Fuzzy FMEA-based Risk Evaluation of Student Software Projects

Zsolt Csaba Johanyák[1,2], Attila Pásztor[1]

[1]GAMF Faculty of Engineering and Computer Science, John von Neumann University, Izsáki út 10, H-6000 Kecskemét, Hungary, johanyak.csaba@nje.hu; pasztor.attila@nje.hu

[2]Bánki Donát Faculty of Mechanical and Safety Engineering, Óbuda University, Népszínház utca 8, H-1081 Budapest, Hungary, johanyak.csaba@bgk.uni-obuda.hu

Abstract: Team project work plays an important role in the curriculum of computer science engineering students. It allows students to synthesize the theoretical and practical knowledge and skills acquired during their previous studies. In the context of project work, student teams are often as-signed a software development task. During the project, teams often encounter various difficulties that can potentially jeopardize the successful completion of the project. A preliminary risk assessment can identify potential hazards in a systematic way, helping the team to prepare for and mitigate the associated risks. This article reports the successful application of Fuzzy Process Failure Mode and Effects Analysis (FPFMEA). The analysis started during the project preparation phase and continued throughout the project life cycle. The article provides valuable insights into how to improve risk management strategies for student-led software projects, helping create a more flexible and successful project development environment. The results demonstrate that this approach enables effective identification of project risks and mitigation of their impact.

Keywords: fuzzy; FMEA; project; evaluation; team work

1 Introduction

The field of software engineering is a typical area where students can acquire up-to-date and practical knowledge through field experience, i.e., by carrying out software projects. In practice, software is almost exclusively a result of teamwork, and therefore it is crucial to include team-based software projects in the curricula of the computer science engineering students. These projects give an exceptional opportunity for students to develop their skills and deepen their theoretical knowledge by trying it in practice as well [1].

However, project work has many pitfalls. Software projects are usually complex, which results in several risks that could hinder the progress of the project or can even endanger the final success. For example, lack of the necessary technical and cooperation skills, different expertise level of the team members, communication difficulties, re-source limits resources occur frequently.

The final success of the project greatly depends on efficient project management and understanding as well as prevention or at least alleviation of the risks endangering the project [2]. A carefully selected and implemented preliminary risk evaluation method can systematically identify, evaluate and rank in an early stage of the project life cycle those factors that endanger the project. Using a robust assessment technique could help the students to develop a tailored risk management strategy to effectively mitigate the risk levels and enhance their understanding of risk assessment. Students usually adopt an incremental and iterative development model [3], thus the used risk evaluation method should support periodic reviews and updates as well.

Process Failure Mode and Effects Analysis (PFMEA) is a well-established and widely used technique aiming the identification and ranking of potential failures and risks associated to the steps of a process [4]. PFMEA focuses on failure modes, causes, their impact and effects as well as the possible mitigation solutions to improve the efficiency, reliability, and quality of the investigated process. Evaluating these factors in the case of a software project carried out by a student team is am inherently subjective process containing vagueness which partly can be traced back to the lack of experience in risk assessment [5].

In the almost 60-year history of fuzzy logic, it has been successfully applied to a wide range of problems, including among others economics [6], aircraft control [7], freight and supply chain management [8], etc. Fuzzy logic proved to be particularly useful when dealing with imprecise input, and human-like reasoning would be advantageous. Therefore, combining FMEA with a fuzzy approach enables the inclusion of subjective or incomplete evaluations [9], the evaluations given with linguistic terms [10], and the handling of inconsistencies [11], thus leading sometimes to more accurate results [12].

The paper presents the application of Fuzzy PFMEA as a systematic approach for identifying and ranking risks in software projects carried out by student teams under instructor supervision. The paper also provides a list of recommended actions to mitigate risks and increase the probability of successful project completion.

The rest of this paper is organized as follows. Section 2 presents a review of related works. Section 3 describes the fuzzy FMEA methodology applied for risk evaluation of the investigated student project. Section 4 presents the results of the analysis and the conclusions are drawn in Section 5.

2 Related Works

Ahtee and Poranen [13] studied risks in software projects carried out by students. They analyzed 76 final reports and identified four major risks faced by students: tools and skills, technological problems, scheduling problems, and working or studying simultaneously with the project.

Koolmanojwong and Boehm [2] analyzed the risks faced by student teams, explored the relationship between effective risk analysis and project success, and discussed how risk patterns influence students' course of action. They collected data using Distributed Assessment of Risk Tool (DART), milestone reports, and surveys.

Thota, Niu, Wang, and Purdy [3] investigated how undergraduate students who took part in agile software teams prioritized and alleviated risks. The top risks identified by the students were significantly different from the ones that originated from industry surveys.

Kirk, Luxton-Reilly, and Tempero [14] investigated the use of project management reference models to identify the main areas of risk for student projects. The computer science education (CSE) literature on group projects was mapped to the PMBOK® Knowledge Areas (KAs), revealing a subset of relevant KAs. The key risks for student projects were categorized using these KAs.

Khuankrue et al. [5] proposed a methodology for applying a fuzzy failure mode and effects analysis (FMEA) model to support project-based software engineering education. The presented methodology uses intelligent agents to construct membership functions for a fuzzy rule-based system. These agents learn from historical data and assist students in developing expertise in risk assessment.

Erbay and Özkan [11] proposed the integration of fuzzy cognitive maps (FCMs) in-to the Fuzzy FMEA methodology to manage the risks of a software project. They aimed to create a technique that allows to take into consideration the relationships between the risks of a project under ambiguous circumstances.

Ibraigheeth and Abdullah [15] reported the development of an expert risk evaluation system based on empirical study and real data from software projects to identify factors that affect project success. They did not focus specifically on student projects, but rather on real-world projects to help decision-makers evaluate the expected risks and estimate risk probability based on critical success factors.

3 Fuzzy FMEA

The Process Failure Mode and Effects Analysis (PFMEA) [16] [17] is a widely used method in quality and risk management that aims to conduct a detailed and flexible analysis of inherent risks and failure possibilities within a process. The goal is to reduce risks and costs while improving overall quality. The Fuzzy PFMEA (FPFMEA) extends the original method by replacing the multiplication-based Risk Priority Number (RPN) calculation with a fuzzy inference-based RPN determination. The introduction of fuzzy logic in PFMEA execution allows for the consideration of subjective risks and uncertain-ties in the analysis, as perceived by the individuals conducting it. Typically, FMEA is carried out in teams, leveraging the expertise of team members. In almost all steps, field experience and knowledge are crucial requirements for achieving final success. When analyzing a software development process, specialists involved in architecture design, implementation, and testing should always be part of the team.

The flow of FPFMEA is presented in Figure 1, with detailed steps described below.

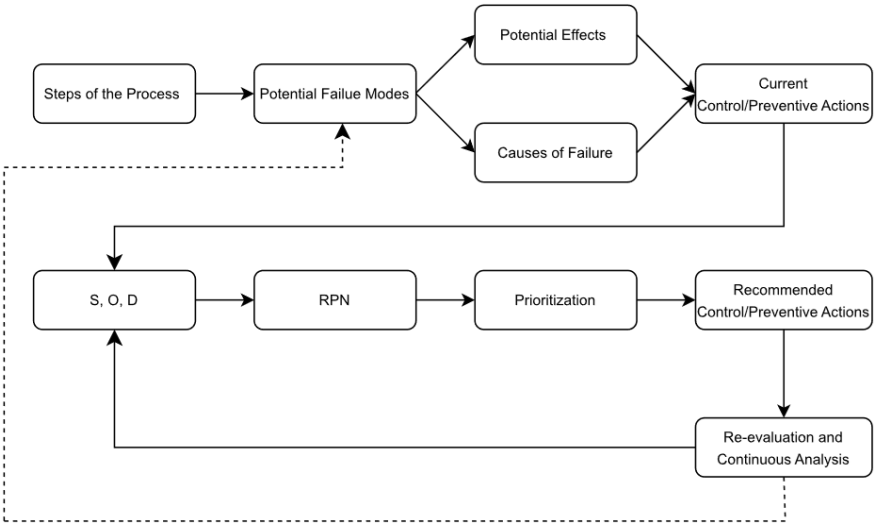

Figure 1
Flow of FPFMEA

Steps of the Process: The analysis begins by determining the steps of the process being investigated. This activity can be supported by creating a flowchart.

Potential Failure Modes: For each step, one or more failure possibilities that could jeopardize the successful completion of the process are identified. Visual tools, such as fishbone (Ishikawa) diagrams, can provide crucial support for organizing ideas of team members.

Potential Effects: Each failure mode can influence the process or its outcome in different ways that have to be identified during this step of the analysis. In several cases, effects can be categorized as internal (e.g., performance loss, software downtime, data loss) and external (e.g., customer dissatisfaction, loss of reputation, legal consequences).

Causes of Failure: Identifying the underlying reasons or mechanisms leading to a specific failure mode is one of the most critical steps of an FMEA. This identification allows the team to address root causes and implement effective corrective actions. This activity can also be supported by a fishbone diagram.

Current Control/Preventive Actions: Documenting actions and measures that counteract the possibility of a given failure mode occurring in the current process form.

Evaluation of Risks Associated with Identified Failure Modes: This activity includes evaluating severity (S), occurrence (O), and detection (D), each rated on a scale from 1 to 10. Severity indicates how serious the risk associated with the potential failure is. It is strongly related to the effects of the failure. Occurrence shows how often a given failure mode is expected to occur. Detection expresses how likely will the failure slip through the current control. The evaluation of the three aspects is usually supported by rating catalogs defined by standards, textbooks, or internal guides of the company. In the case of FPFMEA a fuzzy partition is defined for each of them (e.g. see Figure 3 and Table 1), which allows the team to think in overlapping categories expressed by fuzzy sets. In this case, the rating guides (e.g. Tables 2, 3, and 4) explain the meaning of these categories.

The evaluations (S, O, and D) given by the team are fuzzified in their respective partition, followed by a Mamdani-type fuzzy inference. This inference is based on IF-THEN type fuzzy rules, where the antecedent parts are fuzzy sets from the Severity, Occurrence, and Detection partitions. The output fuzzy set is defuzzified using an arbi-trary defuzzification method, resulting in a value between 1 and 1000. This value is called *Risk Priority Number* (RPN) and represents the risk associated with the given cause-failure mode-effect tuple.

Recommended Preventive and Control Actions: All risks that have an RPN value high-er than a threshold are addressed by the team to find effective measures either to prevent their happening or to ensure that they will be recognized in time. All problems cannot be solved instantly therefore the team prioritizes and addresses problems in decreasing order of RPNs. For each recommended action, a responsible person and a deadline are defined.

Re-evaluation and Continuous Analysis: After the implementation deadline, the team re-evaluates cause-failure mode-effect combinations, focusing on the actions taken and their effect on the RPN value. New actions are prescribed if necessary. The first round of analysis occurs during the planning phase of the process before its actual implementation. However, FMEA is a regular review following process

implementation, and in any round, new failure modes, causes, or effects may be identified (see dashed line in Figure 1). Continuous improvement and periodic reviews may be always necessary to ensure ongoing effectiveness.

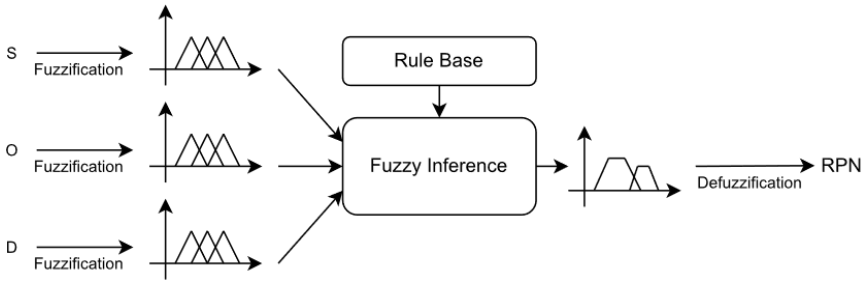

Figure 2
Fuzzy inference based RPN determination

4 Analysis in Practice

Students majoring in computer engineering are required to solve a complex problem in the sixth semester of their studies in a so-called project task. It allows them to synthesize the knowledge acquired as a result of several courses as well as to deepen their practical skills by using a comprehensive approach to problem solving. In a project assignment, four or five students work together as a team under the supervision of an instructor. Their task usually includes research, planning, and execution steps.

In the case of software projects, a preliminary risk assessment can significantly lower the probability of unsuccessful completion of the course and contribute to a higher quality final product. This section presents the results of a fuzzy FMEA-based risk evaluation started right at the beginning of the project work and maintained throughout the project. The analysis was elaborated by the team members and the instructor together. The task was to develop a library containing the implementation of the optimization algorithms: Newton's method, Bees Algorithm, Harmony Search, and Bacterial Memetic Algorithm. The target platform was .NET framework, while the expected tools were C# language and Visual Studio. The library had to be implemented as a class library, which can then be easily used by any application requiring optimization. All the desired methods are used to do optimization over continuous variables using an iterative approach.

As a preparatory step, we defined fuzzy partitions for all three aspects of the analysis (see Table 3 Figure 3) as well as for the RPN (see Figure 4 Table 5). To help the team carry out the analysis for each fuzzy set of the linguistic variables S, O,

and D we also defined a short explanation and an example for each of them (see Tables 2, 3, and 4). A fuzzy rule base was also defined to support the RPN determination. Three example rules are shown below, while the rest of the rule base is available on GitHub through the link given in the Supplementary Materials section. The fuzzy system uses Mamdani type inference. Several programming instructors and a group of students who successfully completed the project work course before took pat in the validation of the rule base and the fuzzy model. Their feedback was incorporated into the final version.

Different defuzzification types were tried in course of the development of the fuzzy model. Eventually the centroid type provided the results that were conforming to our expectation and the demands of the people participating in the validation of the system.

If (S is VL) and (O is VL) and (D is VL) then (RPN is VL)

If (S is VH) and (O is VH) and (D is M) then (RPN is VH)

If (S is VH) and (O is VH) and (D is VH) then (RPN is VH)

Table 1
Linguistic values and fuzzy membership function parameters for the three aspects of FMEA

Linguistic values	Severity (S) / Occurrence (O) / Detection (D) parameters
Very low (VL)	(1.00, 1.00, 3.25)
Low (L)	(1.00, 3.25, 5.50)
Medium (M)	(3.25, 5.50, 7.75)
High (H)	(5.50, 7.75, 10.00)
Very High (VH)	(7.75, 10.00, 10.00)

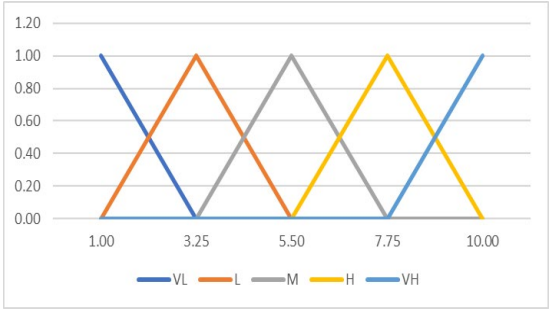

Figure 3
Membership functions for the three aspects (S, O, D) of FMEA

Table 2

Rating guide for Severity with example cases related to the failure "poor communication"

Linguistic values	Explanation and example
Very low (VL)	A minor failure in the software process that does not impact the project timeline or meeting the deadline. For example, there is a slight misunderstanding about the preferred naming convention for variables or functions.
Low (L)	A failure that has less impact on the performance of the team. For example, team members have different interpretations of the algorithmic steps or parameter settings.
Medium (M)	Regular communication breakdowns leading to delays in task completion, occasional rework, and some impact on team morale. For example, a miscommunication that leads to significant deviations from the intended behavior or performance of the optimization algorithms.
High (H)	A failure that has a significant impact on the software project. It causes serious problems, but they can be solved. For example, team members misunderstand the security requirements or fail to address a crucial performance constraint.
Very High (VH)	Very serious failures that are critical for the software project. It has very severe consequences. For example, team members fail to communicate essential requirements, dependencies, or deadlines, resulting in a complete mismatch between expectations and deliverables.

Table 3

Rating guide for Occurence with example cases related to the failure "poor communication"

Linguistic value	Explanation and example
Very low (VL)	The likelihood of the occurrence of the failure is very low. For example, all team members are at the same physical location and they have a history of effective communication and good collaboration.
Low (L)	Rare or temporary failures. For example, team members are in different time zones but have regular virtual meetings, and they use communication tools effectively. Occasional challenges are possible.
Medium (M)	The failure has a moderate chance to occur. For example, the team members are at different locations and they use asynchronous communication tools.
High (H)	There are significant factors or conditions that increase the chances of failure. For example, team members have different communication preferences and their schedule does not allow even virtual meetings.
Very High (VH)	There are strong indicators or conditions that make failure highly probable or imminent. For example, team members are at different locations and do not have any experience in working together or working in a team.

Table 4
Rating guide for Detection with example cases related to the failure “poor communication”

Linguistic values	Explanation and example
Very low (VL)	Very easily recognizable failure or risk. It is sure that it will be detected by the current control measures. For example, the team is well coordinated and automated task management and communication tracking tools are used.
Low (L)	The detection of the failure is highly likely by the current monitoring measures. For example, updates are consistently communicated and the team uses a centralized communication system.
Medium (M)	The detection of the failure is moderately likely using the current detection methods. For example, there are regular team meetings but no logs are kept.
High (H)	The failure is hard to detect, it can easily escape the current control measures. For example, inconsistent reporting methods are used across teams or the project documentation is not consistently updated.
Very High (VH)	It is almost impossible to detect immediately the failure. For example, team members work in different time zones, and there is no established protocol for asynchronous communication. Thus, misunderstandings in project requirements might stay unnoticed for a long time.

Table5
Linguistic values and fuzzy membership function parameters for the risk priority evaluation

Linguistic values	Risk Priority (RPN) set parameters
Very Low (VL)	(1.00, 1.00, 167.50)
Low (L)	(1.00, 167.50, 334.00)
Medium Low(ML)	(167.50, 334.00, 500.50)
Medium (M)	(334.00, 500.50, 667.00)
Medium High (MH)	(500.50, 667.00, 833.5,)
High (H)	(667.00, 833.50, 1000.00)
Very High (VH)	(833.50, 1000.00, 1000.00)

Thus, the student team guided by an instructor could start their work with planning the steps of the development process (Figure 5). It starts with the designing of the Application Programming Interface (API) of the library followed by the development of implementation of each algorithm. After choosing the current algorithm the team has to study the theoretical description, do the implementation, and test it. The latter two steps could involve some repetitions if the results of the tests revealed the necessity of a modification in the implementation. Next, the module has to be integrated in the library followed by integration tests that could also result in some modification in the implementation. Having the integration process successfully finished the team can choose the next optimization method to

be implemented. The above presented steps can include one or more substeps as well (Table 6).

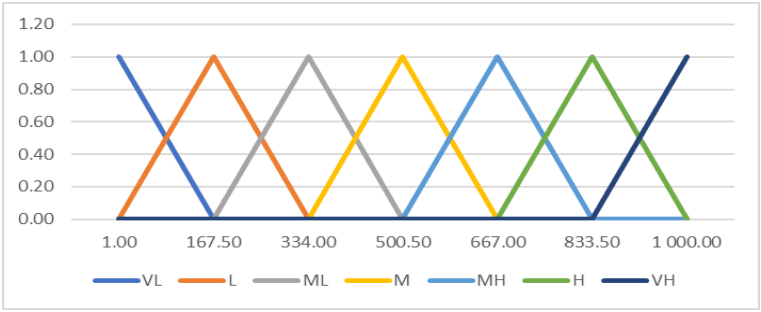

Figure 4
Membership functions for the risk priority evaluation

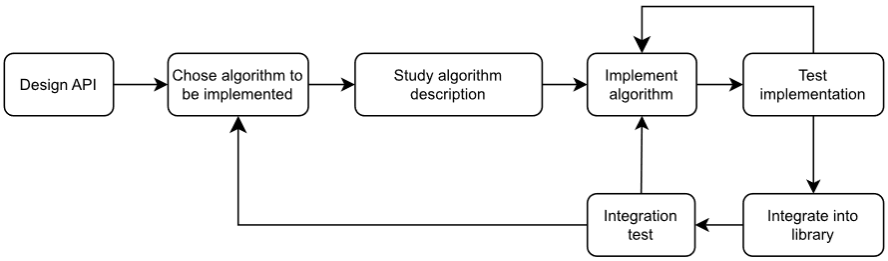

Figure 5
Flow of the optimization library development

In the case of each substep of the above presented flow the team carefully investigated all the possible failure modes that could threat the successful completion of the project. After evaluating the S, O, and D aspects the failure-effect-cause tuples were ordered in decreasing order of their RPN values followed by the definition of counter measures aiming the mitigation of their risk. The RPN values of the discovered risks in decreasing order are presented in the bar chart in Figure 6. Under each bar the number indicates to which failure-effect-cause tuple it belongs. The first ten rows of the FMEA table containing the first ten most critical risks are presented in Table 7. The rest of the analysis is available on GitHub through the link given in the Supplementary Materials section.

Table 6
Substeps of the development process

Main step	Substep
Communication	Communication
Teamwork	Teamwork
Design API	Study existing data structures

	Create API
Study algorithm description	Study algorithm description
Implement algorithm	Search for existing implementation
	Data structure design
	Adapt an existing sample solution
	Implement algorithm
	Chose default values for hyperparamters
	Performance optimization
	Documentation
Unit test	Unit test
Integrate into library	Compatibility check
	Integration with the visual interface
Integration test	Integration test

After carrying out the recommended actions each failure-effect-cause tuple was re-evaluated and a significant decrease of the RPN values could be noticed. The results obtained in the case of the ten most significant risks are presented in Table 8. The rest of the results of the re-evaluation is available on GitHub through the link given in the Supplementary Materials section. The bar chart of the whole list of re-evaluated RPN values are shown in Figure 6.

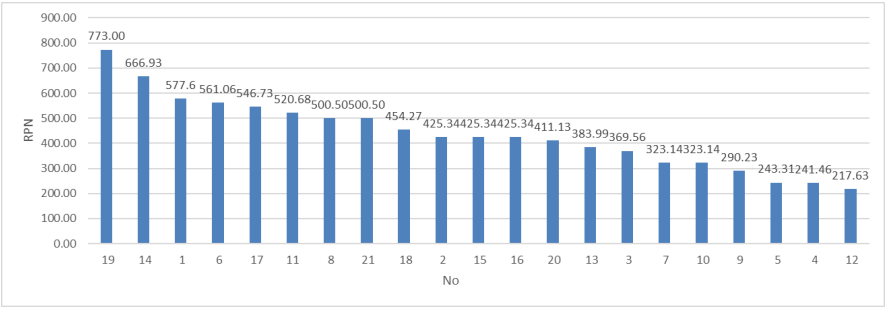

Figure 6
The RPN values of the risks in decreasing order

Table 7
The ten most critical risks, their evaluation, and the recommended actions

					Current status					
No	**Substep**	**Potential failure mode**	**Potential effect of failure**	**Potential cause of failure**	**Control, preventio**	**S**	**O**	**D**	**RPN**	**Recommended action**
19	Unit test	The test plan does not give full code coverage	Implementation not tested properly	Missing test cases	-	8	7	7	773.00	Usage of automatic code coverage evaluation tools.
14	Chose default values for hyperparamters	Sub-optimal values	Reduced efficiency of the algorithm	Lack of attention or lack of knowledge	-	8	3	8	666.93	Hyper-paramter review during team meetings
1	Communication	Poor communication	Team members misunderstand the security requirements	Insufficient security related knowledge	-	8	3	7	577.6	Dedicate meeting for discussing security requirements
6	Study algorithm description	Incorrect interpretation	Incorrect implementation	Lack of basic theoretical knowledge	-	5	4	8	561.06	Discussion of the algorithm during team meatings
17	Documentation	Poor documentation	Increased likelihood of introducing errors during updates	Incomplete or unclear comments in the code	-	4	6	6	546.73	Regular code review
11	Implement algorithm	Lack of error/exception handling	Unhandled exceptions leading to program crashes or unexpected behavior	Overlooking potential failure points	-	7	3	6	520.68	Emphasize on team meetings the importance of error handling mechanisms
8	Search for existing implementation	The sample solution is not correct	Incorrect implementation	The developer of the sample solution made a wrong implementation	Unit test	9	2	5	500.50	Code review and test on sample data
21	Integration with the visual interface	Integration not possible	The visual interface of the library lacks the implementation of the algorithm	Inadequate background knowledge	-	9	5	2	500.50	Consultation, request for help
18	Documentation	Poor documentation	Increased likelihood of introducing errors during updates	Neglecting documentation during development	-	4	5	5	454.27	Maintain up-to-date documentation. Conduct code reviews to ensure clarity
2	Teamwork	Ineffective collaboration	Duplication of efforts	Poorly defined responsibilities	-	5	3	5	425.34	Define roles and responsibilities. Conduct regular team meetings and code reviews.

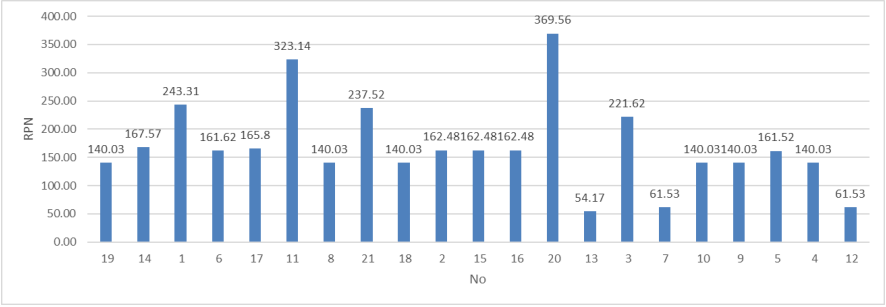

Figure 7
The RPN values of the risks after the re-evaluation

Table 8
Re-evaluation of the most critical risks after taking actions

		Action results				
No	Potential failure mode	Action taken	S	O	D	RPN
19	The test plan does not give full code coverage	Usage of automatic code coverage evaluation tools.	4	2	1	140.03
14	Sub-optimal values	Hyper-paramter review	7	1	1	167.57

		Action results				
No	Potential failure mode	Action taken	S	O	D	RPN
		during team meetings				
1	Poor communication	Dedicated meeting for discussing security requirements	8	1	2	243.31
6	Incorrect interpretation	Algorithm discussed	5	1	2	161.62
17	Poor documentation	Regular code review	4	3	1	165.8
11	Lack of error/exception handling	Emphasize on team meetings the importance of error handling mechanisms. Create unit tests for stress testing.	7	2	2	323.14
8	The sample solution is not correct	Code review and test on sample data	3	2	1	140.03
21	Integration not possible	Consultation, request for help	5	2	2	237.52
18	Poor documentation	Maintain up-to-date documentation. Conduct code reviews to ensure clarity	4	2	1	140.03
2	Ineffective collaboration	Define roles and responsibilities. Conduct regular team meetings and code reviews.	5	1	1	162.48

Conclusions

The preliminary risk assessment presented in this paper yielded positive results. The team identified important risk factors and defined countermeasures that mitigated the risks associated with the cause-failure-effect tuples. For instance, the use of automatic code coverage evaluation tools reduced the RPN value by 82% in cases where the unit tests failed to provide full code coverage. In addition, introducing hyper-parameter reviews during team meetings helped avoid sub-optimal values and reduced the related RPN value by 75%. The team also made significant efforts to improve inter-team communication and included regular code reviews, which helped alleviate multiple problems and ensure correct implementation of optimization algorithms.

Although there are still some topics where the RPN reduction could be considered moderate, including the analysis as the first step of the project work led to clear improvements in the approach and attitude of students towards software development and teamwork. It also helped them to synthesize and apply their theoretical and practical knowledge and experiences acquired during previous courses and individual projects.

After the successful execution of the FPFMEA for a student team work software project one year later in the case of half of the student teams similar analyses were

carried out. At the end of the semester these teams managed to get in average about 19% higher evaluation (points) for their projects.

Further research will focus on improvement of the fuzzy model by considering alternative inference and model building techniques like the ones presented in [18], [19], [20], and [21].

Supplementary Materials

The definition of the fuzzy system using Matlab's Fuzzy Logic Toolbox's FIS format as well as the Excel file containing the FMEA table and the charts with the RPN values can be downloaded at: https://github.com/jzscsaba/FFMEARiskEvalStudSwtProj

References

[1] C. Farré, X. Franch, M. Oriol, and A. Volkova, "Supporting Students in Team-Based Software Development Projects: An Exploratory Study," in *Research Challenges in Information Science: Information Science and the Connected World*, Vol. 476, S. Nurcan, A. L. Opdahl, H. Mouratidis, and A. Tsohou, Eds., in Lecture Notes in Business Information Processing, Vol. 476, Cham: Springer Nature Switzerland, 2023, pp. 568-576, doi: 10.1007/978-3-031-33080-3_39

[2] S. Koolmanojwong and B. Boehm, "A look at software engineering risks in a team project course," in *2013 26th International Conference on Software Engineering Education and Training (CSEE&T)*, San Francisco, CA, USA: IEEE, May 2013, pp. 21-30, doi: 10.1109/CSEET.2013.6595233

[3] V. R. C. Thota, N. Niu, W. Wang, and C. Purdy, "Students' Perceptions of Software Risks," in *2017 ASEE Annual Conference & Exposition Proceedings*, Columbus, Ohio: ASEE Conferences, Jun. 2017, p. 28871, doi: 10.18260/1-2--28871

[4] A. Boldizsár, E. Török, and A. Pásztor, "Supplier Qualification Using FMEA in a Meat Company," *Periodica Polytechnica Transportation Engineering*, Vol. 51, No. 4, pp. 323-328, 2023, Accessed: Dec. 28, 2023, [Online] Available: https://pp.bme.hu/tr/article/view/22548

[5] I. Khuankrue, F. Kumeno, Y. Ohashi, and Y. Tsujimura, "Applying Fuzzy Rule-Based System on FMEA to Assess the Risks on Project-Based Software Engineering Education," *JSEA*, Vol. 10, No. 07, pp. 591-604, 2017, doi: 10.4236/jsea.2017.107032

[6] S. Gáspár, Z. Musinszki, I. Z. Hágen, Á. Barta, J. Bárczi, and G. Thalmeiner, "Developing a Controlling Model for Analyzing the Subjectivity of Enterprise Sustainability and Expert Group Judgments Using Fuzzy Triangular Membership Functions," *Sustainability*, Vol. 15, No. 10, p. 7981, May 2023, doi: 10.3390/su15107981

[7] J. Vascak, P. Kovacik, K. Hirota, and P. Sincak, "Performance-based adaptive fuzzy control of aircrafts," in *10th IEEE International Conference on Fuzzy Systems (Cat. No.01CH37297)*, Melbourne, Vic., Australia: IEEE, 2001, pp. 761-764. doi: 10.1109/FUZZ.2001.1009066

[8] R. Karmakar, S. K. Mazumder, M. B. Hossain, C. B. Illes, and A. Garai, "Sustainable Green Economy for a Supply Chain with Remanufacturing by Both the Supplier and Manufacturer in a Varying Market," *Logistics*, Vol. 7, No. 3, p. 37, Jul. 2023, doi: 10.3390/logistics7030037

[9] O. M. Testik and E. T. Unlu, "Fuzzy FMEA in risk assessment for test and calibration laboratories," *Quality & Reliability Eng*, Vol. 39, No. 2, pp. 575-589, Mar. 2023, doi: 10.1002/qre.3198

[10] E. Kadena, S. Koçak, K. Takács-György, and A. Keszthelyi, "FMEA in Smartphones: A Fuzzy Approach," *Mathematics*, Vol. 10, No. 3, p. 513, Feb. 2022, doi: 10.3390/math10030513

[11] B. Erbay and C. Özkan, "Fuzzy FMEA Application Combined with Fuzzy Cognitive Maps to Manage the Risks of a Software Project," *European Journal of Engineering and Formal Sciences*, Vol. 2, No. 2, p. 7, Jun. 2018, doi: 10.26417/ejef.v2i2.p7-22

[12] C. Arslan Kazan, H. Koruca, and B. Karatop, "Cost Optmization with Internet Supported Fmea and Fuzzy Fmea Analysis," *Mehmet Akif Ersoy Üniversitesi İktisadi ve İdari Bilimler Fakültesi Dergisi*, Vol. 10, No. 2, pp. 950-970, Aug. 2023, doi: 10.30798/makuiibf.1097590

[13] T. Ahtee and T. Poranen, "Risks in Students' Software Projects," in *2009 22nd Conference on Software Engineering Education and Training*, Hyderabad, India: IEEE, 2009, pp. 154-157. doi: 10.1109/CSEET.2009.31

[14] D. Kirk, A. Luxton-Reilly, and E. Tempero, "Risks in Student Projects," in *Australasian Computing Education Conference*, Virtual Event Australia: ACM, Feb. 2022, pp. 143-152. doi: 10.1145/3511861.3511877

[15] M. A. Ibraigheeth and S. Abdullah, "Fuzzy Logic Driven Expert System for the Assessment of Software Projects Risk," *ijacsa*, Vol. 10, No. 2, 2019, doi: 10.14569/IJACSA.2019.0100220

[16] A. Koncz, Z. C. Johanyák, and L. Pokorádi, "Fuzzy approaches in failure mode and effect analysis," *Int. J. Artif. Intell.*, Vol. 19, No. 1, pp. 56-76, 2021, Accessed: Dec. 28, 2023 [Online] Available: https://www.aut.upt.ro/~rprecup/IJAI_82.pdf

[17] L. Pokorádi, S. Koçak, and E. Tóth-Laufer, "Fuzzy failure modes and effects analysis using summative defuzzification methods," *Acta Polytechnica Hungarica*, Vol. 18, No. 9, pp. 111-126, 2021, Accessed: Dec. 28, 2023 [Online] Available: http://acta.uni-obuda.hu/Pokoradi_Kocak_Toth-Laufer_116.pdf

[18] I. A. Zamfirache, R.-E. Precup, R.-C. Roman, and E. M. Petriu, "Reinforcement Learning-based control using Q-learning and gravitational search algorithm with experimental validation on a nonlinear servo system," *Information Sciences*, Vol. 583, pp. 99-120, Jan. 2022, doi: 10.1016/j.ins.2021.10.070

[19] T. Tompa, S. Kovacs, D. Vincze, and M. Niitsuma, "Demonstration of expert knowledge injection in Fuzzy Rule Interpolation based Q-learning," in *2021 IEEE/SICE International Symposium on System Integration (SII)*, Iwaki, Fukushima, Japan: IEEE, Jan. 2021, pp. 843-844, doi: 10.1109/IEEECONF49454.2021.9382734

[20] S. Blazic, D. Dovzan, and I. Skrjanc, "Cloud-based identification of an evolving system with supervisory mechanisms," in *2014 IEEE International Symposium on Intelligent Control (ISIC)*, Juan Les Pins, France: IEEE, Oct. 2014, pp. 1906-1911, doi: 10.1109/ISIC.2014.6967642

[21] Z. C. Johanyák, "Fuzzy rule base identification using an incremental approach," *Gradus*, Vol. 8, No. 2, pp. 129-136, 2021, doi: 10.47833/2021.2.CSV.004