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

Mihailo Stupar1, Pavle Milošević1, Bratislav Petrović1

1University of Belgrade, Faculty of Organizational Sciences

**Abstract**: In this paper, we propose a decision support system for enhancing scrum methodology based on fuzzy logic. Scrum is a very popular agile methodology within the software and product development. In the basic scrum, requirements which describe a certain task do not have a clear interpretation. Also, the traditional model does not take into account the experience of the developers nor the logical dependencies of input variables. Fuzzy inference is particularly useful for this purpose, because it incorporates logic in inference process and inputs are presented using linguistic quantifiers. The proposed system consists of three main components: a fuzzy inference system, an aggregation operator and a feedback function. The aggregation function is used to aggregate task 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 smaller deviation in the final iterations.

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

**1. Introduction**

Agile methodologies provide a structure and a set of principles for collaborative software development. They were created in the 1990s as a reaction to the deficiency of strictly panned projects, where the top-down approach is assumed. They are focused on activities that directly add values to the product and make a customer more satisfied (Fowler & Highsmith, 2001, Dingsoyr et al., 2012)

In the most of agile development methods, product development is divided into small increments that minimize the amount of up-front planning and design. Each iteration must deliver some meaningful unit and bring benefits to the product itself, with each iteration having its own development phase. Each iteration consists of planning, development, testing and documentation phase (Nerur et al., 2005). The order of these phases is irrelevant and they can be developed in parallel, because the only important thing is to have small pieces of a unit completed upon the end of a single iteration. We can find same phases in classical waterfall methodology, but they need to be executed in an explicit order and the customer’s first encounter with the product is only after the development phase, which is probably somewhere close to the end of project. The main benefits and limitations of agile methods are identified and analyzed in literature (Dyba & Dingsoyr, 2008).

Scrum is a very popular agile methodology 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, i.e. sprint planning meeting, where they decide which tasks should be included in the following sprint. They have a list of features and/or tasks made by experts that should be implemented by the end of the project. These requirements are collected in the Product Backlog (Duechting et al., 2007). Team members estimate the completion time for each task. Here, they use story points as estimation measure (Cho 2008). Each task is weighted with a certain number of points, i.e. story points. A number of story points 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, the manager, i.e. scrum master, decides (with agreement of the whole team) how many story points each task is actually worth based on the weights of each task.

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 relative in nature (Coelho & Basu 2012) ‘nothing more than points’, i.e. relative values, as opposed to absolute values. Looking at the values one 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 undervalue some tasks giving them smaller weights. Furthermore, the scrum master has a difficult task of aggregating all assessments into an appropriate value of story points.

The problems regarding scrum and other agile methodologies are interesting for both researchers and practitioners. The problems are usually solved by introducing some soft computing method, e.g. fuzzy logic, genetic or evolutionary algorithms or neural networks, to support or optimize the decision-making process. Bayesian networks were proposed as tool for problem detection in a process of development scrum-based software projects (Perkusich et al., 2015. Chaves-González et al. (2015) were dealing with the next release problem in software development using a multiobjective swarm intelligence evolutionary algorithm. Muker et al. (2014) aimed to enhance the quality of scrum-based software products by optimizing their performance. Beikkhakhian et al. (2015) proposed the three-component system that consists of fuzzy hierarchical analysis method, TOPSIS and AHP method, is used for ranking suppliers in agile supplier selection problem. Luo et al. (2009) developed model based on radial basis function artificial neural network to support supplier selection decision-making process in agile supplier chains. PROMETHEE and AHP are used for selection the optimal agile software development method (Sharma & Bawa, 2016), while fuzzy logic is used to for selection of an appropriate software development life cycle (Öztürk, 2013).

In this paper, we aim to propose a decision support system based on fuzzy logic that deals with the main problems in scrum methodology. A scrum master needs to think about all parameters and its logical dependencies when deciding on the final value of story points for each task. Since fuzzy logic is a useful tool to deal with problems that include imprecise and vague data (Lin et al., 2006), it is ideal to be used for this purpose. The proposed fuzzy logic-based system could enhance efficiency in scrum planning phase. The developers’ task estimation and the scrum master’s knowledge are inputs in the system, while the output is weight (story point value) of the observed task. In this way, we aim to accelerate the decision-making process in scrum, reduce subjectivity of developers and the scrum master and to include dependencies of variables using logic-based rules. The system is dynamic since it utilizes feedback function in each iteration to assess accuracy of developers’ predictions in order to give a more precise prediction over time.

This paper is structured as follows. In Section 2 the basic concepts of fuzzy logic and FIS are given. Further, we provide a brief overview the applications of fuzzy logic within scrum methodology. In Section 3 we introduce our fuzzy logic-based system and its components: FIS, an aggregation operator and a feedback function that should improve accuracy of the system. We create and present a simulation of the proposed system in Section 4. Finally, the main conclusions and ideas for further research are listed in Section 5.

**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, 1996, 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. Fuzzy logic is based on the fuzzy set theory (Zadeh, 1965). Fuzzy sets generalize classical sets, since bivalent membership functions of classical sets are special cases of the membership functions of fuzzy sets, i.e. the membership functions of fuzzy sets take values from the whole unit [0,1] interval. The most common fuzzy membership functions are triangular-shaped (characterized with three values representing its vertices), trapezoidal-shaped (characterized with four values representing its vertices), PI-shaped (characterized with four values where the fist and the last are locating "feet" of the curve, while the others locate its "shoulders"), S-shaped, Z-shaped, Bell-shaped functions, etc. The fuzzy membership function are used for fuzzification, the process of transformation continues variables to [0,1] interval. The inverse process is called defuzzification. The most popular defuzzification methods (Runkler, 1997) are centre of gravity, bisector of area, etc.

In fuzzy logic/fuzzy set theory, operators of conjunction/intersection and disjunction/union are realized using different functions that are referred to as *t*-norms and *t*-conorms (or *s*-norms), respectively. The *min* operator is the standard choice for fuzzy intersection, while algebraic product and *Lukasiewicz* norm are also frequently used (Klement et al., 2004). The *max* operator, probabilistic sum and *Lukasiewicz* *t*-conorm are the corresponding as *t*-conorms. The most common negation operator in fuzzy logic is a standard fuzzy 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 frequently used fuzzy-based technique, applied in different areas such as finance (Yunusoglu & Selim, 2013, Chourmouziadis & Chatzoglou, 2016), medicine (Marzuki et al., 2014, Dragovic et al., 2015), risk assessment (Camastra et al., 2015, Pamučar et al., 2016), etc. FIS are particularly useful in problems where inputs, i.e. fuzzy sets, are expressed as linguistic expressions such as easy, not-that-hard, extremely hard, etc. Just like an algebraic variable takes numbers as values, a linguistic variable takes words or sentences as values. Thus, fuzzy inference systems are operating similarly to human perception and they are easy to interpret and analyze. Furthermore, the modelling using FIS is significantly simpler comparing to classical modelling techniques.

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. The IF part is a logical condition that should be fulfilled in order to the THEN part be realized. IF-THEN rules are commonly specified by a field expert, although they can also be learned from the existing data (Jang, 1993). Two most important types of the FIS are Mamdani (Mamdani, 1977) and Takagi–Sugeno (Takagi & Sugeno, 1985). In the Mamdani system, both inputs and outputs are presented as fuzzy sets, so these systems are very easy to interpret. In the Takagi– 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 and not so close to human perception.

After the process of modelling, i.e. modelling inputs/outputs using fuzzy membership functions and determining a set of IF-THEN rules, the fuzzy inference process consists of four crucial steps: fuzzification, rule evaluation, aggregation and defuzzification. The first step in fuzzy inference is to convert linguistic expressions to values on the unit interval [0,1] using previously defined input membership functions. Further, fuzzy rules are evaluated using chosen operators for t-norm, t-conorm and fuzzy negation. Results of all IF-THEN rules are aggregated into a single fuzzy set. Finally, defuzzification is applied to convert a fuzzy set 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 seems to be an appropriate 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. Mohammed and Darwish (2016) aimed to overcome unclear and ambiguous indicators of agility evaluation using fuzzy logic in order to develop framework for calculating success metrics of agile software projects. Colomo-Palacios et al. (2012) developed a fuzzy logic-based recommender system for supporting the development process in scrum environments and helping to form the most suitable team for different tasks. Bach-Dąbrowska and Wojnar (2013) used fuzzy logic and role patterns to improve candidate assessment and selection process for agile IT project teams. Fuzzy logic is also used for effort estimation accuracy using trapezoidal membership functions (Raslan et al., 2015). 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 decision support system that can be a valuable or even a complete replace 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: a fuzzy inference system, an aggregation function and a feedback function. FIS is used to model inputs and asses output for each developer. Further, estimated scores for each developer is aggregated using suitable function. The proposed system is dynamic and experts’ knowledge is used only in the first iteration in order to set up initial parameters. In further iteration feedback function is used to control system.

**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 (Orlowsky et al., 2006), we decided to exclude the expert group from our system. Experts are usually project leaders and they 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** | **Type** | **Values of parameters** |
| *Junior* | PI - shaped | 0.00. 0,00, 1.20, 2.65 |
| *Intermediate* | PI - shaped | 1.50, 2.70, 3.70, 5.05 |
| *Senior* | PI - shaped | 3.60, 5.20, 6.25, 8.15 |

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 [1,7]. Membership functions’ parameters are shown in Table 2.

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

Table 2: Membership function parameters for EST variable

|  |  |  |
| --- | --- | --- |
| **Variable** | **Type** | **Values of parameters** |
| *Overestimated* | PI - shaped | 0.00, 0.00, 1.85, 3.45 |
| *Well Estimated* | PI - shaped | 2.05, 3.25, 4.60, 6.00 |
| *Underestimated* | PI - shaped | 4.50, 6.30, 7.30, 9.70 |

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 [1,7]. 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** | **Type** | **Values of parameters** |
| *Easy* | PI - shaped | 0.00, 0.00, 1.75, 3.55 |
| *Medium* | PI - shaped | 1.95, 3.35, 4.50, 5.95 |
| *Heavy* | PI - shaped | 4.65, 6.15, 7.25, 9.15 |

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…), 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 output interval. So, instead of using these values, the parameters of output membership functions are rescaled according to scrum master suggestions (Table 5).

Table 5: Output membership functions’ parameters

|  |  |  |
| --- | --- | --- |
| **Group name** | **Type** | **Values of parameters** |
| *Easy* | PI - shaped | 0.00, 0.00, 2.70, 15.55 |
| *Medium* | PI - shaped | 3.55, 10.90, 16.05, 25.35 |
| *Complex* | PI - shaped | 16.80, 23.95, 30.00, 38.80 |
| *Very Complex* | PI - shaped | 28.00, 40.00, 50.00, 55.00 |

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 Estimated*) and (EXP is not *Junior*) and (WEI is *Easy*) THEN (OUT is *Easy*)
2. IF (EST = *Overestimated*) and (WEI is not *Easy*) THEN (OUT is *Easy*)
3. IF (EST = *Underestimated*) and (EXP is not *Senior*) and (WEI is *Easy*) THEN (OUT is *Medium*)
4. IF (EST = *Overestimated*) and (WEI is not *Easy*) THEN (OUT is *Medium*)
5. IF (EST = *Overestimated*) and (EXP is not *Senior*) and (WEI is *Heavy*) THEN (OUT is *Complex*)
6. IF (EST = *Underestimated*) and (EXP is not *Junior*) and (WEI is *Heavy*) THEN (OUT is *Complex*)
7. IF (EST = *Well Estimated*) and (EXP is not *Junior*) and (WEI is *Heavy*) THEN (OUT is Very *Complex*)
8. IF (EST = *Underestimated*) and (WEI is not *Easy*) THEN (OUT is *Very Complex*)

The proposed FIS is Mamdani-type. This type of FIS ensures the necessary transparency of the decision-making process. Conjunction operator is evaluated using *min* function, disjunction operator using *max* function and in the defuzzification process we used centre of gravity (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).

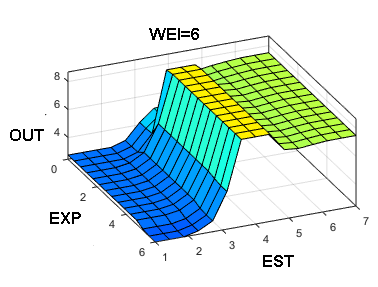


Figure 1: Output values for WEI=6

**3.2** Aggregation function

Each developer has its own input vector and FIS 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. In order to model complex relations and logical dependencies of variables, more advance aggregation functions such OWA operator (Yager, 1988) or logical aggregation (Radojevic, 2008) may 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, represented with REAL\_OUT value. Using this REAL\_OUT and ESTIMATED\_OUT values for the i*th* 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.

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.

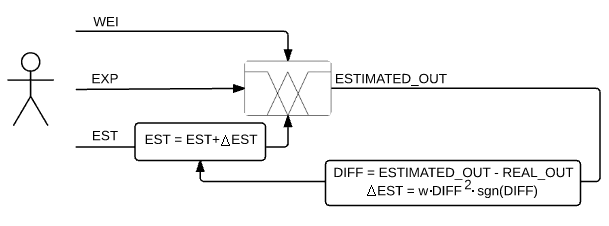


Figure 2: Design of the fuzzy expert system

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.

**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.

uniformly disturbed on the interval [1,7], 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.

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 is increased compared to Sprint 1, while the MSE value is notably 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 accurate over time and gives better 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. Further, we will use various aggregation operators in order to model different problem situations. Finally, we aim to test the proposed system with different parameter setting on the real data in order to evaluate its practical significance.

**References**

1. Fowler, M., & Highsmith, J. (2001). The agile manifesto. Software Development, 9(8), 28-35.
2. Dingsoyr, T., Nerur, S., Balijepally, V., & Moe, N. B. (2012). A decade of agile methodologies: Towards explaining agile software development, Journal of Systems and Software, 85(6), 1213-1221. doi:10.1016/j.jss.2012.02.033
3. Nerur, S., Mahapatra, R., & Mangalaraj, G. (2005). Challenges of migrating to agile methodologies. Communications of the ACM, 48(5), 72-78. doi:10.1145/1060710.1060712
4. Dyba, T., & Dingsoyr, T. (2008). Empirical studies of agile software development: A systematic review. Information and Software Technology, 50(9), 833-859. doi:10.1016/j.infsof.2008.01.006
5. 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.
6. 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.
7. Duchting, 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.
8. Perkusich, M., Soares, G., Almeida, H., & Perkusich, A. (2015). A procedure to detect problems of processes in software development projects using Bayesian networks. Expert Systems with Applications, 42(1), 437-450. doi:10.1016/j.eswa.2014.08.015
9. Chaves-González, J. M., Pérez-Toledano, M. A., & Navasa, A. (2015). Software requirement optimization using a multiobjective swarm intelligence evolutionary algorithm. Knowledge-Based Systems, 83, 105-115. doi:10.1016/j.knosys.2015.03.012
10. Mukker, A. R., Mishra, A. K., & Singh, L. (2014). Enhancing Quality in Scrum Software Projects. International Journal of Science and Research, 3(4), 682-688.
11. Beikkhakhian, Y., Javanmardi, M., Karbasian, M., & Khayambashi, B. (2015). The application of ISM model in evaluating agile suppliers selection criteria and ranking suppliers using fuzzy TOPSIS-AHP methods. Expert Systems with Applications, 42(15), 6224-6236. doi:10.1016/j.eswa.2015.02.035
12. Luo, X., Wu, C., Rosenberg, D., & Barnes, D. (2009). Supplier selection in agile supply chains: An information-processing model and an illustration. Journal of Purchasing and Supply Management, 15(4), 249-262. doi:10.1016/j.pursup.2009.05.004
13. Sharma, A., & Bawa, R. K. (2016). A Framework for Agile Development Method Selection using Modified PROMETHEE with Analytic Hierarchy Process. International Journal of Computer Science and Information Security, 14(8), 846-854.
14. Ozturk, V. (2013). Selection of appropriate software development life cycle using fuzzy logic. Journal of Intelligent & Fuzzy Systems, 25(3), 797-810. doi:10.3233/IFS-120686
15. Lin, C. T., Chiu, H., &Tseng, Y. H. (2006). Agility evaluation using fuzzy logic. International Journal of Production Economics, 101(2), 353-368. doi:10.1016/j.ijpe.2005.01.011
16. Zadeh, L. A. (1996). Fuzzy logic = computing with words. IEEE Transactions on Fuzzy Systems, 4(2), 103-111. doi:10.1109/91.493904
17. Zadeh, L. A. (2008). Is there a need for fuzzy logic?. Information Sciences, 178(13), 2751-2779. doi:10.1016/j.ins.2008.02.012
18. Zadeh, L. A. (1965). Fuzzy sets. Information and Control, 8(3), 338-353. doi:10.1016/S0019-9958(65)90241-X
19. Runkler, T. A. (1997). Selection of appropriate defuzzification methods using application specific properties. IEEE Transactions on Fuzzy Systems, 5(1), 72-79. doi:10.1109/91.554449
20. Klement, E. P., Mesiar, R., & Pap, E. (2004). Triangular norms. Position paper I: basic analytical and algebraic properties. Fuzzy Sets and Systems, 143(1), 5-26. doi:10.1016/j.fss.2003.06.007
21. Yunusoglu, M. G., & Selim, H. (2013). A fuzzy rule based expert system for stock evaluation and portfolio construction: An application to Istanbul Stock Exchange. Expert Systems with Applications, 40(3), 908-920. doi:10.1016/j.eswa.2012.05.047
22. Chourmouziadis, K., & Chatzoglou, P. D. (2016). An intelligent short term stock trading fuzzy system for assisting investors in portfolio management. Expert Systems with Applications, 43, 298-311. doi:10.1016/j.eswa.2015.07.063
23. Dragović, I., Turajlić, N., Pilčević, D., Petrović, B., & Radojević, D. (2015). A Boolean consistent fuzzy inference system for diagnosing diseases and its application for determining peritonitis likelihood. Computational and Mathematical Methods in Medicine, 2015. doi:10.1155/2015/147947
24. Marzuki, A., Tee, S. Y., & Aminifar, S. (2014). Study of fuzzy systems with Sugeno and Mamdani type fuzzy inference systems for determination of heartbeat cases on Electrocardiogram (ECG) signals. International Journal of Biomedical Engineering and Technology, 14(3), 243-276. doi:10.1504/IJBET.2014.059673
25. Camastra, F., et al. (2015). A fuzzy decision system for genetically modified plant environmental risk assessment using Mamdani inference. Expert Systems with Applications, 42(3), 1710-1716. doi:10.1016/j.eswa.2014.09.041
26. Pamucar, D., Bozanic, D., & Komazec, N. (2016). Risk Assessment of Natural Disasters Using Fuzzy Logic System of Type 2. Management, 80, 23-34. doi:10.7595/management.fon.2016.0016
27. Jang, J. S. (1993). ANFIS: adaptive-network-based fuzzy inference system. IEEE Transactions on Systems, Man, and Cybernetics, 23(3), 665-685. doi:10.1109/21.256541
28. Mamdani, E. H. (1977). Application of fuzzy logic to approximate reasoning using linguistic synthesis. IEEE Transactions on Computers, 100(12), 1182-1191.
29. Takagi, T., & Sugeno, M. (1985). Fuzzy identification of systems and its applications to modeling and control. IEEE Transactions on Systems, Man, and Cybernetics, (1), 116-132. doi:10.1109/TSMC.1985.6313399
30. 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.
31. Mohammed, A. H., & Darwish, N. R. (2016). A Proposed Fuzzy based Framework for Calculating Success Metrics of Agile Software Projects. International Journal of Computer Applications. 137(8), 17-23. doi:10.5120/ijca2016908866
32. Colomo-Palacios, R., Gonzalez-Carrasco, I., Lopez-Cuadrado, J. L., & Garcia-Crespo, A. (2012). ReSySTER: A hybrid recommender system for scrum team roles based on fuzzy and rough sets. International Journal of Applied Mathematics and Computer Science, 22(4), 801-816. doi:10.2478/v10006-012-0059-9
33. Bach-Dąbrowska, I., & Wojnar, J. (2013). Role patterns in it projects teams: Design of a selection module using fuzzy logic techniques. Foundations of Management, 5(1), 7-20. doi:10.2478/fman-2014-0001
34. Raslan, A. T., Darwish, N. R., & Hefny, H. A. (2015). Towards a Fuzzy based Framework for Effort Estimation in Agile Software Development. International Journal of Computer Science and Information Security, 13(1), 37-45.
35. Kurian, T. (2011). A Fuzzy Based Approach for Estimating Agility of an Embedded Software Process. http://www.siliconindia.com/events/siliconindia\_events/Global\_Embedded\_conf/Globa\_Embedded\_Conf\_PPT\_final\_tisni.pdf
36. 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.
37. Sivanandam, S. N., Sumathi, S., & Deepa, S. N. (2007). Introduction to fuzzy logic using MATLAB (Vol. 1). Berlin: Springer. doi:10.1007/978-3-540-35781-0
38. Yager, R. R. (1988). On ordered weighted averaging aggregation operators in multicriteria decision making. IEEE Transactions on systems, Man, and Cybernetics, 18(1), 183-190. doi:10.1109/21.87068
39. Radojevic, D. (2008). Logical aggregation based on interpolative Boolean algebra. Mathware & Soft Computing, 15(1), 125-141.
40. Coelho, E. Basu, A (2012). Effort Estimation in Agile Software Development using Story Points, IJAIS, ISSN: 2249-0868 (<https://pdfs.semanticscholar.org/0ae7/5cc0678f6b01f1066093b19cad9dd4379385.pdf>)
41. Cho, J. (2008). Issues and challenges of agile software development with scrum, <http://iacis.org/iis/2008/S2008_950.pdf>

**About the Authors**

**Mihailo Stupar**

University of Belgrade, Faculty of Organizational Sciences

[stupar.mih@gmail.com](mailto:stupar.mih@gmail.com).

Mihailo Stupar is an associate developer at msg global solutions. His major professional interests include software development, data analysis, machine learning, cloud computing

**Pavle Milošević**

University of Belgrade, Faculty of Organizational Sciences

[pavle.milosevic@fon.bg.ac.rs](mailto:pavle.milosevic@fon.bg.ac.rs).

Pavle Milošević is a teaching associate at the University of Belgrade, Faculty of Organizational Sciences. His major professional interests include: fuzzy logic, intuitionistic fuzzy sets, system theory, machine learning, metaheuristics and time series analysis.

Bratislav Petrović, University of Belgrade, Faculty of Organizational Sciences, [bratislav.petrovic@fon.bg.ac.rs](mailto:bratislav.petrovic@fon.bg.ac.rs).

TEKST I SLIKA