Scientific report

1. Project information

Project title:

Romanian:

Abordări noi bazate pe soft computing pentru evoluția software-ului: modificări si gestionarea defectelor

English:

nOvel appRoaCHEs based on Soft compuTing for softwaRe evolution: change and defect management

Code: PN-III-P1-1.1-TE2021-0892
Contract no.: 138/2022

Acronym: ORCHESTRA
https://www.cs.ubbcluj.ro/~avescan/orchestra

2. Project objectives

2.1. Objective 1. Automatic bug priority classification based on the information from the bug reports.

Activities

A 1.1. Synthesizing the most current knowledge in the field of bug priority classification;

A 1.2. Development of an algorithm using soft computing approaches for automatic bug report prioritization

A 1.3. Validation of the proposed algorithm on a set of existing bug reports

Bug tracking systems receive a large number of bugs on a daily basis. The process of maintaining the integrity of the software and producing high-quality software is challenging. The bug-sorting process is usually a manual task that can lead to human errors and be time-consuming.

The purpose of Bug reports priority classification model based on CNN is first, to conduct a literature review on the bug report priority classification approaches and secondly, to replicate existing approaches with various classifiers to extract new insights about the priority classification approaches.

To synthesize the most actual research for Bug reports classification priority prediction a Systematic Literature Review (SLR) has been conducted. We selected ten articles following the first selection procedure that have in common the study of the same issue but using different

methods of prediction and data processing. The conclusion was that all these approaches in software engineering have in common a classification of a software bug based on the textual analysis of the report, on the basis of which certain characteristics will be extracted in order to determine the priority or severity of the respective bug.

The aim of this SLR was to answer 3 questions:

1. What are the most relevant research papers in this field? Answer: Drone [TLS13] in terms of evaluation metrics, but also methods based on neural networks: [ULS18], [ULI20]
2. How should we approach this issue in software engineering and how has it been addressed in the past? Answer: All these approaches in software engineering have in common a solution of the classification problem of a software product defect based on the textual analysis of a report, on the basis of which certain characteristics will be extracted meant to determine the priority or severity of the respective bug.
3. What are the main methods studied in this context? Answer: One of the most common methods in this context is to use emotion-based features to classify bugs [ULI20],[ULS18], [TLS13] and to use machine learning techniques to classify.

We conducted experiments on CNN based approach classification model and SVM to perform a multiclass prioritization (p1-p5) of bug reports. The first step in the approach was to transform the bug reports into a vector of words by applying NLTK techniques: remove punctuation, stop word removal, lemmatization, and tokenization. To perform the mentioned methods, we used Python programming language and implemented the code from the Scikit learn library of Python and NLTK. Also in addition to that for the CNN model we applied the word2vector model using a predefined embedding layer.

The data set used for this purpose was extracted from [Bugzilla] in csv data format and preprocessed to serve as data entry for our implemented algorithms.

We operated a Naive Bayes classifier approach in order to automatically prioritize bug reports using the report information retrieved from the summary. Factors extracted from bug reports were further tokenized using Natural Language Processing [NLTK] and served as training and testing data for a Naive Bayes classifier. The first step was to extract the [Bugzilla] dataset that includes JDT project bug reports, followed by cleaning these data using [NLTK]. Furthermore, we applied the mentioned model to predict the priority class p1-p5.

We also constructed a CNN based classifier, having 3 layers and an original embedded layer constructed using the Word2Vec method. The first step was to create the embedded layer to which we will apply the other 3 layers using different activation functions. Next, we forward the output of the filter to create a one-dimensional vector. Following this step, the text sequence is transmitted to a CNN, and the model is processed. Finally, we apply a softmax layer that contains the probability distribution of the priority levels.

In addition to the classifiers implemented by us, we also evaluated these 'Bug prediction methods' using [WEKA]. This is a collection of machine learning algorithms for data mining tasks. In this extended evaluation project we applied SVM,CNN, and Naive Bayes on our Eclipse project dataset using Weka. For this WEKA process, we evaluated the performance using 4 different datasets from the same product: [ECLIPSE] but performing a cross validation under 4 different projects: JDT, CDT, PDE and Platform. The next step was to perform 4 cross-

validation experiments in Weka using (CNN, SVM and Naive Bayes) where each dataset represented the test data and each of the other 3 datasets was joined as the training data.

Given the bug reports, the performance of the proposed approach is evaluated by calculating the priority-specific precision P, the recall R, and the F1 score F1. The same evaluation metrics were used to determine the performance of our algorithms implemented: CNN and Naive Bayes. Based on these evaluation metrics, we performed an analysis on the data obtained for the 4 projects from Eclipse, but also on 3 different algorithms: SVM, CNN, and NB where we also implemented a NB and CNN algorithm using NLTK and Python libraries.

2.2. Objective 2. Code Smell investigation and software bugs.

Activities

A 2.1. Synthesizing the most current knowledge in the field of code smells, testing code smells, and the impact on errors

A 2.2. Development of a code smells detector for SQL queries;

A 2.3. Validation of the proposed algorithm for a real software system

Databases are widely used today in almost all projects. The code source is not always perfect, with no errors. But the main difference between code smells and bugs is the impact that they have on the project: bugs can determine that the app does not function properly, while code smells mean not respecting the basic concepts of the programming language.

The purpose of SQL code smells research is two-fold: firstly, to propose a tool for the automatic detection of three SQL smells and secondly, to validate the results obtained both through human evaluators and through tools comparison. In this research, the three SQL code smells being investigated are missing primary key, missing table columns, and implicit columns, which can be identified in three SQL statements, CREATE TABLE, INSERT INTO, and SELECT. These SQL code smells are supported by CoolPackSmell, our proposed SQL code smell detector.

To synthesize the most actual research studies for code smells a Systematic Literature Review (SLR) has been conducted. The SLR has been carried out from multiple perspectives [Del2017], [Tai2020], [Mie2021], [Pre2021]. There are articles that consider SQL code smells from a student's perspective, with researchers interested in finding SQL misconceptions of students. Other approaches investigate how various tools to detect multiple smells of SQL code are used in the IT industry.

The first step of the SLR methodology consists of defining keywords that define the domain. Starting with *SQL code smells*, a list of 4 keywords was created: *SQL code smells (S1)*, *SQL faults (S2)*, *SQL query bugs (S3)*, and *database schema smell (S4)*. Using all keywords, a search string was defined: *S1 or S2 or S3 or S4*. This was used to search for the articles in 4 databases: IEEE Library, ACM Library, SCOPUS, and Web of Science. Taking into account the research paper languages, which must be English, the complete article to be available online, and to describe SQL code smell from the student or programmers' perspective, the number of research papers that meet the criteria was reduced from 7053 to 15.

CoolPackSmell is a tool inspired by *DbDeo* [Sha2018] - SQL code smell detection by syntax analysis using *sqlparser*, which supports the detection of *missing primary key*, *missing columns table* and *implicit columns* SQL code smells. Note that these SQL code smells are not bugs but indicate a possible problem in the source code.

For *missing primary key* detection, the CREATE TABLE statement having a list of columns is considered. Each column is analyzed and added to a *columns_list* list. After that, each element of the

list is checked if it is the primary key (check if the PRIMARY KEY keyword is mentioned in the CREATE TABLE definition) and a list containing columns that compose the primary key is returned. So, if the result is an empty list, the *missing primary key* code smell is detected in the SQL statement.

To check if *missing columns table* code smells are present, the tool checks if the columns table is mentioned. In the beginning, a variable columns_mentioned has the default value True, - the columns of the table are mentioned. Next, each item in a list containing the elements of the parsed SQL statement, INSERT INTO, is analyzed. If a group of type Identifier is identified, only the name of the table is mentioned, without mentioning its columns (when the name of the table is accompanied by its columns, the group is of type Function). So, when columns,_mentioned is False *missing columns table* code smell is detected in the SQL statement.

For implicit columns, the code smell SELECT statement is checked. If the second element is an asterisk sign, the SQL statement contains an SQL code smell: implicit columns.

To validate the proposed approach, two empirical studies are used: empirical validation through human evaluators and empirical validation through tool comparison.

The human evaluators are 10 students from the Database program at Babes-Bolyai University. They are asked to perform two tasks: manual detection and refactoring on multiple SQL statements, and the second task is based on the result generated by CoolPackSmell: students have to refactor the SQL statements that have the SQL code smells already detected by the proposed tool.

All the students are in their second year of the master's degree in this specialization, so all the participants in the study have experience in the SQL language, acquired both in the undergraduate courses and in the graduate courses.

First, both questionnaires begin with a succinct definition of each SQL code smell. Then, each questionnaire is composed of six sections. The first section contains general data about the participant: agreement, gender, how much he/she used the SQL language during the master, and the experience with SQL code smells.

The following three sections are dedicated to containing the same questions applied to each SQL code smell from the first set of SQL statements:

- Did you identify this SQL code smell? If yes, please justify your answer (how, which steps you performed, etc),
- Based on your experience with the SQL code smell identification-based tasks, how difficult was to IDENTIFY this code smell? (1- Very easy, 2- Easy, 3- Neutral, 4- Hard, 5- Very hard),
- Based on your experience with the SQL code smell identification-based tasks, how much time did you need to IDENTIFY this SQL code smell? Time expressed in minutes.

In the last section, corresponding to the detection of missing table columns, in addition to the 3 questions listed above, a code refactoring task is added: *Now, that you identified all SQL code smells, please refactor the code (include all statements containing all SQL code smells)*, with open answer. Students must rewrite the statements in the SQL statement set so that they no longer contain SQL code smells.

The next section is dedicated to the second task, the automated detection of SQL code smells. Students receive two photos: one containing the second set of instructions and the result generated by the tool proposed in this paper. They are asked to refactor the SQL statements so as to remove the SQL code smells detected by the tool - P*lease refactor the code/statements containing SQL code smells, taking into consideration the result generated by the tool*. Students are also asked a question about inspiration from sources other than the definitions and explanations in the questionnaire, *Did you search for any other information on the internet?*.

The final section includes some statements related to the students' experience of this study.

- The domain of the subject system (SQL code smells) was perfectly clear to me. (1 - Strongly Disagree, 2 - Disagree, 3 - Neutral, 4 - Agree, 5 - Strongly Agree)

- The tasks I had to perform were perfectly clear to me. (1 - Strongly Disagree, 2 - Disagree, 3 - Neutral, 4 - Agree, 5 - Strongly Agree)
- I experienced no major difficulties in performing the tasks. (1 - Strongly Disagree, 2 - Disagree, 3 - Neutral, 4 - Agree, 5 - Strongly Agree)
- Any other comments regarding the tasks and SQL code smells.

The second study is carried out compared to the results obtained with the SQLInspect tool [Nag2018]. *SQLInspect* is also used to extract SQL statements from Java source code. However, both tools run on the same 5 projects. After the results from *SQLInspect* and *CoolPackSmell* are generated, the first step is to manually identify the SQL code smells supported by *CoolPackSmell* in the SQL statements extracted by *SQLInspect*. After the results generated by *CoolPackSmell* are validated by manual analysis, a comparison between *SQLInspect* and *CoolPackSmell* is performed for Implicit columns SQL code smell detection.

After extracting SQL statements from source code using *SQLInspect*, the manual analysis is performed to confirm that CoolPackSmell has correctly identified the SQL code smells. For all projects, CoolPackSmell has identified the same number of each type of SQL code smells that the team member responsible with this task, Alexandra Pasca, has identified.

After this, the results obtained from *CoolPackSmell* tool are compared with result generated by *SQLInspect*: can be confirmed that *CoolPackSmell* is able to detect Implicit columns SQL code smell from open source projects and to obtain the same result as the comparison tool - *SQLInspect*.

The students had to complete two questionnaires each consisting of manual and automatic detection of SQL code smells, followed by refactoring, correctly identified SQL code smells, with some claiming that the example in the automatic SQL code smell detection task helped them complete the manual detection task. From the results and responses obtained from the participants, it can be concluded that the *CoolPackSmell* tool facilitates code refactoring. Therefore, the empirical study described in this paper validates the effectiveness of the described tool.

The second part of the empirical study revealed that *CoolPackSmell* detects with accuracy SQL code smells from SQL statements extracted from Java open-source projects. Through tool comparison with *SQLInspect*, a more complex validation is used to detect Implicit columns SQL code smell. The comparison was successful, *CoolPackSmell* detecting the same SQL queries containing Implicit columns SQL code smell as *SQLInspect*.

2.3. Objective 3. Quality attributes and bug prediction

Activities

3.1.Synthesizing the most current knowledge in the field of error prediction and quality attributes;

A 3.2. Development of soft computing approaches to predict quality attribute defects (partially in 2022, partially in 2023)

In this objective, our aim is to investigate an innovative approach, namely the use of soft computing and software metrics for bug prediction and quality attributes investigation with the purpose of improving the quality of software systems, studying the relations between various quality attributes and bugs.

We have started with a replication study of a paper in the area of software defect prediction, published in the Springer journal Frontiers of Computer Science. We have started our study with a Systematic Literature Review (SLR) in which three project members were involved. Each member considered papers from one of the following databases: ACM, IEEE,

Springer. In the inclusion/exclusion criteria, for each article we have taken into consideration the proposed approach, the data sets (public or not), and the results obtained. Each of the three team members involved in this process proposed three papers for the next step of our SLR. So a total of nine papers were finally considered by each of the three members, and each member had to vote for one paper. The final decision was taken by joint agreement.

In the following, we have started to perform experiments to reproduce the results from the selected paper. We have of course tried to identify and use the same set of tools and datasets in order to best replicate the published results. We consider each results table at a time and, besides trying to replicate the results, we also investigate any possible improvements. So far, we can say that the results have been successfully reproduced as we could also significantly improve them in some cases. It remains to be seen whether these improvements are general and whether they are indeed statistically significant.

3. Research visits and collaborations

3.1. Research visit to The Netherlands

Between September 22-29, 2022, 4 members of the team (Andreea Vescan, Andreea Gabin, Alexandra Pasca and Radu Gacean) visited the collaborator prof. Alexander Serebrenik at the Eindhoven University of Technology. During the visit of the members to Eindhoven University of Technology, there were meetings and discussions both with members of the ORCHESTRA group and PhD students and professors of the Software Engineering and Technology cluster of Eindhoven University of Technology (TU/e).

Prof. Alexander Serebrenik introduced us to the Ph.D. students he coordinates, but also to his colleagues from the Software Engineering and Technology cluster. There were meetings in which TU/e members presented to grant members their research topics and the results obtained during the studies, but also meetings in which grant members presented the objectives and how they achieved their fulfillment.

3.1.1. Presentations of researchers from The Netherlands

At the first meeting, Hamid Mohayeji Nasrabadi, one of the Ph.D. Students of Prof. Alexander Serebrenik presented us his research topic regarding the Dependabot with focus on security issues. Another meeting was on the usage of logs in embedded software engineering, where Nan Yang presented us her study that involves interviews with software engineers to understand how they analyze and use the logs [Nan2022].

3.1.2. Presentations of researchers from Romania

For the first objective, on 26 September, Andreea Galbin presented her work regarding the Bug Priority classification models implemented by her. The presentation focus was based mainly on the results obtained on the CNN model and Naive Bayes to solve this software

engineering issue. Besides our team and prof. Alexander Serebrenik, Nathan Cassee, a PhD student specialized in Sentiment Analysis, attended the meeting. This concluded in suggestions on new approaches that could be used in order to perform Sentiment Analysis on the bug reports, but also some suggestions on improvements regarding the selected methods and data source.

During the grant visit, on September 28, Alexandra Pasca presented the tool implemented during the grant objectives that supports three SQL code smells: missing primary key, missing table columns, and implicit columns. Furthermore, an empirical study using human validators has been described, mentioning the work in progress for the empirical study based on comparison with another tool. Beside prof. Alexander Serebrenik, prof. George Fletcher - the leader of the Database group in Eindhoven and Daphne Miedema - PhD student in the Database group working on database education, and specifically on issues related to SQL, attended the presentation. The subject and proposed approach led to discussions about the maintenance and efficiency of the source code that contains SQL statements.

On 27 September 2022, prof. Andreea Vescan had an invited talk at the Colloquium meeting of the Software Engineering and Technology cluster of Eindhoven University of Technology. The topic presented was *Test Case Prioritization approaches in Regression Testing*. Regression testing is testing activity meant to detect and minimize the side effects of performed changes to other parts of the software and can be done following different strategies: retest all, test case prioritization, test case selection, test suite minimization, and hybrid approach. For the beginning, a systematic literature review has been performed in order to find out research publication based on test case prioritization. In the first part of the studies, contributions have been added on a test case prioritization perspective.

The based approach for test case prioritization is Genetic Algorithm (GA). Research methodology that implies three steps has been explained: data synthesis, filtering based on modified requirements, and metric computation. The proposed approach for GA identifies an optimal permutation of the active tests by generating a population of permutations and evaluates them based on the cost associated with each test (GA-cost method), the minimal number of tests from the permutation that cover all faults and all the requirements (GA-noTests method) or other more criteria: the number of useful tests, effectiveness, the efficiency (GA-TEE method). Then the process of population renewal by genetic operators is repeated, and then the best permutation(s) from the last generation is chosen.

Experiments using the GA-cost approach reported the best results in terms of costs, followed by GA-noTests and GA-TEE. When considering the number of tests required to cover all requirements and all faults, the experiments with GA-cost and GA-noTests reported close results, while GA-TEE identifies longer test schedules.

The next approach presented is test case prioritization based on Neural Network. We proposed a Test Case Prioritization based on Neural Networks Classification (TCP-NNC) approach to be further used in the test case prioritization strategy. The proposed approach incorporates among other factors the associations between requirements, tests, and discovered faults, based on which an artificial neural network is trained, in order to be able to predict priorities for new test cases. The proposal is evaluated through experiments designed on both a real and a synthetic dataset, considering two different sets of features with different neural network architectures. The metrics observed include accuracy, precision, and recall, while their

results imply that the proposed method is feasible and effective. Among the proposed models, the one with Adam optimizer and three-layered architecture is the best obtained. Statistical tests are also used to compare various proposed models from various perspectives. NN architecture, optimizer, number of used features, used dataset and validation method.

Feedback received from the TU/e members improves both approaches discussed during the visits: bug report priority classification and SQL code smells detection.

The feedback provided for Bug Priority Classification helped us to improve our approaches on bug priority classification and helped us with some ideas on cleaning our dataset and how to run the experiments.

For SQL code smell detector, the feedback helped us to better motivate our study and to evaluate it both with human validators and on open source projects, resulting in a comparison with another tool.

3.2. Research visit to Romania

Between November 7-14, 2022, the visit of Prof. Alexander Serebrenik from the Eindhoven University of Technology took place. During his visit to our faculty, there were meetings and discussions both with members of the ORCHESTRA group and in a workshop organized during this period with members of the computer science department.

Prof. dr. Serebrenik Alexander had a presentation during the organized workshop within the ORCHESTRA research grant on the 3 November 2022. The workshop organized was the aim to enforce the relationships between the Software Engineering community's members and to build a bridge with people from the connected domains. Colleagues from our department were honored to take part in this workshop, organized on the occasion of Prof. Alexander Serebrenik's visit to the Informatics Department at our faculty. This workshop was a very good opportunity to open new directions of research, Prof. Serebrenik being a prominent personality in the field of Software Engineering.

4. Events

4.1. Research Seminars

During the grant, the team had 2 seminars with four meetings (each seminar consisted of two meetings): the first two meetings have been scheduled during the research visit of the team in The Netherlands, Eindhoven (September 26 and 28, 2022); the last meetings regarding Objectives 1 and 2 have been scheduled in Romania (November 10, 2022), during the visit of prof. Alexander Serebrenik at Babes-Bolyai University.

On the topic of bug priority classification, our team and prof. Alexander Serebrenik participated in a seminar where we discussed our project on “Bug priority classification”. In the first seminar 2 members of Eindhoven University attended and we also were part of a presentation from a Ph.D. student about Sentiment Analysis. On 10 November, we presented to Prof. Alexander Serebrednik our results based on the feedback received in Eindhoven’s

seminar and the next approach that we conducted on our study. We also received feedback on the entire study and mentioned important aspects that could help in further research on this topic.

For the SQL code smells topic, the team participated in a seminar in which the tool and results obtained for the SQL code smell detector have been discussed. At the first seminar, three members of the Eindhoven University of Technology attended and Alexandra Pasca presented the tool implemented, describing also the results obtained by performing an empirical study with the human validators. On 10 November, the methodology and comparison with another tool were presented. In addition, Professor Alexander Serebrenik provided us with feedback on the entire study and mentioned important aspects that need to be mentioned in an article for this type of study.

4.2. Research Workshop

4.2.1. *Gender and Community Smells*

by Prof. Dr. Serebrenik Alexander

Alexander Serebrenik had an invited talk on gender and community smells. The presentation took place on Wednesday, 9 November 2022, 12:30 – 13:30, at the location, FSEGA Building, Room C335, Teodor Mihali Street, No. 58-60, 400591, Cluj-Napoca, Romania.

Alexander Serebrenik had an invited talk on gender and community smells. The abstract of the presentation is presented below.

> Abstract: Community smells are patterns that indicate suboptimal organization and communication of software development teams that have been shown to be related to suboptimal organization of source code. Given a long-standing association of women and communication mediation, we have conducted a series of studies relating gender diversity to community smells, as well as comparing the results of the data analysis with the perception of developers. To gain further insight into the relationship between gender and community smells, we replicate our study focusing on Brazilian software teams; indeed, culture-specific expectations on the behavior of people of different sexes might have affected the perception of the importance of gender diversity and refactoring strategies when mitigating community smells. Finally, we extend the prediction model by including variables related to national diversity and see how the interaction between national diversity and gender diversity influences the presence of community smells.
>
> This talk is based on a series of papers published in 2019-2022 and co-authored with Gemma Catolino, Filomena Ferrucci, Stefano Lambiase, Tiago Massoni, Fabio Palomba, Camila Sarmento, and Damian Andrew Tamburri.
>
> **Bio**: Alexander Serebrenik is a full professor of social software engineering at the Eindhoven University of Technology, The Netherlands. His research goal is to facilitate the evolution of software by taking into account the social aspects of software development. His work tends to involve theories and methods both from within computer science (e.g., theory of sociotechnical coordination, methods from natural language processing, machine learning) and from outside of computer science (e.g., organizational

psychology). He is actively involved in the organization of scientific conferences and is a member of the editorial board of several journals. He has won multiple best paper and distinguished reviewer awards.

4.2.2. *Learning and Applying Best Practices for Code Evolution*
by assoc. dr. Dig Danny

Danny Dig had an invited talk on adapting best practices for code evolution and learning. The presentation took place on Thursday, November 10, 2022 17:30 –18:30 (EET) online.

The abstract of the presentation is presented below.

Abstract: Our community is now at a crossroads. Despite the extraordinary rise in popularity of Python-based ML systems, they do not benefit from advances in the field of software evolution. The science and tooling for developing and evolving ML code bases are significantly behind. To reduce this gap, we conducted the first and most fine-grained study on code change patterns in a diverse corpus of 1000 top-rated ML systems. We triangulated our findings with surveys from hundreds of ML engineers and reveal four major trends in how ML engineers evolve their code. Then we designed, implemented, and evaluated a new generation of Data-driven code transformation tools that transplant the best practices for code evolution into new projects. More than 90% of the patches that we submitted to famous open-source projects like TensorFlow and PyTorch were accepted, showing the usefulness and applicability of our tools.

We launch a call to action for researchers, tool builders, ML library vendors, developers, and educators. This will inspire and equip you so that you can make a difference, with people who make a difference, at a time when it makes a difference. I will conclude with the lessons we are learning from working with industry, and invite colleagues in academia and students to join forces to make a lasting impact through our recently launched Center on Intelligence for IoT systems: http://ppicenter.org

Bio: Danny Dig is a professor of CS at the University of Colorado at Boulder and an adjunct professor at the University of Illinois at Urbana-Champaign.

He enjoys research in Software Engineering, with a focus on interactive program analysis and transformations that improve programmer productivity and software quality. He successfully pioneered the field of refactoring in cutting-edge domains including ML, mobile, concurrency and parallelism, component-based, testing, and end-user programming. He earned his Ph.D. from the University of Illinois at Urbana-Champaign where his research won the best Ph.D. dissertation award and the First Prize at the ACM Student Research Competition Grand Finals. He did a postdoctoral research at MIT.

He (co-)authored 60+ journal and conference papers that appeared in top places in SE/PL. His group's research was recognized with 9 paper awards at the flagship conferences in SE, 4 award runner-ups, and 1 most influential paper award. He received the NSF CAREER

> award, the Google Faculty Research Award (twice), and the Microsoft SE Innovation Award (twice). With his students, they released dozens of software systems. Some of these are shipping with the official release of the popular Eclipse, NetBeans, Android Studio, and Visual Studio development environments which are used daily by millions of developers. His group's research is funded by NSF, Boeing, IBM, Intel, Google, Microsoft, NEC, and Trimble. He travels all over the world to inspire and equip the next generation of technical leaders.

4.2.3. Training sessions

Training sessions were organized both online and in physical format with topics related to communication, time management, and various aspects that are important to be considered when working in a team, what makes a group to be transformed into a team.

We have attended two training sessions related to communication and team management in general.

The first training (15 September) was online and we approached topics such as public speaking, assertive communication, time management, and problem solving. The coaches are professional psychologists with extra expertise in the topics approached. In addition to some presentations using slides, the training was mostly hands-on (we played different games). Also, any theoretical concept introduced on a slide was accompanied by examples.

For the second training session (8 November), we met face to face and the main focus was on group dynamics. The theoretical concepts presented using slides were accompanied by examples and practical exercises.

The topics approached (from both sessions) are extremely useful for the team as a whole, as well as for each member individually. We have learned why teams fail (some of the reasons), about the different stages, situations, and transformations that might take place within a team from the beginning of a project until the end. We have also discussed the ultimate form of conversation, coaching, as well as about different types of communication, how to identify and deal with them and how to transition towards an assertive type of communication.

5. Project progress summary and dissemination of results

5.1. Project progress summary

Objective 6 regarding the *Project management* activities was employed using various tools to help obtain optimal results of the team during the project implementation.

The activities of the project started with a kickoff meeting that had the objective of meeting with the entire team, to discuss the activities of the project for 2021, to outline the expectations of each member, and to establish the norms in the team. Several other meetings took place, specific for each of the three objectives of 2022 of the project, with discussions on the identified problems that may interfere with the research (available dataset, applied methods and methodology, design of experiments). Alternative solutions were discussed and established with the team.

The results obtained within the project are as follows:

- related to dissemination: project webpage, workshops and presentations during research visits in The Netherlands and Romania, posters of the seminars and workshops,
- related to project management: project monitoring plan, training session on communication and time management, training session on team
- related to scientific activities: knowledge transfer sessions between group members and with external collaborators during the research visits, workshops and presentations during research visits in The Netherlands and Romania, manuscripts sent to ISI journals for two of the scientific objectives of 2022 (under review), systematic literature review results for the objective 3 and scientific experiments as replication study such that in 2023 this objective will have the results sent to an ISI journal.

5.2. Publications. Conferences and Journals.

A preliminary study for each of the two objectives was accepted to a conference as an oral presentation at the Session of Scientific Communications of Students, https://www.cs.ubbcluj.ro/scss/.

- Galbin-Nasui Andreea, Vescan Andreea, Bug reports priority classification model, 2022
- Pasca Alexandra, Vescan Andreea, Empirical study on Detecting SQL code smells, 2022

A more complex study was conducted for each of the two objectives and the results were integrated into a manuscript that was submitted to the ISI journal, that is, *Information and Software Technology* being a Q1 with IF 3.862.

- Galbin-Nasui Andreea, Vescan Andreea, Bug reports priority classification models. Replication study, ISI journal, 2022 (under review)
- Pasca Alexandra, Vescan Andreea, Empirical study on Detecting SQL code smells, ISI journal, 2022 (under review)

5.3. Dissemination of results

The Objective 7 of the project regarding *Dissemination and exploitation* was operationalized by several activities that are outlined in what follows.

Creating and maintaining the web page of the project with information regarding various meetings, visits, and events that took place under the ORCHESTRA grant umbrella.

The website contains the summary of the project, the objectives, the team members, and contacts. The webpages were updated with the presentations and workshops that took place during the visits of the researchers to The Netherlands and Romania.

The workshop meetings in Romania were disseminated on the project website, on the Faculty of Mathematics and Computer Science website, and also by email to various research

collaborators from both universities (prof. dr. Andrian Marcus from University of Texas at Dallas, prof. dr. Alexander Egyed from Johannes Kepler University, prof. dr. Alina Campan and prof. dr. Marius Truta from Northern Kentucky University) and from IT firms (EVOZON, Nagarro, Altom, Synopsys, Porsche engineering, Yonder).

As a result, the provided performance criteria were met: at least one paper submitted for each scientific objective for publication in an ISI/WoS journal with high impact factor, for the third scientific objective that is split into 2022 and 2023 years, the submission to an ISI/WoS journal will be done in 2023, research visits, knowledge transfer sessions between group members and with external collaborators. Furthermore, the project objectives for 2022 have been met and all associated activities have been completed and carried out in accordance with the project implementation plan.

6. Executive Summary of Activities

Software maintenance and evolution play an important role in obtaining a qualitative software system, either during the development phase or after release. The changes that occur during software development are one of the important factors that threaten the quality of the software system; therefore, methods and tools to achieve a better state of the system is a must.

The main objective of the project is to explore and develop an innovative set of soft computing techniques for maintenance and evolution-related activities of software with respect to software changes. In 2022 two directions were approached entirely (bug priority classification, code smells for SQL queries) and another one (quality attribute prediction) was started with the final activity in 2023.

Two research visits were employed during 2022, four researchers from the group visited in September the research team of professor Alexander Serebrenik in The Netherlands at the Eindhoven University of Technology and in November we received a visit from the same professor. Workshops were organized during the visit to Romania, with a presentation on *gender and community smells.* Another online presentation was provided by Assos. prof. Danny Dig with the title *Learning and Applying Best Practices for Code Evolution*. During the visits, several other presentations and seminars took place to disseminate the project work, the respective expertise of the two teams, and help foster potential new collaborations.

Project Director,
Assoc. prof. dr. Vescan Andreea

Bibliography

[Nan2022] Yang, N., Cuijpers, P. J. L., Hendriks, D., Schiffelers, R. R. H., Lukkien, J. J., & Serebrenik, A. (Accepted/In press). An interview study on the use of logs in embedded software engineering. Empirical Software Engineering, XX(X). https://www.win.tue.nl/~aserebre/EMSE2022Nan.pdf

[Mie2022] Daphne Miedema, Efthimia Aivaloglou, and George Fletcher. 2021. Identifying SQL Misconceptions of Novices: Findings from a Think-Aloud Study. In Proceedings of the 17th ACM Conference on International Computing Education Research (ICER 2021). Association for Computing Machinery, New York, NY, USA, 355–367. https://doi.org/10.1145/3446871.3469759

[Del2017] Delplanque, Julien & Etien, Anne & Auverlot, Olivier & Mens, Tom & Anquetil, Nicolas & Ducasse, Stéphane. (2017). CodeCritics applied to database schema: Challenges and first results. 432-436. 10.1109/SANER.2017.7884648.

[Tai2022] T. Taipalus, "Explaining Causes Behind SQL Query Formulation Errors," 2020 IEEE Frontiers in Education Conference (FIE), 2020, pp. 1-9, doi: 10.1109/FIE44824.2020.9274114.

[Pre2021] Presler-Marshall, Kai & Heckman, Sarah & Stolee, Kathryn. (2021). SQLRepair: Identifying and Repairing Mistakes in Student-Authored SQL Queries. 199-210. 10.1109/ICSE-SEET52601.2021.00030.

[Sha2018] T. Sharma, M. Fragkoulis, S. Rizou, M. Bruntink, and D. Spinellis, "Smelly Relations: Measuring and Understanding Database Schema Quality," *2018 IEEE/ACM 40th International Conference on Software Engineering: Software Engineering in Practice Track (ICSE-SEIP)*, 2018, pp. 55-64.

[Nag2018] C. Nagy and A. Cleve, "SQLInspect: A Static Analyzer to Inspect Database Usage in Java Applications," *2018 IEEE/ACM 40th International Conference on Software Engineering: Companion (ICSE-Companion)*, 2018, pp. 93-96.

[Bug20] Bugzilla -> url=https://www.bugzilla.org/, 2020. Online; accessed May 2020.

[Ecl22] Eclipse. url = https://www.eclipse.rg/, 2022. Online; accessed,May 2022

[NLTK] Nltk. url = https://www.nltk.org/, note = Online; accessed May 2022, year=2022,

[ULS18] Qasim Umer, Hui Liu, and Yasir Sultan. Emotion based automated priority prediction for bug reports.IEEE Access, 6:35743-35752, 2018

[ULI20] Qasim Umer, Hui Liu, and Inam Illahi. Cnn-based automatic prioritization of bug reports. IEEE Transactions on Reliability,69(4):1341–1354, 2020.

[TLS13] Yuan Tian, David Lo, and Chengnian Sun. Drone: Predicting priority of reported bugs by multi-factor analysis. In 2013 IEEE International Conference on Software Maintenance, pages 200–209,2013.