprise. To successfully plan the work, the Agile team must consider two important factors:

1. Team capacity to deliver work for the next program increment
2. Team velocity, which indicates how fast the team can complete items in the backlog

Story points are used in Agile project management and development to estimate the level of effort [9]. They represent the metric used to express the level of effort required to implement and deliver stories of the team backlog [10]. By definition, story points do not correlate directly to the actual hours; in fact, there is no linear relationship between story points and hours required to complete the work. Therefore, it allows the Agile team to think abstractly about the effort required to complete the work. When the Agile team employs story points to estimate the development effort, the estimation must consider all three main factors conjointly: the complexity of the story, the size of the work, and the uncertainty and risk. It was observed that the difficulty to estimate work effort with story points increases significantly when the team must take into account conjointly all three main factors.

In SAFe, estimation of effort is usually performed relative to a common starting baseline. Therefore, the team must establish a baseline story, from which all subsequent stories will be estimated relatively to it. A baseline story provides the same starting point for all team members and is designated as one story point [7]. A baseline story describes a basic functionality, which could be as simple as "the effort to write a batch command file or produce a test case".

Finally, estimation of effort must involve all members of the development team, because each team member brings a different perspective of the work required to perform and deliver the story. Initially, the team establishes a sequence of numbers that will be used to estimate the effort. The modified Fibonacci format is very popular and is generally used by Agile teams: 0.5, 1, 2, 3, 5, 8, 13, 20…. The method commonly used to estimate the effort with story points is called "Planning Poker," which is based on consensus of all team members, and uses Poker cards with assigned numbers representing values (story points). To start a planning

session, the team reviews a story from the backlog and discusses it briefly. Following the discussion, team members select the card with the value that represents their estimate, and then all cards are revealed. The card numbers revealed represent the story points each team member believes should be assigned to implement and deliver this story and was estimated relatively to the base user story. If every member estimation is the same, then the team accepts this estimation and moves on to the next item in the backlog. If not, team members share and discuss their rationale, and then repeat the Planning Poker estimation process until they reach a consensus.

2.3. Fuzzy Logic

Fuzzy logic represents an artificial intelligence (AI) technique that could be used to simulate human reasoning. As opposed to the usual "true" or "false" Boolean logic (1 or 0), fuzzy logic represents an approach to computing based on the "degree of true". This type of logic manipulates values more than just a simple true or false. Furthermore, when linguistic variables are used, they can be represented by the degree of *truthfulness* or *falsehood*. For this reason, fuzzy logic can handle complex control problems at low computational cost and still preserve the subtle aspect of human thinking. In fact, fuzzy logic control is not just another control method—it represents a quality behavior generator. Compared to other intelligent techniques, fuzzy rule-based systems are easy to design and implement because they require only the usage of a rule-based set extracted from a knowledge database [11]. A fuzzy rule-based expert system contains fuzzy rules defined by its knowledge database. It derives conclusions or decisions from user inputs and the fuzzy reasoning process. The rule-based expert system uses very descriptive linguistic variable terms (e.g., "*low", "moderate", "high"*) to deal with input and output data like a human operator would. Moreover, the knowledge database can grow incrementally, so the performance of the system improves while it is being developed.

The process of developing a classical expert system is illustrated in Figure 1. Engineers establish a dialogue or interview with a subject matter expert (SME). The goal is to gather experts' knowledge. Then, this knowledge is encoded to construct the rule-based database. The SME reviews and evaluates the rule-based database to provide any necessary adjustments. This process goes through many iterations until the knowledge database is completed.

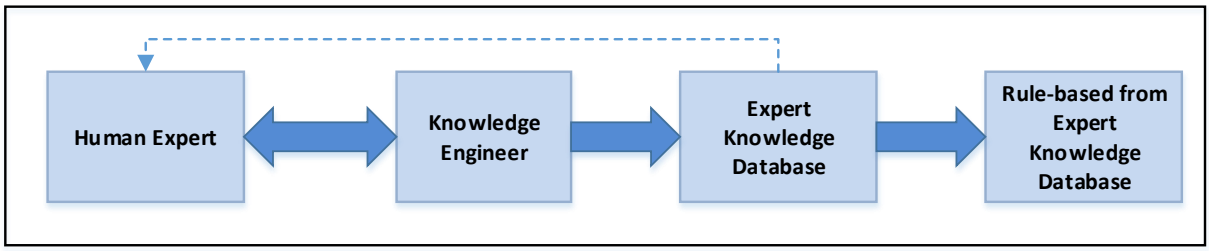

Figure 1. Development of an Expert System

2.4. Membership Functions

Membership functions are a central component to fuzzy logic-based systems. A membership function can be defined as a function that specifies the degree to which a given input belongs to the set. Fuzzy logic was formulated by LoftiZadeh in 1965 [12] as a computation paradigm based on how humans think. In mathematics, fuzzy logic represents the way to process elements that are partially in a set. Generally, a set represents a collection of elements that share common characteristics, for example, a set of "amount of work." The amount of work can be defined as "high" when the number of LOC to produce is greater than 10.000. This set can be graphically represented as in Figure 2. The main characteristic of the crisp set is that an element is either a member or not a member of the set. It does not make any distinction when the number of LOC is 10.000 or 20.000—both situations represent "high" level of effort. On the other hand, 9990 is "low" even though it is only 10 LOC less than the cutoff length. This mathematical function works well for binary operations, but not for real-world situations.

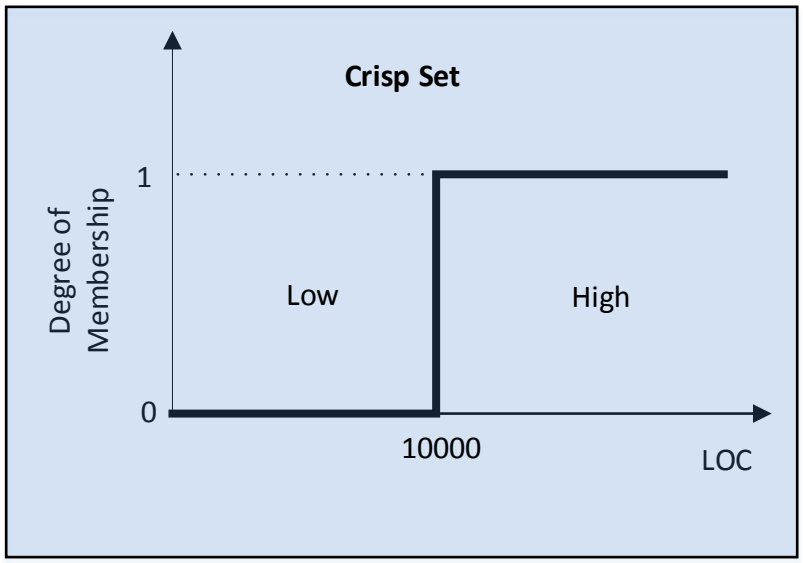

Figure 2. Amount of Work - Crisp Set

Compared to the classical set, a fuzzy set allows members to partially belong to the set. This concept is graphically presented in Figure 3, which shows that there is no clear cut between the "low," "moderate," and "high" categories.The transition between these categories is smooth. In order to use the computer to process linguistic variables, such as "low" or "high," a crisp input, the number of LOC can be converted to fuzzy outputs using a simple membership function similar to the one shown in that figure.

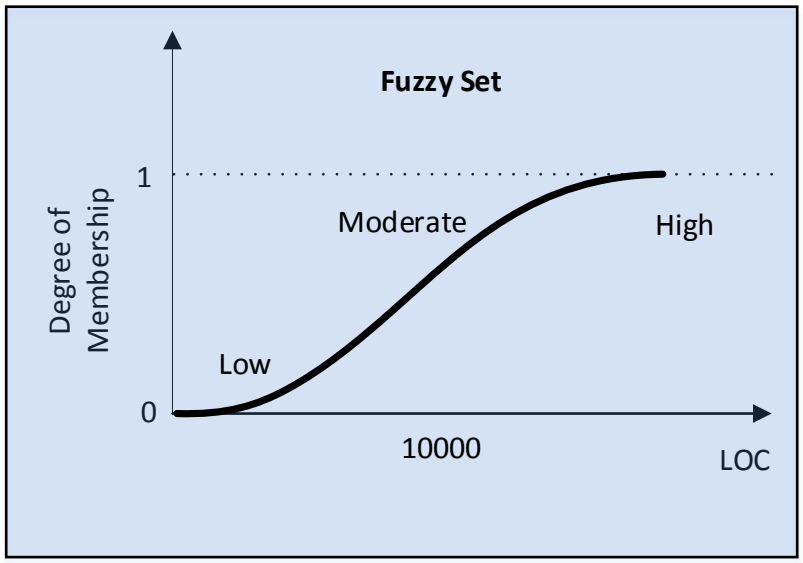

Figure 3. Amount of Work - Fuzzy Set

To summarize, fuzzy logic can be used to handle the concept of partial truth, as opposed to the conventional Boolean logic where an event must be either true or false. As its name suggests, fuzzy logic represents the logic underlying modes of reasoning that are approximate, rather than exact. It was derived from the fact that human reasoning, especially common sense reasoning, is approximate in nature.

III. Fuzzy Inference System

A fuzzy inference system is a process of mapping several crisp inputs into a system to produce a crisp output by using fuzzy ruled-based sets. The basic structure of an inference fuzzy system consists of three main conceptual elements:

1. A rule-based set, which contains a selection of fuzzy rules
2. The definition of the membership functions used in the fuzzy rules

3. A reasoning mechanism, which performs the inference procedure to derive a reasonable conclusion and decision

The concept behind fuzzy reasoning can be represented by fuzzy logic control (FLC), which was derivedfrom control theory and is based on mathematical models of the open-loop control process. FLC has been successfully applied to several practical applications, some as simple as room temperature control and washing machine cycles or some as complex as power systems or nuclear reactors. The components of FLC are illustrated in Figure 4.

The three components of an FLC are:

1. ***Fuzzification*** of crisp input parameters, which is a process that consists of converting crisp input parameters into suitable linguistic values represented by a set of fuzzy logic rules. The fuzzy rules contained in the fuzzy logic set were constructed from the knowledge database. The database provides definitions used to define linguistic control rules and data. The rules characterize the control logic of the domain experts.

2. The ***fuzzy inference engine*** represents the FLC processor and simulates human decision-making based on an established *Fuzzy Rule Base*.

3. ***Defuzzification*** is the process of reconverting fuzzy linguistic parameters back to crisp parameters required for the control process.

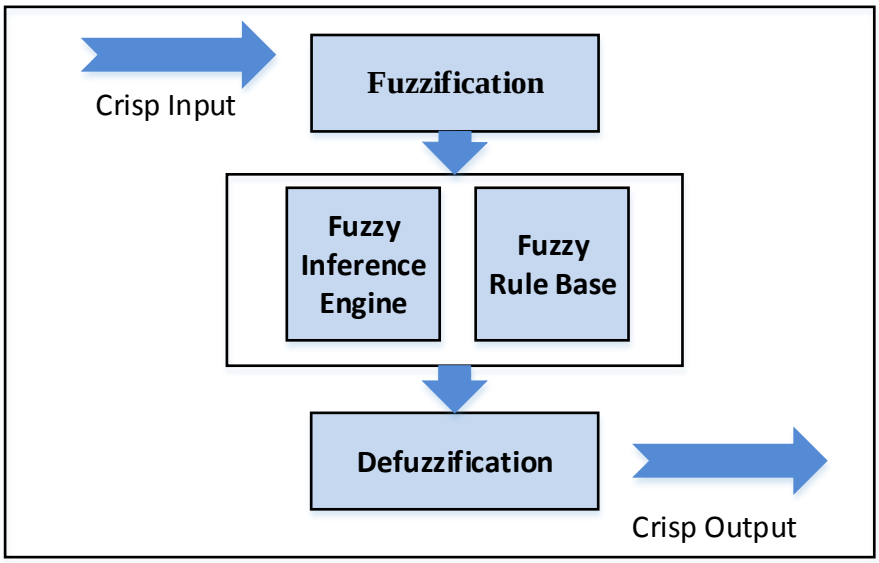

Figure 4. Fuzzy Logic Controller

There are three main types of FLC systems: Mamdani [13], Sugeno[14], and Tsukamoto[15]. In this paper, we will use the Mamdani fuzzy inference system to implement the AI rules for the software estimation effort.

The Mamdani method extends the fuzzy controller for use with intervals and linguistic values as inputs. For example, instead of expressing an input like "Effort to write a batch command file is 1 hour", an equivalent fuzzy logic rule would be "Effort to write a batch command file is low". The latter expression corresponds better to human reasoning. The Mamdani FLC system refers to a fuzzy controller with N inputs X1, X2, …Xn and one single Y output, with a fuzzy logic rule like "If X1 is A1 or X2 is B2 then Y is C1."

To illustrate how a Mamdani FLC is implemented, we examined a typical problem with two inputs (X = Complexity, Y = Uncertainty), one output (Z = development effort), and a set of three rules. In this example, we used membership functions that were either trapezoidal or triangular shape. Many other shapes, such as gauss or bell shape, have been used by different applications [16]. Normally, a specific shape is chosen based on the application and availability of data.

For this example, the following set of rules was considered:

- If the complexity is high and the uncertainty is high, then the development effort is high.
- If the complexity is moderate or the uncertainty is moderate, then the development effort is moderate.
- If the complexity is high, then development effort is high.

"X" is the fuzzy set of "complexity", "Y" is the fuzzy set of "uncertainty", and "Z" is the fuzzy set of "development effort".

Additionally, on a scale of 0 to 10, assuming that the complexity X is equal to 4 and the uncertainty Y is equal to 8. The output Z (e.g. development effort) is to be computed.

Step 1: Fuzzification

Fuzzification consists of projecting the crisp input value (X) to the corresponding fuzzy sets, as illustrated in Figure 5. The projection of a crisp input "X" to a fuzzy variable "A" is mathematically expressed by $\mu_{(X = A)}$. Thus, the input complexity X = 4 projected to the fuzzy variables of "low" with a value of 0.5, "moderate" with a value of 0.4, and "high" with a value of 0.0. The values of 0.5, 0.4, and 0.0 represent the degree of membership that X belongs to the fuzzy variables of "low," "moderate," and "high," respectively. Similarly, the values 0.1 and 0.7 represent the degree of membership that uncertainty Y = 7 belongs to the fuzzy variables of "low" and "high," respectively.

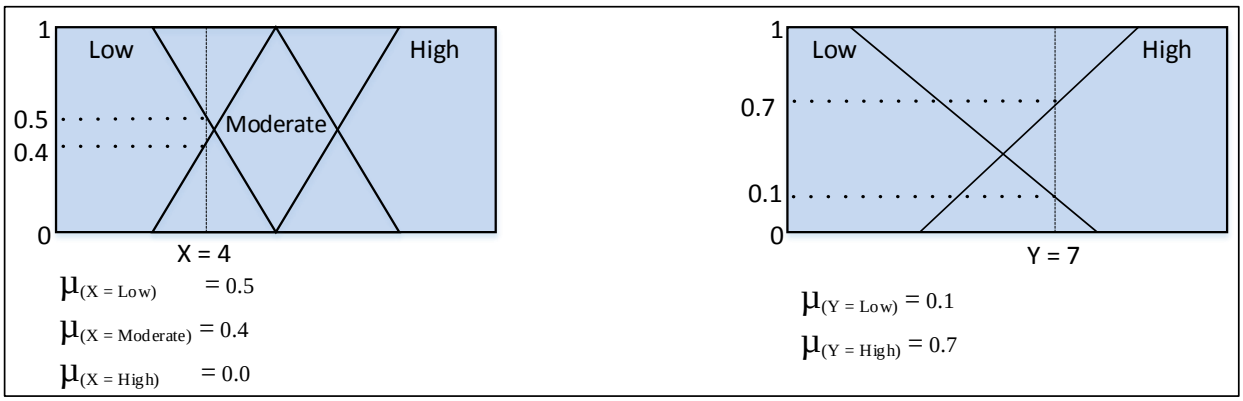

Figure 5. Fuzzification

Step 2: Rules evaluation

In this step, all three rules are evaluated for each fuzzy set of Complexity, Uncertainty, and Development effort, as illustrated in Figure 6.

The conditional clauses in the rule-based set were mathematically computed as follows:

"X ***OR*** Y = **MAX** (X, Y)" and,

"X ***AND*** Y = **MIN** (X, Y)"

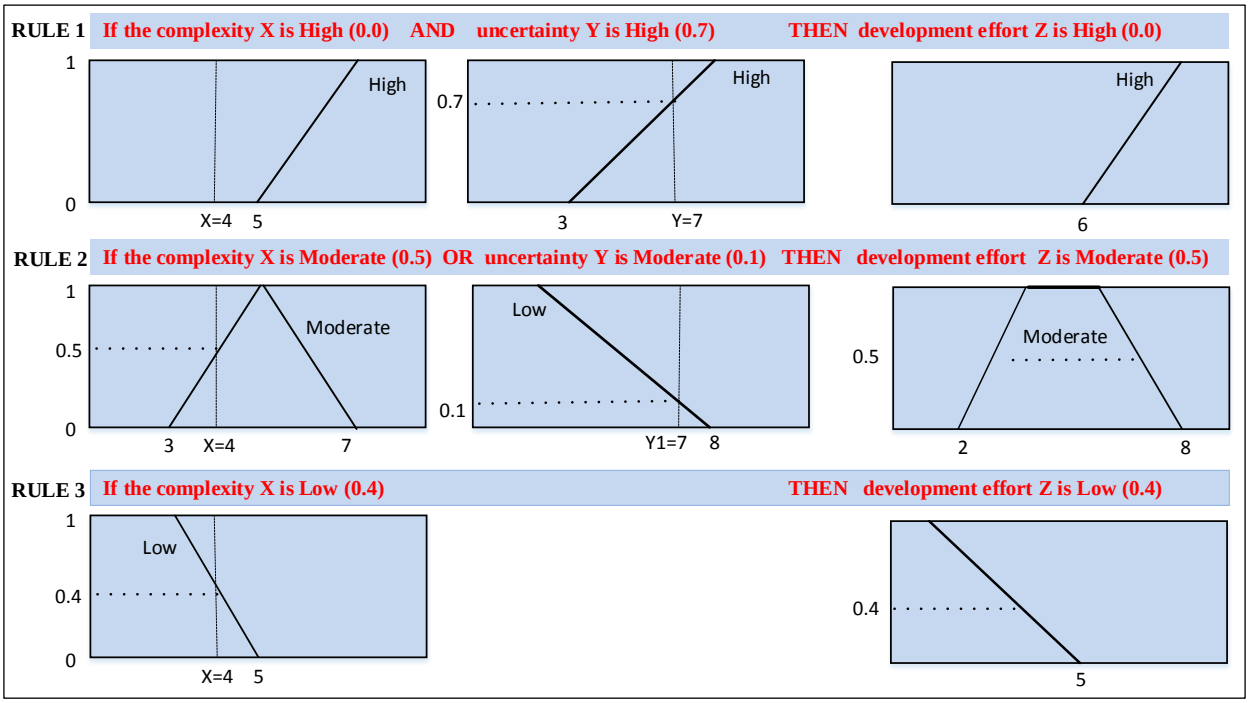

Figure 6. Rules Evaluation

Step 3: Evaluation of the rule output

In this step, all of the three outputs are aggregated. As illustrated in Figure 7, the aggregation is the process of combining the membership functions of all rules (low, moderate, and high) previously clipped or scaled into a single fuzzy set.

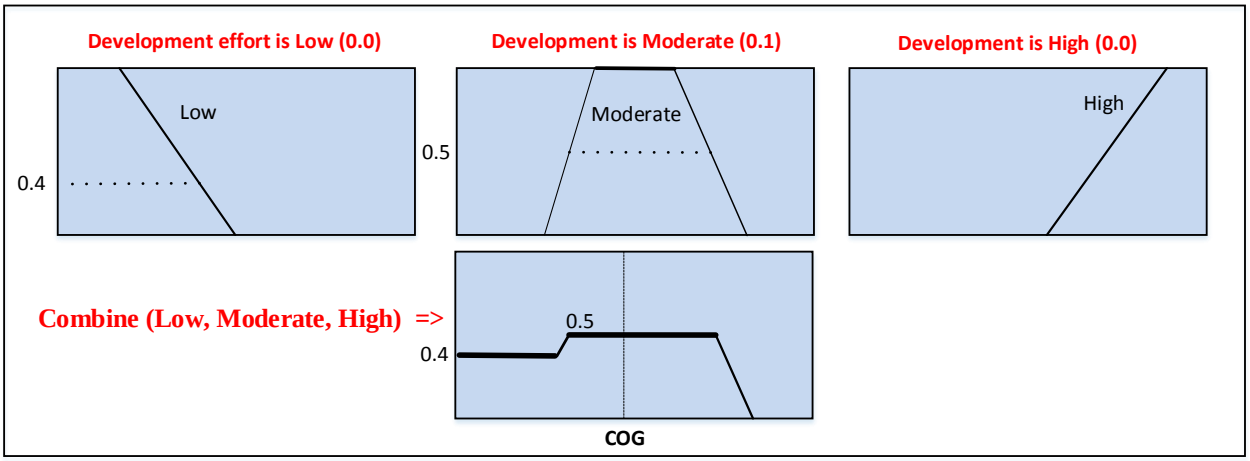

Figure 7. Evaluation of the output

Step 4: Defuzzification

Fuzziness was used to evaluate the rule base. To be useful in a modeling system, the final result must be converted back to a crisp output. This process is known as defuzzification. There are several defuzzification methods, such as central sums, mean of maxima, and left-right maxima, but the most popular is the *centroid* technique. This method determines the point where a vertical line will divide the aggregate set into two equal masses. Mathematically, this point represents the center of gravity (COG) and can be expressed as:

$$COG = \frac{\int_a^b \mu\,(x)\,xdx}{\int_a^b \mu(x)dx}$$

As seen in the example above, a Mamdani FLC represents a simple method to evaluate fuzzy logic rules. Although this method is simple, it remains efficient because it uses a rule-based set in linguistic terms, which are equivalent to human thinking and decision-making.

IV. Use Case

To facilitate the discussion, we will illustrate the concept with a use case: An Agile team responsible for implementing work pertaining to the modeling and simulation of aircraft sensors.

Table 2 represents a subset of an actual backlog of stories from a software development team responsible for the development and implementation of aircraft sensors simulation software. These stories represented actual development work that the team was required to estimate. Each story in this backlog had a different level of complexity. The amount of work required to complete the implementation of each story varied from one story to another story. Finally, the uncertainty of each story represented the potential risk and roadblock that the team must considered during the development cycle.

Table 1. Subset of backlog - Aircraft Sensors simulation software

Story backlog	
1-	Radar – Remove SAR/ISAR mode
2-	Configure Pre-Solic radar to use Max-M-Eyes build
3-	Radar – Add weather mode
4-	Radar buffer saved to shared memory
5-	Upload radar state PDU
6-	Add an operator “Manual track while scan” command

V. Implementation

For each story, the Agile team must estimate the level of effort required, with story point, to complete and deliver it. To estimate the effort, each team member must consider three main factorsconjointly: complexity, size of work, and uncertainty or risk. While the concept itself is simple, it proves tonot be an easy task. For

example, a developer can easily determine how complex a specific task can be or how much uncertainty will be present. However, when attempting to estimate the effort by considering all three factors conjointly, the task of estimation is not necessarily simple.

In this use case, we will demonstrate that while the estimation of effort must consider all three main factorsconjointly, each factor can be estimated separately, and then the fuzzy inference process can be used to combine all these factors to produce the final estimation. To do so, the agile team is required to perform the following two main tasks:

1. Define the fuzzy variables

The fuzzy linguistic variables are defined as follows:

a. Uncertainty – Likert scale from 0 to 20

Low: Negative slope linear function [0-20] and High: Positive slope linear function [0-20]

b. Complexity – Likert scale from 0 to 20

Low [0-8], Moderate [6-14], High [12-20]

c. Amount of Works – Likert scale from 0 to 20

Low [0-8], Moderate [6-14], High [12-20]

d. Level of effort in Story point – Likert scale from 0 to 20

Very Low [0-2], Low [0-8], Moderate [5-13], High [12-20] and Very High [18-20]

Figure 8 illustrates the graphical representation of the defined above fuzzy variables. Linear function and triangular shape are used in this application to facilitate the calibration between the two estimation approaches (e.g. Poker planner and Fuzzy Logic).

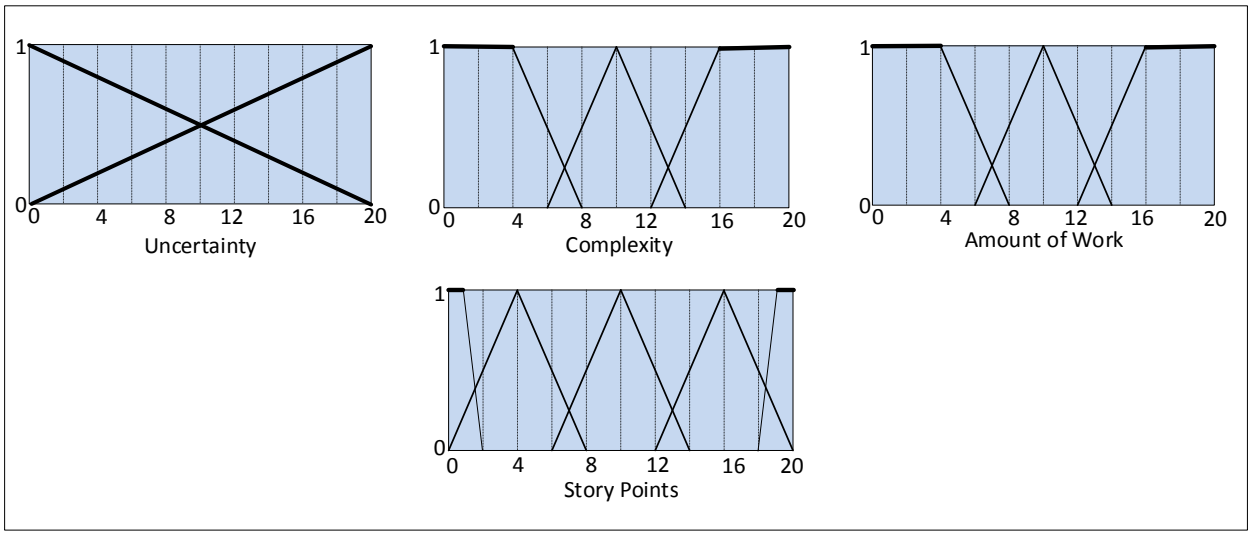

Figure 8. Fuzzy variables definition of Uncertainty (U), Complexity (C), Amount Of work (W) and Story points

2. Construct the expert rule-based database

Typical expert rule is "if the complexity is very low and the uncertainty is low and the amount of work is low, then the level of effort is very low". Eighteen rules (2 x 3 x 3) were constructed using all possible combinations of the input fuzzy variables (e.g., uncertainty, complexity, and amount of work). The resulting rules are presented in Table 1. Dark cells represent a "true" condition. For instance, Rule #1 should read, "if uncertainty is low and complexity is low and amount of work is low, then story points are very low".

Table 2. Fuzzy rules for software development effort
(VL: Very Low, L: Low, M: Moderate, H: High, VH: Very High)

Rule #	Uncertainty		and	Complexity			and	Amount of Work			then	Story Points				
	L	H		L	M	H		L	M	H		VL	L	M	H	VH
1	■			■				■				■				
2	■			■					■				■			
3	■			■						■			■			
4	■				■			■					■			
5	■				■				■				■			
6	■				■					■				■		
7	■					■		■						■		
8	■					■			■					■		
9	■					■				■				■		
10		■		■				■						■		
11		■		■					■					■		
12		■		■						■				■		
13		■			■			■						■		
14		■			■				■						■	
15		■			■					■					■	
16		■				■		■							■	
17		■				■			■						■	
18		■				■				■						■

The fuzzy set as illustrated in Figure 8 and the rules in Table 2 were implemented using the fuzzy inference system (FIS) tool available in the MATLAB® application.
Finally, the team was asked to use the modified Fibonacci sequence to estimate each story using the Planning Pokermethod. With this method, they must consider the uncertainty, complexity, and amount of work for each story**conjointly**. Once the Planning Pokerestimation task is completed, they team was asked to estimate the uncertainty, complexity, and amount of work for each story**sequentially**using the same modified Fibonacci sequence. With this novel approach, the overall estimation of effort in story points can be derived using the fuzzy set and the rules, as defined previously.
Figure 9 illustrates an example of the fuzzy rules evaluation for the story #2 described in the table 2.

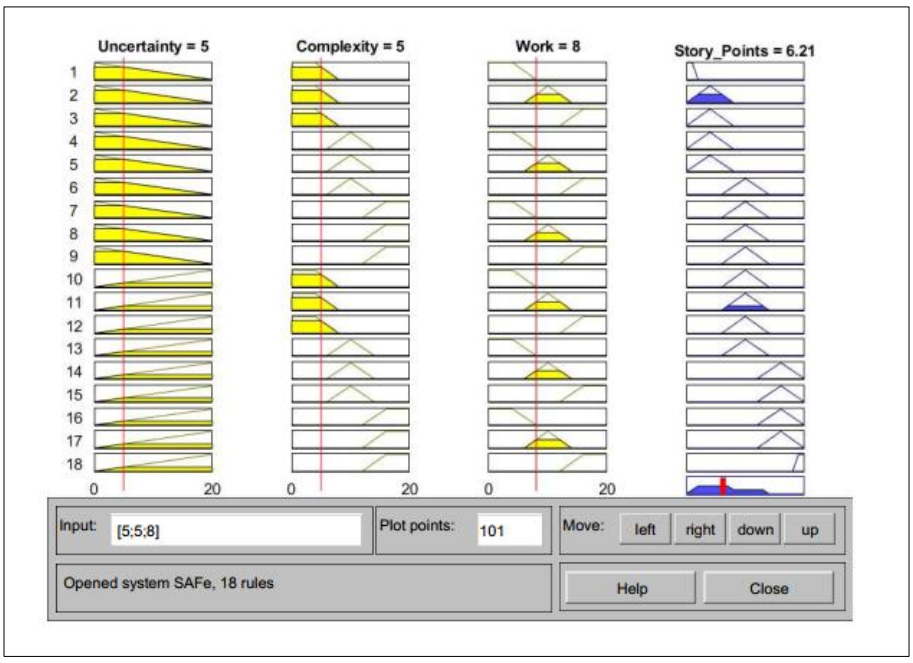

Figure 9. Fuzzy rules evaluation for story #2 (U=5, C=5, W=8)

VI. Results

Table 3 compares the estimation effort for both approaches. The Planning Pokercolumn (SP0) represents the effort estimated in story points using the Planning Poker method. The Fuzzy Logic column (SP1) represents the effort estimated in story points using the proposed fuzzy logic-based method. Column U, C, and W represent the estimation of the uncertainty, complexity, and amount of work, respectively, for each story.

Table 3. Estimation of development effort in Story point for Poker planner and Fuzzy logic methods

Story	Poker Planner	Fuzzy Logic			
	SP0	SP1	U	C	W
1- Radar – Remove SAR/ISAR mode	3	3	1	1	5
2- Configure Pre-Solic radar to use Max-M-Eyes build	8	8	5	5	8
3- Radar – Add weather mode	5	5	1	8	5
4- Radar buffer saved to shared memory	8	8	3	5	8
5- Upload radar state PDU	3	3	1	3	3
6- Add an operator "Manual track while scan" command	20	15	5	15	20

The result was computed using the COG of the aggregation of the story point set. For this story, the computed estimation of effort is 6.21 story points. To be consistent with the Planning Pokermethod, the result is rounded-up to the next integer value of the modified Fibonacci sequence, which is 8.

We can obverse from the obtained results that the estimations compared very well between both methods, with the exception of the story #6 in the Table 3. The discrepancy can be explained by the fact that the team may have overestimated when it used the Poker planner method to estimate this story, as this method required the team to take into account three parameters (e.g. uncertainty, complexity and amount of work) conjointly.

VII. Conclusion

This paper introduced a novel approach in the process of estimating work effort in software development. Because we were dealing with an assessment problem that required an underlying mode of reasoning that is approximate rather than exact, the introduction of fuzzy logic in the estimation of effort was necessary. Additionally, the proposed approach used linguistic variables that are much more correlated with the normal human mode of reasoning.

The development team members believed that it was easier for them to estimate the effort by sequentially considering one parameter at a time instead of three parameters conjointly. Therefore, a software effort estimation model incorporating fuzzy logic can help software development teams to overcome the issue of estimating multiple attributes conjointly (e.g., complexity, uncertainty, and amount of work), where the definition of each attribute that required human judgment is vague.

References

[1]. Sinhal, A., Verma, B. (2013). Software Development effort: A Review. International Journal of Advanced Research in Computer Science and Software Engineering. Vol 3 (6): 1120-1135.

[2]. Avasthy, R., Batra, B., Kundra, H. (2016). A hybrid parametric model for enrichment of software effort estimation. *International Journal of Computer Science and Information Security*. Vol 14 (10).

[3]. Boemh, B. (1981). Software Engineering Economics. Englewood Cliffs, NJ, Prentice Hall.

[4]. Chawla, R., Ahlawat, D. (2015). Software development effort using fuzzy logic framework – An Implementation. *International Journal on Advanced Computer Theory and Engineering*. Vol 4 (4).

[5]. Kamal, S., Kamal, Z., Khan, S., Nasil, J.A. (2013). A fuzzy logic-based software estimation model. International *Journal of Software Engineering and Its Applications*. Vol 7, No 2.

[6]. Su, M. T., Ling, T. C., Phang, K. K., Liew, C. H., Man, P. W. (2007). Enhanced software development effort and cost estimation using fuzzy logic model. *Malaysian Journal of Computer Science*. Vol 20(2).

[7]. Leffingwell, D. (2018). SAFe 4.5 Reference Guide. Scaled Agile Framework for Lean Enterprises. Second Ed. Publisher: Addison-Wesley Professional.

[8]. Leffingwell, D., Yakima, A., Knaster, R., Jemilo, D., Oren, I. (2017). Scaled Agile Framework for lean software and system engineering. Scaled Agile, Inc.

[9]. Georgsson, A. (2011). Introducing Story Points and User Stories to Perform Estimations in a Software Development Organization. Master Thesis, UMEÅ University, Department of Computing Science.

[10]. Hamouda, A. E. (2014). Using Agile Story points as an estimation technique in CMMI organizations. Agile Conference 2014.

[11]. Magdalena, L. (2015). Fuzzy Ruled-Base Systems. Springer Handbook of Computational Intelligence. Chapter 13. 203-218.

[12]. Zadeh L. A. (1965). Fuzzy Sets. Intl J. Information and Control. Vol 8. Issue 3: 338-353.

[13]. Mamdani, E. H., Assilian, S. (1975). An experiment in linguistic synthesis with a fuzzy logiccontroller, Int. J. Man-machine Studies 7: 1–13.

[14]. Sugeno, M., Kang, G. T. (1988). Structure identification of fuzzymodels. Fuzzy Sets and Systems 28: 15-33.

[15]. Tsukamoto, Y. (1979). An approach to fuzzy reasoning method. In Madan M. Gupta, Rammohan K. Ragade, and Ronald R. Yager, editors, Advances in Fuzzy Set Theory and Applications, North-Holland, Amsterdam: 1979, pp.137-149.

[16]. Klir, G. J., Yuan, B. (1995). Fuzzy Set and Fuzzy Logic. Prentice Hall PTR.