**A fuzzy logic-based system for enhancing scrum methodology**

Mihailo Stupar\*1, Pavle Milošević1, Bratislav Petrović1

1University of Belgrade, Faculty of Organizational Sciences

\*Corresponding author, e-mail:stupar.mih@gmail.com

**Abstract:** In this paper, we propose a decision support system for enhancing scrum methodology based on fuzzy logic. The proposed system consists of three main components: a fuzzy inference system, an aggregation operator and a feedback function. In the basic scrum, requirements which describe a certain task do not have a clear interpretation. Also, the basic model does not take into account the experience of the developers nor the logical dependencies of input variables. Fuzzy inference systems are particularly useful for this purpose, because it incorporates logic in inference process and inputs are presented using linguistic quantifiers. Aggregation function is used to aggregate predictions in a single value that uniquely represent a specific task, while a feedback is employed to adjust an input variable to improve system performance. Furthermore, the proposed system is simulated with randomly generated inputs in order to analyze its behavior. The predictions of the system are more accurate and with s smaller deviation in the final iterations.

**Keywords**: scrum methodology, decision support system, fuzzy logic, fuzzy inference system, aggregation operator, feedback function

1. introduction

Scrum as an agile methodology is very popular within the software and product development. Scrum is ideal for projects with aggressive deadlines, complex requirements, and a significant degree of uniqueness (Almseidin et al. 2015). Although development teams are nowadays often distributed all over the world, this methodology is still used to run projects (Sutherland et al. 2007).In scrum, projects move forward through the series of iterations called sprints, and each sprint is typically two to four weeks long.

At the start of each sprint, a whole team of developers has a meeting (sprint planning meeting) where they decide which tasks should be included in the following sprint. They all have a list of features/tasks (made by experts) that should be implemented by the end of the project. These features/requirements are collected in the Product Backlog (Duechting et al. 2007). Each task is weighted with some number of points. In scrum, these points are called story points. A number of story points that are contained within a sprint is well known, but the number of tasks included in the sprint depends on the developers’ estimation on how difficult each task is. In the sprint planning meeting, they negotiate the value of the tasks and how many tasks from the Product Backlog will be included. They all give a value (weight) to each task independently. Later, their manager (scrum master) looks at the values for the tasks and decides (with agreement of the whole team) how many story points each task is actually worth .

Although the scrum methodology is successfully applied in various fields, there are certain issues that should be addressed with regards to the story point estimation. Developers often think that the number of story points for each task corresponds to a number of hours/days needed to complete this task. The first assumption about the concept of story points is that they are just points, relative values, as opposed to absolute values. Anyone looking at the values could be able to compare tasks based on story points without expressing these values in terms of how many hours/days would be needed to complete these tasks. Another issue is related to the experience of developers. Some developers are experienced and tasks are often too easy for them, while others might not be that good at estimation. For example, they often give a task a smaller value than it actually has.

A scrum master needs to think about all these parameters when deciding on the final value of story points for each task. Since fuzzy logic provides a useful tool to deal with problems with the phenomena which are imprecise and vague (Lin et al. 2006), it is ideal to be used for this purpose. Our proposed system based on fuzzy logic could enhance efficiency in scrum planning phase. The system accepts developers’ task estimation and the scrum master’s knowledge as inputs, and the output is weight (story point value) of the observed task. Also, the system should become more stable over time, and achieve a more precise prediction.

This paper is structured as follows. In Section 2 we have described fuzzy logic and FIS in general, and how it was previously used in scrum methodology. In Section 3 we have introduced our fuzzy logic-based system, an aggregation operator and a feedback function that should improve accuracy of the system. Further, we have created a simulation that shows the way our system works, which is explained in Section 4. Finally, Section 5 contains the conclusion and ideas for further research.

2. FUZZY LOGIC

Fuzzy logic is a generalization of classical logic in a sense that it can process all values from the interval [0,1] (Zadeh, 1965, Zadeh, 2008). It may be seen as an attempt to formalize human capability to reason and make rational decisions in an environment of imprecision and uncertainty. In fuzzy logic, operators of intersection and union are realized using different functions that are referred to as t-norms and T-conorms (or S-norms), respectively. Min operator is the standard choice for fuzzy intersection, while algebraic product and Lukasiewicz norm are also frequently used (Ross, 2010). Max operator, probabilistic sum and Lukasiewicz t-conorm are its corresponding as T-conorms. The most common negation operator in fuzzy logic is a standard negation  .

2.1. Fuzzy inference system

A fuzzy inference system (FIS) is a system based on fuzzy logic that utilizes a set of rules to map inputs to outputs. It is the most commonly seen fuzzy methodology. Two most important types of the FIS are Mamdani (Mamdani, 1977) and Takagi–Sugeno (Takagi and Sugeno, 1985). In the Mamdani system, both input and output are fuzzy sets, wich is easier to interpret. In the Takagi–Sugeno system, inputs are fuzzy sets, while the output is a linear combination of its inputs. This type of FIS is a more accurate one, but also more computationally expensive. The fuzzy inference systems are particularly useful in problems where inputs are expressed as linguistic expressions. The FIS is based on IF-THEN rules, fuzzy conditional statements that incorporate logic. They are a collection of linguistic statements that describe how the FIS should make a decision. Just like an algebraic variable takes numbers as values, a linguistic variable takes words or sentences as values. Their IF part is a logical condition that should be fulfilled in order for the THEN part to be realized. IF-THEN rules are commonly specified by a field expert.

The fuzzy inference process consists of three main steps: fuzzification, rule evaluation and defuzzification. The first step in fuzzy inference is to convert linguistic expressions to values on the unit interval [0,1] using a set of input membership functions. Further, fuzzy rules are evaluated using appropriate operators for t-norm, t-conorm and negation. Results of all IF-THEN rules are aggregated into a single fuzzy set. Finally, defuzzification is applied. It is a process that converts a fuzzy set or a fuzzy number into a crisp value, representing the final output.

2.2. Fuzzy logic in the scrum methodology

Fuzzy logic proved to be particularly useful for building an expert system based on logical dependent variables. It is especially suitable when inputs are expressed as linguistic statements. Due to its characteristics, fuzzy logic is widely used as a tool for enhancing scrum methodology.

Sedehi and Martano (2012) introduced a new model based on fuzzy logic that is used to evaluate and monitor scrum projects. Their model has linguistic variables as inputs while output is the level of success of the (part of) scrum project. Kurian (2011) created Sugeno based fuzzy model that should determine and react to changes in an agile process, such as product/software development process. Similar approach can be found in Lin et al. (2006), where they used Mamdani based model to register changes in the enterprise world.

3. design of fuzzy expert system

The goal of this paper is to build a fuzzy expert system that can be a valuable support or even a complete replacement for an expert (scrum master) during the sprint planning phase. The rules that scrum master follows in the decision-making process can be easily expressed linguistically, so the fuzzy logic system is suitable when dealing with this kind of problem (Lin et al. 2006). The proposed system consists of three components: fuzzy inference system (FIS), aggregation function and feedback function. Experts’ knowledge should be used only in the first iteration in order to set up initial parameters.

3.1. Fuzzy inference system

Our fuzzy logic system has three input variables that describe the experience of developers, their estimation skills and their rating for the observed task. The output is a value (story point value) of the observed task.

In case of the experience (EXP) input variable, every developer has a specific status in the company, which is based on their years of experience. Instead of using four groups (junior, intermediate, senior, and expert), which is common in literature (Orlowskyet al. 2006), we decided to exclude the expert group from our system. Experts are more often leaders and they more often define tasks that need to be done instead of implementing them. So, our EXP variable consists of three membership functions, each one representing one level of experience. Every membership function in this paper is a PI - shaped function defined by four parameters. These parameters are specified by an expert using the fuzzy visualization tool in MATLAB (Sivanandam et al. 2007). Parameters for every membership function for EXP variable are shown in Table 1. Values represent the years of experience and they are comma separated.

Table 1: Membership function parameters for EXP variable

|  |  |
| --- | --- |
| **Variable** | **Values of parameters** |
| Junior | 0,0, 1.183, 2.64 |
| Intermediate | 1.52, 2.69, 3.706, 5.04 |
| Senior | 3.579, 5.18, 6.24, 8.16 |

The second input variable describes an accuracy/quality of developer’s estimation on how much the specific task is complex (EST). This variable may be of great importance for the inference process, since some developers constantly miscalculate the task complexity. It is linguistic in nature and represented by three membership functions: Overestimated, Well Estimated, and Underestimated, while the assessments are on the interval . Membership functions’ parameters are shown in Table 2.

Table 2: Membership function parameters for EST variable

|  |  |
| --- | --- |
| **Variable** | **Values of parameters** |
| Overestimated | -1.62, 0.779, 1.85, 3.452 |
| Well Estimated | 2.071, 3.26, 4.61, 5.99 |
| Underestimated | 4.516, 6.29, 7.3, 9.7 |

Bearing in mind that the accuracy of each developer’s estimation may vary over time, EST values will be adjusted after each iteration. The proposed system has a sort of feedback at the end of each iteration, which should improve the validity of EST variable. A detailed explanation of feedback will be given later in the text.

Finally, the third input variable is a weight/rating that the developer gives to each task (WEI). Although human expression can be interpreted with 9 linguistic terms (Lin et al. 2006), in agreement with the scrum master, we decided to represent this variable with three values (Easy, Medium, Heavy) while the expert’s assessments are on the interval . This task weight representation is intuitive, close to human perception and based on natural language. All these variables are PI-shaped as well, and their parameters are represented in Table 3.

Table 3: Membership function parameters for WEI variable

|  |  |
| --- | --- |
| **Variable** | **Values of parameters** |
| Easy | -1.16, 0.76, 1.75, 3.53 |
| Medium | 1.94, 3.34, 4.5, 5.93 |
| Heavy | 4.67, 6.12, 7.24, 9.16 |

The output of the fuzzy system is a numeric value that represents the weight of the task in terms of story points, and all values are split into four groups: Easy, Medium, Complex, and Very Complex. Although the valid story point values in basic scrum methodology belong to modified Fibonacci array (1/2, 1, 2, 3, 5, 8…) (Downey and Sutherland, 2013), we decided to represent the output using a uniform distribution. If we used Fibonacci array, we would have a very small difference between Easy and Medium group and a large gap between Complex and Very Complex groups (see Table 4). As a result, the tasks would be weighted as a Complex or Very Complex with high probability.

Table 4: Story point distribution using Fibonacci array

|  |  |
| --- | --- |
| **Group name** | **Values** |
| Easy | 1/2, 1, 2 |
| Medium | 3, 5, 8 |
| Complex | 13, 21, 34 |
| Very Complex | 55, 89 |

It is obvious that this distribution cannot give useful results because the fuzzy system threats all membership functions equally and Very Complex membership function takes almost half of the full output interval. So, instead of using these values, the parameters of output membership functions are rescaled (Table 5).

Table 5: Output membership functions’ parameters

|  |  |
| --- | --- |
| **Group name** | **Values of parameters** |
| Easy | 0, 0, 2.69, 15.56 |
| Medium | 3.53, 10.92, 16.07, 25.34 |
| Complex | 16.8, 23.96, 30.02, 38.78 |
| Very Complex | 27.95, 39.98, 50, 55 |

By using these inputs and the output, we created a set of rules for our fuzzy system. We follow the scrum master’s thoughts and recommendations during the scrum planning phase. This phase is interactive and this person leads it with their suggestions. These pieces of advice are transformed into the following set of rules:

1. IF (EST = WELL\_EST) and (EXP is not JUNIOR) and (WEI is EASY) THEN (OUT is EASY)
2. IF (EST = OVER\_EST) and (WEI is not EASY) THEN (OUT is EASY)
3. IF (EST = UNDER\_EST) and (EXP is not SENIOR) and (WEI is EASY) THEN (OUT is MEDIUM)
4. IF (EST = OVER\_EST) and (WEI is not EASY) THEN (OUT is MEDIUM)
5. IF (EST = OVER\_EST) and (EXP is not SENIOR) and (WEI is HEAVY) THEN (OUT is COMPLEX)
6. IF (EST = UNDER\_EST) and (EXP is not JUNIOR) and (WEI is HEAVY) THEN (OUT is COMPLEX)
7. IF (EST = WELL\_EST) and (EXP is not JUNIOR) and (WEI is HEAVY) THEN (OUT is V\_COMPLEX)
8. IF (EST = UNDER\_EST) and (WEI is not EASY) THEN (OUT is V\_COMPLEX)

The proposed FIS is Mamdani-type. This type of FIS ensures the necessary transparency of the decision-making process. AND operator is evaluated using *min* function, OR operator using *max* function and in the defuzzification process we used centroid method.

In the Figure 1, it is shown how OUT values change for different EXP and EST values and constant WEI value. If a developer is a good estimator (EST value is close to 4), the OUT value is high (since WEI is high), except in the case of inexperienced developers (EXP < 2). Since inexperienced developers do not yet have sufficient knowledge, they may perceive medium tasks as difficult ones. On the other hand, the developers who usually overestimate their tasks have OUT values smaller than predicted (about 2).

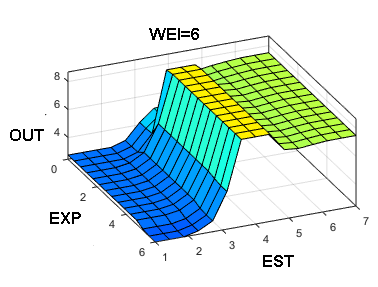


Figure 1: Output values for WEI=6

3.2. Aggregation function

Each developer has its own input vector and the fuzzy system generates a separate output for each task. In other words, for each task we get as many outputs as there are developers in our system. In the second layer, we aim to aggregate these predictions in a single value that uniquely represents a specific task. Since characteristics of each developer are taken into account in FIS, all FIS output values should be treated equally. Therefore, we propose a simple average as the aggregation function.

The scrum master’s insights regarding a certain team and/or a project may be expressed by using various aggregation functions in this layer. For example, if the optimistic estimation is needed, the proper aggregation function is *max* function. For calculating pessimistic estimation, *min* function should be used.

3.3. Feedback

The third and very important layer is feedback function which is represented in the Figure 2. At the end of each iteration/sprint, we can see how difficult each task was and this variable is represented with REAL\_OUT value. Using this REAL\_OUT and ESTIMATED\_OUT values for the ith sprint, we can update EST value for each developer in the (i+1)th sprint. For example, let us assume that the developer D is a good estimator and he states that the task T is a heavy one (WEI). After the sprint, if the task T turned out to be easy in general (REAL\_OUT), the developer D should not be regarded as a good estimator in the next iteration (EST).

The feedback is realized as the quadratic function of the difference between real and estimated difficulty. The function is weighted in order to get a weaker slope of the function. Using this function, we will award or penalize a developer EST value, depending on their good or bad estimation.

Using this set of rules and feedback function, the system should improve itself over time and become stable after a few sprints. Furthermore, we can assess developers’ estimations in order to analyze their tendency to overestimate or underestimate the task over a period of time. At the beginning, we can assume that all developers are good estimators, and by the end of the project, we will see how good they actually are. These (trained) values should be starting points in the next project, so we could expect better results from the start.

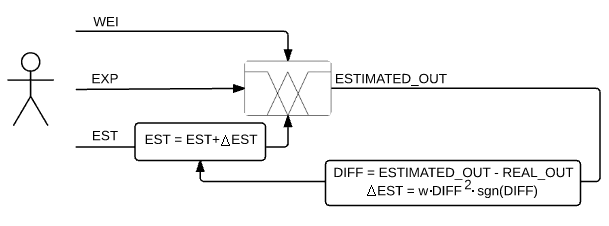


Figure 2: Design of the fuzzy expert system

4. EXPERIMENT

In this section we present the simulation in order to illustrate how the proposed system works. We will simulate the work on the project that consists of two sprints with six tasks. Five experts with various levels of experience participated in the project.

First, the initial values of the estimation accuracy EST and experience EXP variables for each developer are assigned. Further, we assign output values REAL\_OUT for each task in one sprint. REAL\_OUT represents the actual complexity of the task that includes both objective circumstance and human factor. On the other hand, it will also be used as a base for WEI value calculation in this simulation.

We are going to use these output values in order to find out the mean squared error of each task in the estimation process. EST, EXP and REAL\_OUT values are taken from a uniform distribution. Finally, the weight/rating that the developer gives to each task WEI is calculated based on EST and OUT values.

**WEI value calculation**. Values of variable WEI cannot be randomly generated because they depend on characteristic of the tasks (summarized in REAL\_OUT) and the accuracy of developers’ estimations (explained with EST). However, we calculate WEI in such a way that it resembles a developers’ evaluation.

|  |  |
| --- | --- |
|  | (1) |
|  | (2) |

First, in (1) we calculate offset or distance from the center of all EST values. EST values are uniformly disturbed on the interval , so in our case CENTER\_EST is 4. Further, WEI value is calculated using the formula (2). The value of coefficient *c* is obtained through testing and set to 0.35.

**FIS evaluation**. Next, we use EST, EXP and WEI as inputs for our fuzzy inference system in order to get ESTIMATED\_OUT as output. We are going to get as many ESTIMATED\_OUT values for a single task as we have developers on the project. Note that in the first iteration we use INIT\_EST value for EST variable. In further iterations, EST values are going to be calculated with feedback function.

**EST value improvement using feedback**. After each iteration, EST value is updated in accordance with the prediction accuracy. Thus, the adaptability of the system and its robustness to changes are ensured.

|  |  |
| --- | --- |
|  | (3) |
|  | (4) |
|  | (5) |

In the equation (3), we calculate the difference between predicted and real output weights. Later, we use weighted quadratic function (4) to calculate the value which will be added to the EST value from the previous iteration (5). The value of weighting coefficient *w* is set to 0.05.

**The evaluation of the system.** Instead of evaluating our system as a set of iterations (Sedehi and Martano, 2012), we will utilize the difference between the aggregated estimated and actual output (ΔFIN\_OUT) and mean squared error (MSE) for each task as performance measures. We use ESTIMATED\_OUT values from each developer and aggregate them using the average function into ESTIMATED\_OUT\_AVG. This value will be compared to the REAL\_OUT value to obtain the difference between the estimated and actual output. MSE is used as a deviation indicator of individual prediction ESTIMATED\_OUT. Firstly, we calculate mean squared error for each developer separately and then apply average function to them to get a single MSE. This value can show how stable our system is.

Table 6: MSE for Sprint 1

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **TASK** | **ESTIMATED\_OUT\_AVG** | **REAL\_OUT** | **ΔFIN\_OUT** | **MSE** |
| 1 | 3.8620 | 4 | 0.138 | 3.8765 |
| 2 | 5.1015 | 5 | 0.1015 | 8.8885 |
| 3 | 0.8593 | 2 | 1.1407 | 1.3015 |
| 4 | 1.1220 | 3 | 1.878 | 3.5324 |
| 5 | 0.8551 | 2 | 1.1449 | 1.3111 |
| 6 | 1.1198 | 3 | 1.8802 | 3.5687 |
| **AVERAGE** |  |  | **1.0472** | **3.7465** |

Table 7: MSE for Sprint 2

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **TASK** | **ESTIMATED\_OUT\_AVG** | **REAL\_OUT** | **ΔFIN\_OUT** | **MSE** |
| 1 | 3.9489 | 4 | 0.0511 | 1.6600 |
| 2 | 1.2121 | 3 | 1.7879 | 3.2835 |
| 3 | 0.8594 | 2 | 1.1406 | 1.3013 |
| 4 | 1.8630 | 2 | 0.137 | 0.2163 |
| 5 | 4.3738 | 4 | 0.3738 | 3.2725 |
| 6 | 1.1049 | 1 | 0.1049 | 0.1093 |
| **AVERAGE** |  |  | **0.5992** | **1.6405** |

In Tables 6 and 7, the results for two sprints are presented together with ΔFIN\_OUT and MSE. In general, the average accuracy of prediction in Sprint 2 increased compared to Sprint 1, while the MSE value decreased. Based on MSE values, we can see that our system becomes more and more stable over time. In the beginning, even in cases when it produced a good prediction of the task complexity, i.e. for task 2 in Sprint 1, it had high deviation of individual assessments. For Sprint 2, MSE values are significantly lower. Therefore, we can state that besides greater estimation accuracy, the proposed fuzzy logic-base system improves the stability of individual assessments.

5. CONCLUSION AND FUTURE WORK

Fuzzy logic has been widely used to assist decision makers in a number of different domains. In this paper, it is utilized as a basis for building a decision support system to determine the weight of the tasks in agile methodology such as scrum. The system consists of three modules: a fuzzy inference system, an aggregation operator and a feedback function.

We consider that there is no need to use unique, predefined values for estimation in scrum (as Fibonacci series), but that we can efficiently use linguistic variables to achieve the same goal. Using the proposed fuzzy inference systems, we enhance each developer’s story point estimation in accordance with their experience and previous prediction accuracy. The knowledge of the scrum master is transformed to fuzzy rules that are easy to interpret and fully resemble human reasoning. The output variables are further aggregated in a final estimation using a simple average. The feedback is applied to update the variable that represents each developer’s quality estimation in order to increase adaptability to changes and poor assessments. We have simulated the proposed system and showed that it becomes more stable over time and gives more accurate predictions of the tasks.

The proposed system could be more accurate if we take into account additional input variable in the FIS that represents how often requirements are going to change in the sprint. Some tasks/requirements are not fixed during the whole sprint and this may lead to not meeting the deadline. The idea for future work is to incorporate this variable into the proposed system and to add more rules that will treat these tasks differently from the stable ones. Also, we aim to use various aggregation operators in order to model different problem situations.

references

Almseidin, M., Alrfou, K., Alnidami N.,&Tarawneh A. (2015). A Comparative Study of Agile Methods: XP versus SCRUM. *International Journal of Computer Science and Software Engineering,* 4(5), 126-129*.*

Downey, S., & Sutherland, J. (2013). Scrum metrics for hyperproductive teams: how they fly like fighter aircraft. In *System Sciences (HICSS), 2013 46th Hawaii International Conference on*, 4870-4878. IEEE.

Düchting, M., Zimmermann, D., &Nebe, K. (2007). Incorporating user centered requirement engineering into agile software development. In *Human-computer interaction. Interaction design and usability*, 58-67. Berlin: Springer.

Kurian, T. (2011).A Fuzzy Based Approach for Estimating Agility of an Embedded Software Process.Retrieved from http://www.siliconindia.com/events/siliconindia\_events/Global\_Embedded\_conf/Globa\_Embedded\_Conf\_PPT\_final\_tisni.pdf

Lin, C. T., Chiu, H., & Tseng, Y. H. (2006). Agility evaluation using fuzzy logic. *International Journal of Production Economics*, 101(2), 353-368.

Mamdani, E. H. (1977). Application of fuzzy logic to approximate reasoning using linguistic synthesis. *IEEE Transactions on Computers*, 100(12), 1182-1191.

Orłowski, C., Bach-Dąbrowska, I., Kapłański, P., &Wysocki, W. (2014). Hybrid Fuzzy-ontological Project Framework of a Team Work Simulation System. *Procedia Computer Science*, 35, 1175-1184.

Ross, T. J. (2010). *Fuzzy logic with engineering applications (*3rd ed.). Chichester: John Wiley & Sons.

Sedehi, H., &Martano, G. (2012). Metrics to Evaluate & Monitor Agile Based Software Development Projects-A Fuzzy Logic Approach. In *Software Measurement and the 2012 Seventh International Conference on Software Process and Product Measurement (IWSM-MENSURA), 2012 Joint Conference of the 22nd International Workshop on,* 99-105. IEEE.

Sivanandam, S. N., Sumathi, S., &Deepa, S. N. (2007). *Introduction to fuzzy logic using MATLAB* (Vol. 1). Berlin: Springer.

Sutherland, J., Viktorov, A., Blount, J., &Puntikov, N. (2007). Distributed scrum: Agile project management with outsourced development teams. In *System Sciences, 2007. HICSS 2007. 40th Annual Hawaii International Conference on*, 274a-274a. IEEE.

Zadeh, L. A. (1965). Fuzzy sets. *Information and control*, 8(3), 338-353.

Zadeh, L. A. (2008). Is there a need for fuzzy logic?.*Information sciences*, 178(13), 2751-2779.