Teamwork Distribution: Local vs. Global Software Engineering Project Development Teamwork

https://doi.org/10.3991/ijet.v15i18.15489

Ismail Al-Taharwa
The University of Jordan, Aqaba, Jordan
i_taharwa@ju.edu.jo

Abstract—Deliverable and course project become the preferred mean to measure learner competency and attainment of intended learning outcomes in IT-fields. Proper setup and evaluation of teamwork projects remains a crucial challenge for e-learning systems. This study investigates the possibility to improve the early prediction of academic software engineering project failure by treating teamwork differently according to the distribution of teamwork participants. Two configurations of teamwork distribution are considered. In the first configuration, a teamwork may include international participants, but all team participants are affiliated to the same institution, namely local teamwork. In the second configuration, a teamwork may include participants from different institutions, namely global teamwork. Software engineering projects are approached from two distinct perspectives. First, obeying the best practices during the system development life cycle (SDLC), namely, process perspective. Second, characteristics of the final deliverable deployed at each milestone of the SDLC, namely, product perspective. A publicly released dataset collected by a designated e-learning environment is leveraged to validate the proposed approach. Results indicate a noticeable variance among local and global distributions. These results put evidence that the reasons behind software engineering teamwork project failure may vary depending on the distribution of the teamwork, local vs. global. Consequently, it advises to customize e-learning systems according to the teamwork distribution differently.

Keywords—E-learning, Software engineering, Project development, Machine learning, Project failure, Teamwork distribution.

1 Introduction

Software engineering project management is gaining more attention from different perspectives. Large-scale projects that involve teamwork are becoming the new trend for software development [1,2]. To shorten the development period and minimize testing & maintenance efforts, software developers prefer to collaborate in teamwork [3]. Students, practitioners, learners, and trainees start to rely on the new paradigm of software development [4-6]. The availability of massive open online courses (MOOCs) platforms eliminate geographical barriers, language differences, and cultural diversity among learners [7]. Wars and political conflicts are a motive for off-class collaboration in many regions in the world. Covid-19 pandemic [8] resulted in an emergent and inevitable need to shift toward e-learning [9-11].

A software engineering project is the proper method for evaluating practical skills that match the top goals in Bloom's taxonomy [12]. Measuring learner's competency in projects remains a challenge due to many factors. Those factors include estimation of tacit knowledge transfer, quantifying on lab practices, dealing with distant learning issues, obeying to good practices, and evaluating teamwork dynamics [13-18]. Academic software engineering teamwork requires special care. Lack of experience, variance in knowledge levels, and acceptance of other personalities are extra challenges against academic teamwork [14,19]. Undergraduate students or even graduate students may make slight modifications to the project idea or substantial changes, ask to change their teamwork, drop the project, or fail to deliver the final product [20].

A key challenge against successful software engineering is the early prediction of project failure. Unclear project goals, poor teamwork communication, imprecise project requirements, and many other factors may contribute to project failure [21]. Despite there is a fair number of studies exploring project failure analysis and prediction from the industrial perspective [22, 23]. There is a limitation and shortage in studies that investigate the issue of academic software engineering project failure. This study investigates the possibility of enhancing the early prediction of academic software engineering project failure by considering teamwork distribution. Specifically, two distinct categories of teamwork are considered. First, teamwork participants who are affiliated to the same academic institution, namely local teamwork. Second, teamwork participants who are affiliated to different institutions, namely, global teamwork [5, 24].

In order to assert this study objective, a real-world, publicly released dataset is utilized [24]. Decision tree classifier was applied to three different distributions of the dataset. The first and the second distributions deal with local and global teamwork, respectively. The third distribution deals with both local and global teamwork in order to provide a ground truth. Results indicate a relatively noticeable variance in prediction rates. This observation emphasizes the need to treat educational software engineering teamwork differently according to the teamwork distribution. In this paper, there are four contributions that answer the following research questions:

- RQ1: Do academic educators need to consider teamwork distribution when modeling evaluation systems for software engineering projects?
- RQ2: How does teamwork distribution affect the early prediction of academic software engineering project failure?
- RQ3: How does teamwork distribution affect the dynamics of software engineering teamwork?
- RQ4: Does the shortage in real-world data affect prediction models of software engineering? Can techniques of instance re-sampling alleviate such effect?

The rest of this paper is organized as follows, Section 2 investigates the Background and related work. The methodology is elaborated in Section 3. Results are presented and discussed in Section 4 and Section 5, respectively. Finally, this paper is concluded in Section 6.

2 Background

Recently, there has been an increasing attention towards knowledge discovery and management in education. Many researchers are concerned with mining education data and processes from different angles [25-28]. Finding new strategies to evaluate learner's knowledge instead of traditional exam-based strategies has been investigated widely [27,29,30]. Building automated tutoring systems that can deliver materials in an appropriate format that suites learner's level of knowledge and utilize computer and multimedia aided design [31,32]. Adaptive online assessment and examination systems have been proposed and widely used in real-world contexts to accommodate a wider range of competency levels [33]. Quite recently, engineering-based approaches were proposed to enhance the measurement of the learner's attention level by leveraging brain computer interface (BCI) [34-36].

Software engineering is a branch of computer science that includes the design, development, evaluation, and implementation of computer software [37,38]. System testing and maintenance remains a key activity in software engineering that is concerned with error reporting and correction [39-42]. Academic software engineering projects are increasingly gaining attention as a reliable method to assess competency levels among ICT students [4,14,19]. Qualitative and quantitative analysis based on questionnaires have been proposed and evaluated in various ways [43,44]. The principal motive was to explore the proper criteria to evaluate the project's final product. However, few research efforts made toward the assessment and evaluation of academic software engineering projects. In [19], the authors considered a student peer assessment scale in addition to the lecturer's overall scale to enhance the reliability of the evaluation. Quite recent studies shed light on the importance of incorporating maintained practices across the project development process in addition to the final product [14,44]. The principal purpose is to quantify to which extent do students obey to the good practices of software engineering.

Academic shift toward large-scale software engineering project development is affected by proper collaboration and coordination among project participants [2,4] Quite recent research efforts anticipated failure among open source software engineering projects due to poor teamwork collaboration [22]. Dhir et al. [20] investigated success and failure factors among academic software engineering projects, focusing on the agile approach of software development. In [21], authors found that poor establishment of requirements in addition to technical skills shortage are the critical risk factors behind software engineering project failure. Research efforts in [45] advise an automated approach for source code assessment leveraging long short-term memory (LSTM) neural network. They advocate the use of their proposed model in programming education. Particularly in software engineering to improve code debugging and error correction. Their proposed model focuses on product perspective, i.e., source code, restrictively. Further, there is no consideration or mention for teamwork assessment.

The aforementioned works tackled the issue of providing a comprehensive academic software engineering e-learning system from distinct perspectives. Petkovic et al. [24] study connected the dots to reshape the contemporary software engineering project delivery, administration, monitoring, and evaluation activities. They designated a dedicated online environment for project development, namely SETAP. SETAP system considers characteristics of each deliverable handed at the end of each SDLC milestone, namely product characteristics. It keeps tracking all team activity measures (TAM) during SDLC, namely process characteristics. Lately, Petkovic et al. [5] utilized data collected by the SETAP system to generate predictive models that evaluate software projects developed by student teamwork in software engineering class. The key objective was to enhance the early prediction of software engineering project failure. Further elaboration of SETAP collected data is provided in section 3.1.

SETAP system allowed for some joint software engineering projects [24]. Consequently, there were two types of software engineering teams, i.e., Local and global. While the former refers to the teamwork whose participants are affiliated to the same academic institution, the latter refers to the teamwork whose participants are affiliated to different academic institutions. In this study, we compare prediction models for each group of teamwork. In [46], the authors investigate the possibility of explaining random forest results by feature reduction. They maintain an explainability method which was previously applied to biomedical data [47]. They used statistical measurements, i.e., average and standard deviation, of k-top features to stress differences among local and global teamwork distributions. This study investigates possibilities to improve the early prediction of software engineering projects by generating a distinct model for each teamwork distribution. Further, the issue of imbalanced data is considered.

3 Methodology

In order to explore any expected variation in predicting software engineering project failure due to the teamwork distribution, the SETAP dataset has been exploited. The proposed system architecture is shown in Figure 1. First, pre-processing activities were conducted. Second, Machine learning techniques, particularly decision trees, were applied to the pre-processed data. Third, prediction results were categorized according to the teamwork distribution.

3.1 Dataset description

This work exploits the original data set deployed by Petkovic [5,24]. It is composed of both product and process characteristics collected through the SETAP project during the fall semester of 2012 to the fall semester of 2015 for 74 distinct teams. Students' activities and behaviors are collected and evaluated according to a predefined rubric with five different millstones during the semester. Those milestones correspond to the key phases in the system development life cycle (SDLC). Further, student activities are aligned to teamwork features, as explained in [24]. Teamwork features are reproduced for eleven distinct time intervals such that instance, the first five-time intervals correspond with each of the five predefined milestones. The remaining time intervals correspond to teamwork behaviors alongside different aggregation of the first five-time intervals. The purpose of such aggregation is to record the dynamics of teamwork behavior during the project development life cycle. A detailed description of the data set is provided by [5].

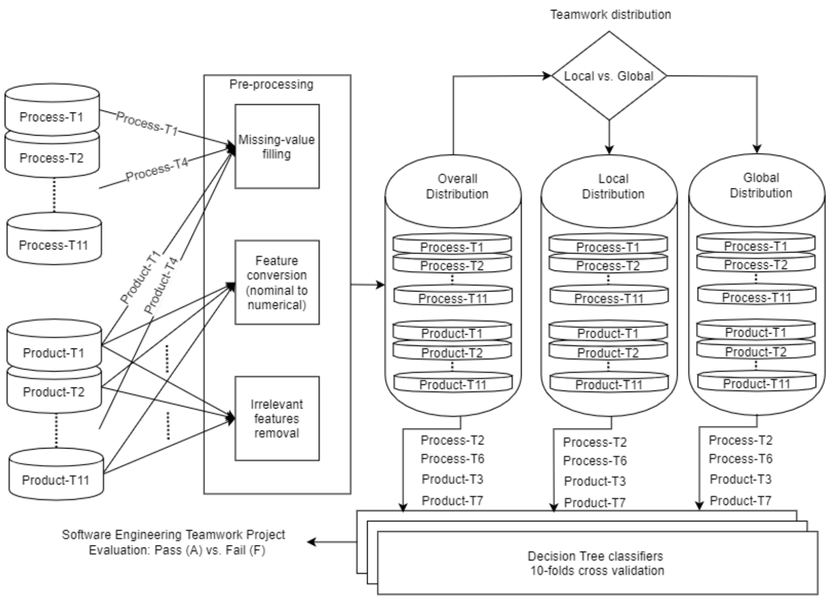


Fig. 1. System architecture

In order to gain the full advantage of the SETAP data, some pre-processing have been incurred. Table 1 compares the characteristics of the original data available at the UCI repository [48] and the pre-processed version. First, some features listed in the product files have been removed. Those features are descriptive, have nothing to do with teamwork characteristics. Those features are year, semester, timeInterval, teamNumber, and semesterId. Second, teamDistribution and teamLeadGender features in the software files have been transformed into binary numerical format instead of nominal ones similar to the product files. For timeDistribution feature, zero value expresses local teamwork. One-value expresses global teamwork. For teamLeadGender characteristic, zero-value used to express teamwork lead by male participants. One-value used to express teamwork lead by female participants. Third, missing data has been pre-processed. Some teams missed to submit certain deliverables at specific time stamps.

Table 1. Dataset pre-processing

Criteria	**Original dataset [5]**		**Pre-processed dataset**	
	Process	***Product***	***Process***	***Product***
Time intervals	11	11	11	11
Number of features	84	115	84	110
Missing data	T1, T4	T1, T4	-	-
Nominal features	-	2	-	-

3.2 Prediction models

Decision tree classifier is used to build classification models to predict software engineering project failure [49]. 10-folds cross-validation method was the evaluation method [50]. Weka data mining toolbox maintained in all experiments [51].

3.3 Data imbalance

A potential shortcoming accompany academic data is the suffer from imbalanced data, which means that data may be distributed unevenly according to the target variable. In the case of predicting software engineering project failure, usually, the majority class consists of the successfully passed projects. Failed projects form the minority class. Section 4.3.4 investigates the opportunities to improve the early prediction of software engineering project's failure by handling the issue of imbalanced data. The synthetic minority over-sampling technique (SMOTE) is employed as the re-sampling approach [52].

4 Experiments and Results

Experiment setups and evaluation metrics are presented first. Subsequently, experiments and corresponding results are presented. First, modeling of software engineering projects for distinct teamwork distribution is considered. Second, early prediction of software engineering project failure is emphasized. Third, a better understanding of teamwork dynamics is sought. Forth a better early prediction of software engineering project failure is explored by handling the issue of imbalanced data. Further, threats to validity are presented.

4.1 Experimental setup

In order to investigate prediction results for software engineering project failure concerning teamwork distribution, the dataset was divided into two categories according to the teamwork distribution characteristic. Additionally, the aggregate set of all projects regardless of teamwork distribution characteristic was maintained to serve as a baseline reference. Table 2 shows configurations of the three distributions of the dataset against the process and product perspectives. The local category limited to projects performed by local teamwork members who are colleges at the same academic institution. The global category is limited to projects performed by distributed teamwork members who are not studying at the same academic institution. The overall category includes both local and global teamwork distributions. Local and global categories are mutually exclusive.

Table 2. Distributions of dataset

Distribution	**Process**		**Product**	
	Passed (A)	***Failed (F)***	***Passed (A)***	***Failed (F)***
Local	43	16	34	25
Global	6	9	8	7
Overall	49	25	42	32

Prediction models were built for each category of teamwork distribution separately. Prediction results were investigated at particular time intervals to facilitate the failure prediction of software engineering project development as early as possible. Second and sixth-time intervals were considered to predict project failure in terms of process perspective. Third and seventh-time intervals were considered to predict project failure in terms of product perspective. In all experiments, J48, Weka's decision tree implementation, is maintained to train and test prediction models. Only J48's MinNumObj parameter was tuned while other parameters kept unchanged. For each time interval, thirty distinct models have been constructed with different values of MinNumObj, ranging from 1 to 30. Only models attaining the best results are reported below.

4.2 Evaluation metrics

Accuracy is the most preferred statistical metric to evaluate prediction models. It is calculated as the ratio of correctly predicted instances to the overall considered set of test cases. In software engineering teamwork project prediction modeling, Accuracy refers to the amount of correctly predicted projects, both passed (i.e., A-labeled) and failed projects (i.e., F-labeled) compared to the overall considered projects. Binary-class prediction models, commonly known as classification models, are better elaborated using the confusion matrix, a visual representation of classification results [53]. The key benefit of this representation is to differentiate errors in prediction according to the type of misclassification error. Usually, in binary classification models, one class is treated as a positive class, while the other is considered negative. Consequently, the confusion matrix differentiates two types of errors. False Positives (FP) and False Negatives (FN). While the former refers to negative instances that are wrongly classified as positive. The latter refers to positive instances that are wrongly classified as negative. Table 3 illustrates the organization of the classification results in the Confusion Matrix.

Table 3. Confusion Matrix

		Predicted evaluation of SETP	
		Failed (F)	***Passed (A)***
Actual evaluation of SETP	Failed (F)	TP	FN
	Passed (P)	FP	TN

In alignment with predicting failed software engineering project, the set of failed Software Engineering Teamwork Projects (SETP), i.e., F-labeled, is considered the positive class. While, the set of passed SETP, i.e., A-labeled, is considered as the negative class. Consequently, FP refers to successfully passed projects that are wrongly classified as failed ones. Similarly, FN refers to failed projects that are wrongly classified as passed ones. Interestingly, Accuracy is better described in terms of quantities given by the confusion matrix, as shown in Eq. 1. Furthermore, several performance metrics are derived from those quantities. Substantially, Precision, Recall, alternatively Sensitivity, and F-measure metrics are the most commonly used evaluation metrics for classification models [53]. These metrics are described in Eq. 2, Eq. 3, and Eq. 4, respectively.

$$\text{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN} \times 100\% \tag{1}$$

$$\text{Precision Rate} = \frac{TP}{TP + FP} \tag{2}$$

$$\text{Recall Rate} = \frac{TP}{TP + FN} \tag{3}$$

$$\text{F - Measure Rate} = 2 \times \frac{precision \times recall}{precision + recall} \tag{4}$$

Precision metric captures degradation in predicting positive class due to FP error, i.e., misclassifying a passed project as a failed one. Recall metric captures degradation in predicting positive class due to FN error, i.e., misclassifying a failed project as a passed one. F-measure generally represents the harmonic mean of both Precision and Recall metrics. Therefore, it is maintained to measure the overall performance of prediction models. Whereas, recall metric is maintained to evaluate completeness of prediction models in terms of predicting failed SETP.

4.3 Experiment results

The result of software engineering project failure prediction are presented here from two perspectives. First, process perspective, second, product perspective. Earlier perspective emphasizes teamwork behaviors leveraging specially prepared e-learning environment, namely SETAP. The latter perspective concerned with the quality of the submitted deliverable. Due to concern with early prediction of project failure, reported results with respect to the process perspective are limited to the second and sixth-time intervals. These two intervals are assumed to be adequate to capture the degree of collaboration and commitment among team members. Teamwork members are assumed to prepare detailed requirements and specifications by the end of these two-time intervals. With respect to the product perspective, reported results are limited to the third and seventh-time intervals. Prior to these time intervals, teamwork deliverables are a kind of documentation and design blueprints. The first software deliverable appears at the third-time interval.

Software engineering project development modeling: Considering process characteristics, prediction models from the perspective of the software engineering project development process were built and tested. Tables 4 and 5 show the prediction rates among the local, global, and overall teamwork distributions for the second-time interval, ProcessT2, and sixth-time interval, ProcessT6, respectively. Results are reported in terms of the confusion matrix and the accuracy rate. Based on the second-time interval, ProcessT2, results, prediction models of local and overall teamwork distribution perform better in predicting successfully passed projects compared to failed ones with 83.7% and 79.6% prediction rates, respectively. On the contrary, global teamwork distribution performs better in predicting failed projects compared to the successfully passed ones with a 78.0% prediction rate. The local teamwork distribution outperforms global teamwork distribution in modeling software engineering project development in terms of Accuracy with a 77.9% accuracy rate.

In light of the sixth-time interval, ProcessT6, results, both local and overall teamwork distributions performed well in predicting successfully passed projects compared to the failed projects with 83.7% and 75.5% prediction rates respectively. Similar to the second-time interval, ProcessT2, results, global teamwork distribution was the worst in modeling successfully completed projects with a 16.6% prediction rate. In general, the prediction model of local teamwork distribution was the best in predicting software engineering project development in terms of Accuracy with a 79.6% accuracy rate. Interestingly, this result is quite better than the result obtained by considering the second-time interval, ProcessT2. Whereas, global teamwork distribution results in terms of Accuracy degraded quite significantly when considering the sixth-time interval, ProcessT6, with more than 13 absolute percentage points decrease.

Table 4. Results of Software Engineering Project Modeling: Process Perspective, second-time interval (*ProcessT2*)

Dataset	**Local**		**Global**		**Overall**	
Predicted Actual	***A***	***F***	***A***	***F***	***A***	***F***
A	36	7	3	3	39	10
F	6	10	2	7	8	17
Accuracy	77.9%		66.6%		75.6%	

Table 5. Results of Software Engineering Project Modeling: Process Perspective, sixth-time interval (*ProcessT6*)

Dataset	**Local**		**Global**		**Overall**	
Predicted Actual	***A***	***F***	***A***	***F***	***A***	***F***
A	36	7	1	5	37	12
F	5	11	2	7	7	18
Accuracy	79.6%		53.3%		74.3%	

Considering product characteristics, prediction models from the perspective of software engineering project final product, i.e., deliverable at each considered time interval, were built and tested. Tables 6 and 7 show the prediction rates among the local, global, and overall teamwork distributions for the third-time interval, ProductT3, and seventh-time intervals, ProductT7, respectively. Results are reported in terms of the confusion matrix and the accuracy rate. Based on the results of the third-time interval, ProductT3, the prediction of global teamwork distribution performed better in predicting successfully passed projects compared to the failed ones with a 62.5% prediction rate. Prediction models of local and overall teamwork distributions were better in predicting failed projects compared to the successfully passed ones with 68.0% and 65.6% prediction rates, respectively. The local teamwork distribution outperformed global teamwork distribution in modeling software engineering projects in terms of accuracy with a 62.7% accuracy rate.

In light of the seventh-time interval, ProductT7, results, all three teamwork distributions performed better in predicting failed projects compared to the successfully passed projects with 72%, 71.4%, and 71.87% prediction rates for local, global, and overall distributions respectively. The local teamwork distribution outperformed the global distribution in predicting successfully passed projects with more than 20.0 absolute percentage points. In general, the prediction model of local teamwork distribution was the best in predicting software engineering project development in terms of accuracy with a 71.1% accuracy rate. Interestingly, this result is significantly better than the result obtained by considering the third-time interval, ProductT3, with improvement exceeds 8.0 absolute percentage points. Global teamwork distribution retained the same accuracy rate among both third and seventh-time intervals, ProductT3 and productT7, which was 60.0%.

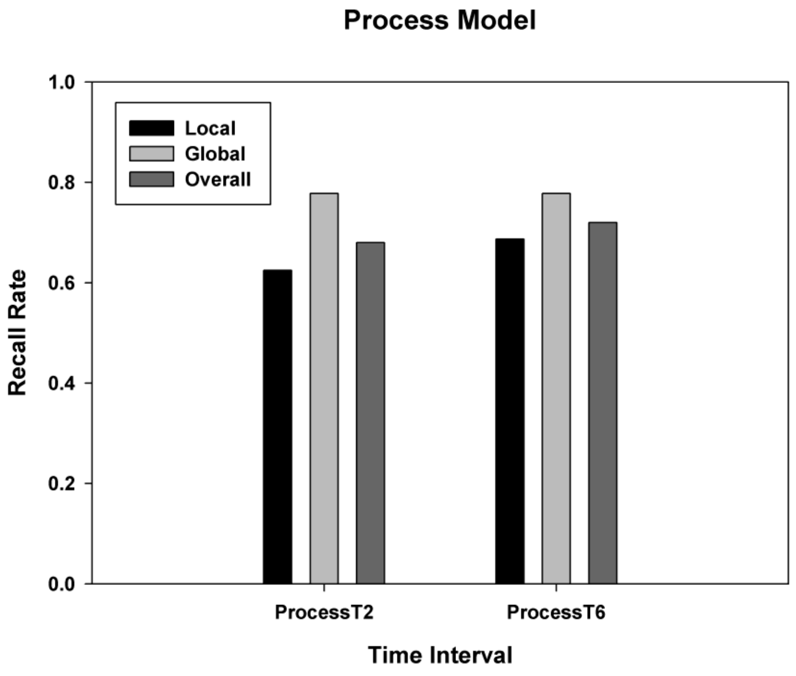
Table 6. Results of Software Engineering Project Modeling: Product perspective, third-time interval (*ProductT3*)

Dataset	**Local**		**Global**		**Overall**	
Predicted Actual	***A***	***F***	***A***	***F***	***A***	***F***
A	20	14	5	3	25	17
F	8	17	3	4	11	21
Accuracy	62.7%		60.0%		62.1%	

Table 7. Results of Software Engineering Project Modeling: Product perspective, seventh-time interval (*ProductT7*)

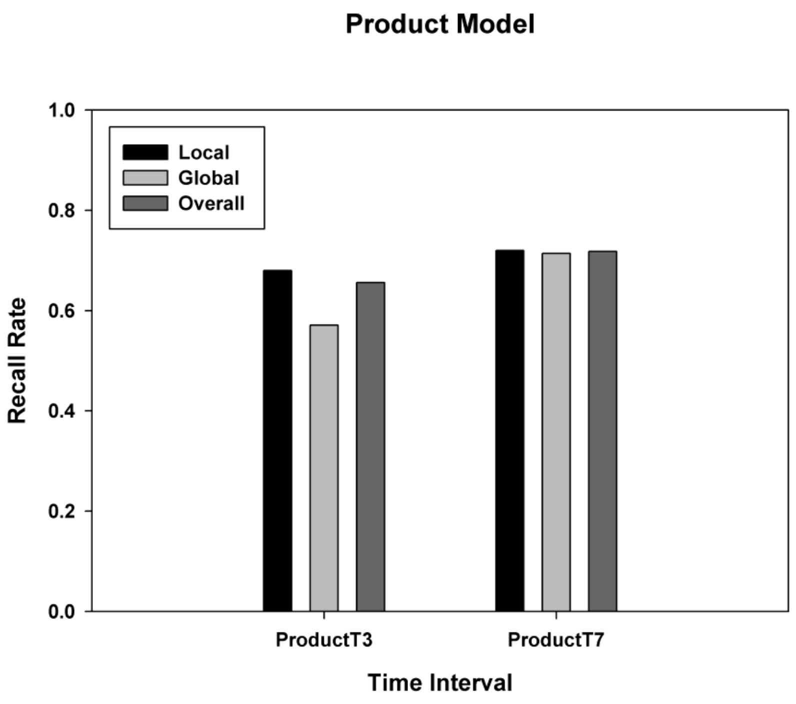
Dataset	**Local**		**Global**		**Overall**	
Predicted Actual	***A***	***F***	***A***	***F***	***A***	***F***
A	24	10	4	4	28	14
F	7	18	2	5	9	23
Accuracy	71.1%		60.0%		68.9%	

Software engineering project failure prediction: To evaluate prediction models in terms of predicting software engineering project failure, attained results in terms of recall metric are reported here. Figure 2 compares the results of prediction models in terms of recall rate from both process and product perspectives. Considering process perspective, prediction models of global teamwork distribution outperformed prediction models of local teamwork distribution for both second and sixth-time intervals, ProcessT2 and ProcessT6, respectively. From a product perspective, while prediction models of local teamwork distribution outperformed the prediction model of global teamwork distribution significantly for the third-time interval, ProductT3. Both distributions attain quite similar rates of recall metric for the seventh-time interval, ProductT7.



(a)

Software engineering project teamwork dynamics analysis: Teamwork dynamics is a crucial aspect of software engineering teams. Since software engineering projects span for a fairly long period, at least one academic semester for academic projects. Monitoring teamwork dynamics through the life cycle of software engineering project development is a good indicator of teamwork members collaboration and harmony. Teamwork dynamics are highly relevant to the process perspective of software engineering development. Consequently, process prediction models of the sixth-time interval, ProcessT6, are compared to process prediction models of the second-time interval, ProcessT2. While the second-time interval, ProcessT2, statistics are limited to the teamwork behaviors during the project's second milestone. The sixth-time interval, ProcessT6, records accumulate teamwork behaviors alongside the project's first and second milestone time spans.



(b)

Fig. 2. Software engineering project failure prediction in terms of recall rate of prediction model (a) Process models considering second and sixth-time intervals (*ProcessT2* and *ProcessT6*); (b) Product models considering third and seventh-time intervals (*ProductT3* and *ProductT7*)

Figure 3 Compares aggregate and individual time intervals performance in modeling software engineering projects against process characteristics in terms of F-Measure metric. Interestingly, results exhibit adverse behaviors of local teamwork distribution and global one. While local teamwork distribution attained a higher F-Measure rate at ProcessT6, Global teamwork distribution made a higher F-Measure rate at ProcessT2. Former results indicate a better teamwork communication at the early stage of project development by local teamwork members. Latter results indicate energized teamwork dynamics at the latter stages of project development by global teamwork members. This improvement among global teamwork distribution is intuitive due to communication and cultural barriers at the early stages of software development.

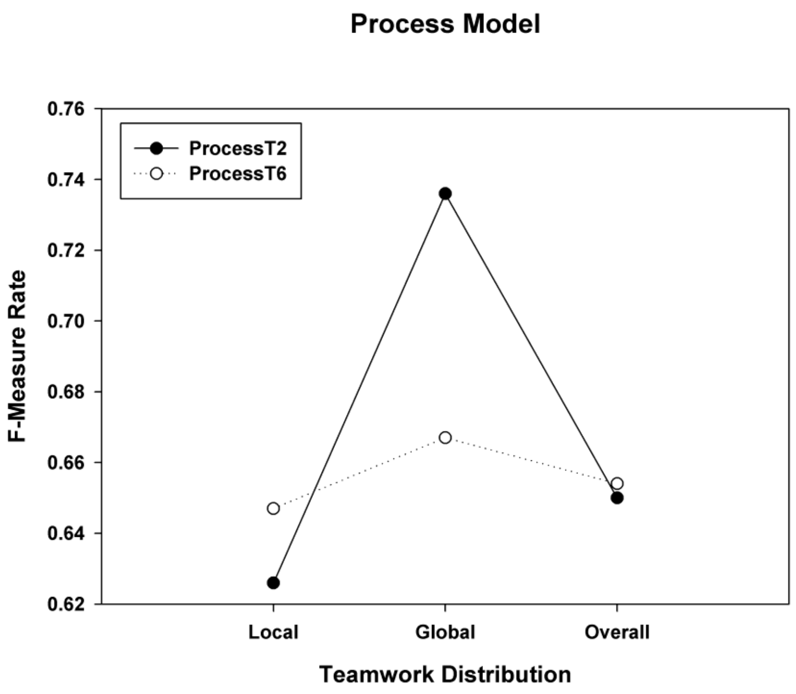


Fig. 3. Software engineering project teamwork dynamics considering process perspective second and sixth-time intervals (*ProcessT2* vs. *ProcessT6*)

Imbalanced dataset: Considering the maintained distributions of software engineering projects, i.e., Table 2. The only distribution that exhibits data imbalance is the local distribution from a process perspective. Such that the failed project represents the minority set with 0.27% of the whole distribution. This class was augmented by 27 synthetically made instances to match the number of the successfully passed projects. Both second and sixth-time interval data exhibit significant improvements after re-sampling. The sixth-time interval, i.e., ProcessT6, gained the best performance. Figure 4 compares the prediction rates of failed software engineering projects before and after employing SMOTE re-sampling to ProcessT6 local distribution. Results are compared in terms of both recall rate and F-measure rate metrics.

Threats to validity: The key issue that makes the experimental results of this research questionable is the shortage of real-world data. The major concern raised by this issue is the opportunity to generalize obtained results using a relatively short data set. I.e., while the complete data set deals with 74 distinct teamwork, global distribution restricted to 15 distinct teamwork only. New trends in collaborative education, e.g., joint-campuses, and joint programs, are expected to tackle this issue in the next few years. COVID-19 pandemic caused an unprecedented shift toward the e-learning system among both educators and learners. E-learning and MOOCs platforms witnessed a massive surge in activities and behaviors. A considerable amount of real-world data is expected to be made available by the end of the 2019-2020 academic year, which will inevitably include software engineering resources, as IT students are the first to employ e-learning technologies and practices.

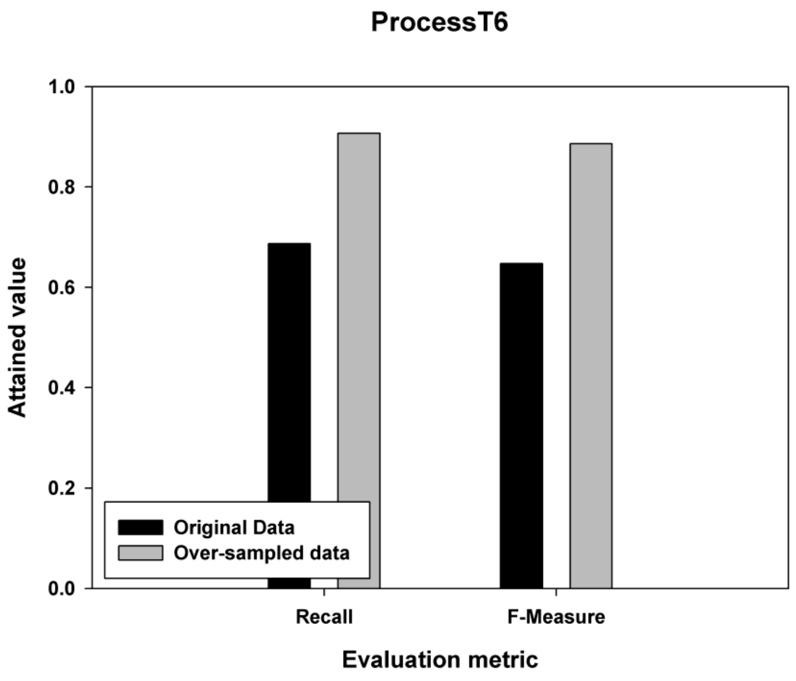


Fig. 4. Comparison between locally developed software engineering projects before and after instance re-sampling using SMOTE approach [52] considering process sixth-time interval (*ProcessT6*)

5 Discussion and Implications

This section discusses the results attained in Section 4. First, a relative improvement is noticed in the results attained by lateral and aggregate time intervals compared to the earlier and singleton time intervals. Such improvement is reasonable since these intervals convey a better understanding of teamwork behaviors. Further, aggregate time intervals, i.e., sixth to eleventh intervals, have an extra advantage, which is the ability to predict the dynamics of student behaviors among different time intervals. As extra and successive time intervals considered, results are expected to improve. However, as the essential purpose of this work is the early prediction of software engineering project failure, experiments are restricted to the early stages of the development. From processing characteristics perspective, analysis restricted to the first two-time intervals (ProcessT1 and ProcessT2) and their aggregation, i.e., the sixth-time interval (ProcessT6). However, from the product perspective, the third-time interval (ProductT3) was considered too. Subsequently, the aggregation of the first three-time intervals, i.e., the seventh-time interval (ProductT7), was investigated.

While the first two-time intervals cover requirement analysis restrictively, the first product implementation is delivered at the third-time interval.

Imbalance between successfully passed and failed projects is expected to generate overfitting models, i.e., a significant bias in prediction results towards the prediction of the successfully passed projects at the expense of losing prediction of failed software engineering projects. Techniques of instances re-sampling successfully mitigating this issue.

Inconsistent project grading between process perspective and product perspective may cause some wondering. However, it is a very common case in software engineering. Usually, students tend to do better in the theoretical part, i.e., process perspective, compared to the technical solution, i.e., product perspective. Instructors tend to form teamwork to guarantee equal progress in both perspectives. Nevertheless, teamwork may exhibit superior performance in one perspective compared to the other. It depends on the knowledge levels and practical competencies of teamwork members.

Findings obtained by this research are assumed to improve the measurement of learner's competency levels. While this study studies student's remote behavior based on prediction models, quite recent studies investigate the possibility to measure levels of learner attention utilizing an engineering-based brain computer interface [34-36]. Combining both approaches is presumably promising to achieve better learner's knowledge and competency levels measurement.

6 Conclusion

In the last years, learning and education systems have primarily stressed the importance of employing proper e-learning systems for better educational process management and monitoring. This paper investigates the importance of treating software engineering projects differently depending on the teamwork distribution. Two particular arrangements of teamwork distributions are considered. Those are local and global teamwork. The former is limited to teamwork participants who are affiliated to the same academic institution. The latter refers to the teamwork, whose participants are affiliated to different academic institutions. Attained experiment results support the stated proposal from three different perspectives. First, teamwork project modeling. A noticeable difference in predicting final project evaluation observed between locally, and globally developed software engineering projects approaches 11 and 26 absolute percentage points in terms of accuracy from process and product perspectives, respectively. Second, early prediction of teamwork project failure. While globally developed projects outperform locally developed ones from a process perspective, locally developed projects outperform globally developed ones from a product perspective. Third, monitoring of teamwork dynamics. Interestingly, globally developed projects exhibit a significant improvement in teamwork dynamics in the latter project phases compared to the locally developed projects.

7 References

[1] Adriano, C. M. (2019). Microtasking Software Failure Resolution: Early Results. ACM SIGSOFT Software Engineering Notes, 44(1), 36–36. https://doi.org/10.1145/3310013.3310016

[2] Dingsøyr, T., Bjørnson, F. O., Moe, N. B., Rolland, K., & Seim, E. A. (2018). Rethinking coordination in large-scale software development. Proceedings of the 11th International Workshop on Cooperative and Human Aspects of Software Engineering - CHASE 18. https://doi.org/10.1145/3195836.3195850

[3] Lindsjørn, Y., Sjøberg, D. I., Dingsøyr, T., Bergersen, G. R., & Dybå, T. (2016). Teamwork quality and project success in software development: A survey of agile development teams. Journal of Systems and Software, 122, 274–286. https://doi.org/10.1016/j.jss.2016.09.028

[4] Raibulet, C., & Fontana, F. A. (2018). Collaborative and teamwork software development in an undergraduate software engineering course. Journal of Systems and Software, 144, 409–422. https://doi.org/10.1016/j.jss.2018.07.010

[5] Petkovic, D., Sosnick-Perez, M., Okada, K., Todtenhoefer, R., Huang, S., Miglani, N., & Vigil, A. (2016). Using the random forest classifier to assess and predict student learning of Software Engineering Teamwork. 2016 IEEE Frontiers in Education Conference (FIE). https://doi.org/10.1109/fie.2016.7757406

[6] Straub, J., & Whalen, D. (2013). An Assessment of Educational Benefits from the OpenOrbiter Space Program. Education Sciences, 3(3), 259–278. https://doi.org/10.3390/educsci3030259

[7] Shafiq, H., Wani, Z. A., Mahajan, I. M., & Qadri, U. (2017). Courses beyond borders: A case study of MOOC platform Coursera. Library Philosophy and Practice, 1-15.

[8] World Health Organization. (2020). Coronavirus disease 2019 (COVID-19): situation report, 72.

[9] Zhang, W., Wang, Y., Yang, L., & Wang, C. (2020). Suspending Classes Without Stopping Learning: China's Education Emergency Management Policy in the COVID-19 Outbreak. Journal of Risk and Financial Management, 13(3), 55. https://doi.org/10.3390/jrfm13030055

[10] Wang, C., Cheng, Z., Yue, X.-G., & Mcaleer, M. (2020). Risk Management of COVID-19 by Universities in China. Journal of Risk and Financial Management, 13(2), 36. https://doi.org/10.3390/jrfm13020036

[11] Gonzalez, T., Rubia, M. D. L., Hincz, K., Lopez, M. C., Subirats, L., Fort, S., & Sacha, G. M. (2020). Influence of COVID-19 confinement in students' performance in higher education. https://doi.org/10.35542/osf.io/9zuac

[12] Krathwohl, D. R., & Anderson, L. W. (2009). A taxonomy for learning, teaching, and assessing: A revision of Bloom's taxonomy of educational objectives. Longman.

[13] Boiangiu, C.-A., & Stănică, I.-C. (2019). The MOSAICS Model of Educational Approaches for Teaching the Practice of Software Project Management. Education Sciences, 9(1), 26. https://doi.org/10.3390/educsci9010026

[14] Ju, A., & Fox, A. (2018). TEAMSCOPE: measuring software engineering processes with teamwork telemetry. Proceedings of the 23rd Annual ACM Conference on Innovation and Technology in Computer Science Education - ITiCSE 2018. https://doi.org/10.1145/3197091.3197107

[15] Axinte, S.-D., Petrica, G., & Barbu, I.-D. (2017). Managing a software development project complying with PRINCE2 standard. 2017 9th International Conference on Electronics, Computers and Artificial Intelligence (ECAI). https://doi.org/10.1109/ecai.2017.8166435

[16] Liheng, H. E., Qiang, Y. A. N. G., & Qisheng, B. A. O. (2019). Construction quality evaluation system of graduation project based on the whole process. Bulletin of Surveying and Mapping, (10), 138.
[17] Yilmaz, M., Tasel, F. S., Gulec, U., & Sopaoglu, U. (2018). Towards a process management life-cycle model for graduation projects in computer engineering. Plos One, 13(11). https://doi.org/10.1371/journal.pone.0208012
[18] Naser, F. H., & Naser, M. H. (2019). Rules for Managing Methods of Implementation and Evaluation of Graduation Projects in Engineering Departments. Engineering and Technology Journal, 37(4 C), 465-474.
[19] Kennedy, I. G., & Vossen, P. H. (2017). Teamwork assessment and peerwise scoring: Combining process and product assessment. Bildungsräume 2017.
[20] Dhir, S., Kumar, D., & Singh, V. B. (2018). Success and Failure Factors that Impact on Project Implementation Using Agile Software Development Methodology. Advances in Intelligent Systems and Computing Software Engineering, 647–654. https://doi.org/10.1007/978-981-10-8848-3_62
[21] Menezes, J., Gusmão, C., & Moura, H. (2018). Risk factors in software development projects: a systematic literature review. Software Quality Journal, 27(3), 1149–1174. https://doi.org/10.1007/s11219-018-9427-5
[22] Coelho, J., & Valente, M. T. (2017). Why modern open source projects fail. Proceedings of the 2017 11th Joint Meeting on Foundations of Software Engineering - ESEC/FSE 2017. https://doi.org/10.1145/3106237.3106246
[23] Bajwa, S. S., Wang, X., Duc, A. N., & Abrahamsson, P. (2016). "Failures" to be celebrated: an analysis of major pivots of software startups. Empirical Software Engineering, 22(5), 2373–2408. https://doi.org/10.1007/s10664-016-9458-0
[24] Petkovic, D., Sosnick-Perez, M., Huang, S., Todtenhoefer, R., Okada, K., Arora, S., … Dubey, S. (2014). SETAP: Software engineering teamwork assessment and prediction using machine learning. 2014 IEEE Frontiers in Education Conference (FIE) Proceedings. https://doi.org/10.1109/fie.2014.7044199
[25] Gomede, E., Barros, R. M. D., & Mendes, L. D. S. (2020). Use of Deep Multi-Target Prediction to Identify Learning Styles. Applied Sciences, 10(5), 1756. https://doi.org/10.3390/app10051756
[26] Bogarín, A., Cerezo, R., & Romero, C. (2017). A survey on educational process mining. Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery, 8(1). https://doi.org/10.1002/widm.1230
[27] Chen, J., & Zhao, J. (2018). An Educational Data Mining Model for Supervision of Network Learning Process. International Journal of Emerging Technologies in Learning (iJET), 13(11), 67. https://doi.org/10.3991/ijet.v13i11.9599
[28] Zain, J. M., & Herawan, T. (2014). Data Mining for Education Decision Support: A Review. International Journal of Emerging Technologies in Learning (iJET), 9(6).
[29] Misut, M., & Misutova, M. (2017). Software Solution Improving Productivity and Quality for Big Volume Students Group Assessment Process. International Journal of Emerging Technologies in Learning (iJET), 12(04), 175. https://doi.org/10.3991/ijet.v12i04.6608
[30] Gütl, C. (2008). Moving towards a fully automatic knowledge assessment tool. International Journal of Emerging Technologies in Learning (iJET), 3(1).
[31] Tsortanidou, X., Karagiannidis, C., & Koumpis, A. (2017). Adaptive Educational Hypermedia Systems based on Learning Styles: The Case of Adaptation Rules. International Journal of Emerging Technologies in Learning (iJET), 12(05), 150. https://doi.org/10.3991/ijet.v12i05.6967

[32] Xie, Z. (2020). Symmetry for Multimedia-Aided Art Teaching Based on the Form of Animation Teaching Organization and Social Network. Symmetry, 12(4), 671. https://doi.org/10.3390/sym12040671
[33] Baneres, D., Baró, X., Guerrero-Roldán, A.-E., & Rodriguez, M. E. (2016). Adaptive e-Assessment System: A General Approach. International Journal of Emerging Technologies in Learning (IJET), 11(07), 16. https://doi.org/10.3991/ijet.v11i07.5888
[34] Katona, J.; Kovari, A. A Brain–Computer Interface Project Applied in Computer Engineering. IEEE Transactions on Education2016,59, 319–326. https://doi.org/10.1109/te.2016.2558163
[35] Katona, J.; K ̋ovári, A. The evaluation of BCI and PEBL-based attention tests. Acta Polytechnica Hungarica 2018,15. https://doi.org/10.12700/aph.15.3.2018.3.13.
[36] Katona, J.; K ̋ovári, A. Examining the Learning Efficiency by a Brain-Computer Interface System. Acta Polytechnica Hungarica 2018,15. https://doi.org/10.12700/aph.15.3.2018.3.14.
[37] Page, C. (2017). Software Engineering. Larsen and Keller Education.
[38] Sommerville, I. (2016). Software Engineering. 10th ed.; Pearson.
[39] Schulz, H., van Hoorn, A., & Wert, A. (2020). Reducing the maintenance effort for parameterization of representative load tests using annotations. Software Testing, Verification and Reliability, e1712. https://doi.org/10.1002/stvr.1712
[40] Garousi, V., Felderer, M., & Kılıçaslan, F. N. (2019). A survey on software testability. Information and Software Technology, 108, 35-64. https://doi.org/10.1016/j.infsof.2018.12.003
[41] Yadav, D. K., & Dutta, S. K. (2019). Test case prioritization using clustering approach for object-oriented software. International Journal of Information System Modeling and Design (IJISMD), 10(3), 92-109. https://doi.org/10.4018/ijismd.2019070106
[42] Yu, T. (2017, September). Simevo: Testing evolving multi-process software systems. In 2017 IEEE International Conference on Software Maintenance and Evolution (ICSME) (pp. 204-215). IEEE. https://doi.org/10.1109/icsme.2017.29
[43] Oguz, D., & Oguz, K. (2019). Perspectives on the Gap Between the Software Industry and the Software Engineering Education. IEEE Access, 7, 117527-117543. https://doi.org/10.1109/access.2019.2936660
[44] Frezza, S., Daniels, M., & Wilkin, A. (2019). Assessing students' IT professional values in a global project setting. ACM Transactions on Computing Education (TOCE), 19(2), 1-34. https://doi.org/10.1145/3231710
[45] Rahman, M., Watanobe, Y., & Nakamura, K. (2020). Source Code Assessment and Classification Based on Estimated Error Probability Using Attentive LSTM Language Model and Its Application in Programming Education. Applied Sciences, 10(8), 2973. https://doi.org/10.3390/app10082973
[46] Petkovic, D., Barlaskar, S. H., Yang, J., & Todtenhoefer, R. (2018, October). From Explaining How Random Forest Classifier Predicts Learning of Software Engineering Teamwork to Guidance for Educators. In 2018 IEEE Frontiers in Education Conference (FIE) (pp. 1-7). IEEE. https://doi.org/10.1109/fie.2018.8659102
[47] Petkovic, D., Altman, R. B., Wong, M., & Vigil, A. (2018). Improving the explainability of Random Forest classifier-user centered approach. In PSB (pp. 204-215). https://doi.org/10.1142/9789813235533_0019
[48] Blake, C. L., & Merz, C. J. (1998). UCI repository of machine learning databases, 1998.
[49] Safavian, S., & Landgrebe, D. (1991). A survey of decision tree classifier methodology. IEEE Transactions on Systems, Man, and Cybernetics, 21(3), 660–674. https://doi.org/10.1109/21.97458

[50] Fushiki, T. (2009). Estimation of prediction error by using K-fold cross-validation. Statistics and Computing, 21(2), 137–146. https://doi.org/10.1007/s11222-009-9153-8
[51] Hall, M., Frank, E., Holmes, G., Pfahringer, B., Reutemann, P., & Witten, I. H. (2009). The WEKA data mining software: an update. ACM SIGKDD explorations newsletter, 11(1), 10-18. https://doi.org/10.1145/1656274.1656278
[52] Chawla, N. V., Bowyer, K. W., Hall, L. O., & Kegelmeyer, W. P. (2002). SMOTE: Synthetic Minority Over-sampling Technique. Journal of Artificial Intelligence Research, 16, 321–357. https://doi.org/10.1613/jair.953
[53] Powers, D. M. (2011). Evaluation: from precision, recall and F-measure to ROC, informedness, markedness and correlation.

8 Author

Ismail Al-Taharwa is an assistant professor at the Computer Information Systems department, The University of Jordan, Aqaba Campus, Jordan. Ismail ALTaharwa received his B.Sc. degree in Computer Science and its Applications from The Hashemite University, Zarqa, Jordan, in 2005. He continued his studies at AL-Balqa’ Applied University, Al-Salt, Jordan, granted an M.Sc. degree in Computer Science in 2008. His master thesis emphasized improving mobile robotics path planning solutions leveraging Artificial Intelligence techniques, especially Computational Intelligence and Evolutionary Computations. He Awarded merit-based scholarship offered by the National Science Council of Taiwan (R.O.C.) to pursue his graduate studies at Taiwan’s top-ranked universities. Taking the valuable opportunity, ALTaharwa granted his Ph.D. in Computer Science and Information Engineering from National Taiwan University of Science and Technology (Taiwan Tech), Taipei, Taiwan (R.O.C.) 2014. He did his research in Intelligent Systems (IS) Lab under the supervision of Prof. Lee, Hahn-Ming. His Ph.D. dissertation emphasized on intelligent cybersecurity and information security solutions leveraging Machine learning techniques. His recent research interests include information security, Software Engineering, Business Intelligence, and E-learning.

Article submitted 2020-05-09. Resubmitted 2020-06-15. Final acceptance 2020-06-15. Final version published as submitted by the authors.