come a long way, creating models that can help significantly in planning for future migration flows, but more research is needed to create more all-encompassing models.

REFERENCES:

1. Bijak, J., Disney, G., Findlay, A. M., Forster, J. J., Smith, P. W., & Wiśniowski, A. (2019). Assessing time series models for forecasting international migration: Lessons from the United Kingdom. Journal of Forecasting, 38(5), 470-487.

2. Fuchs, J., Söhnlein, D., & Vanella, P. (2021). Migration Forecasting—Significance and Approaches. Encyclopedia, 1(3), 689-709.

3. Skjerpen, T., & Tønnessen, M. (2021). Using future age profiles to improve immigration projections. Population Studies, 75(2), 255-267.

4. Nair, R., Madsen, B. S., Lassen, H., Baduk, S., Nagarajan, S., Mogensen, L. H., ... & Urbak, S. (2020). A machine learning approach to scenario analysis and forecasting of mixed migration. IBM Journal of Research and Development, 64(1/2), 7-1.

Ihor Polataiko

Vasyl Stefanyk Precarpathian National University,

Ivano-Frankivsk, Ukraine

igor.bogdanovich39@gmail.com

BUILDING OF EFFICIENT APPLICATION SOFTWARE SYSTEMS FOR EDUCATIONAL INDUSTRY BY STUDENTS ENGINEERING TEAMS

This paper presents the development processes of a student engineering organization called PNUdev, which functions at Vasyl Stefanyk Precarpathian National University.

Keywords: *organization, management, processes, student groups, training, education, sofware engineering.*

To grow successful software engineers, educational institutions, that teach software engineering-related fields should provide students not only with theoretical knowledge and practical assignments but also with a significant amount of production-close development experience. Having an environment, where students with very little or without industry experience could be able to practice and gain required skills is essential. Also, institutions, usually, have some category of students, which are typically in their senior years at the institution, who has some good amount of industry experience (1 or 2 years) and would like to teach and mentor more junior students. This type of experience could be incredibly useful for the mentioned category of students because it gives them the ability to better structure their own knowledge while teaching other students, and receive hands-on experience of technical leading of the projects. Apart from that, most educational institutions have an information technology (IT) department, that manages and develops software for internal needs.

Students engineering organization, called PNUdev, was created and currently functions at Vasyl Stefanyk Precarpathian National University.

Multiple studies were conducted in the field of innovative approaches to software engineering education and project management processes. Paper [1] describes an application of short-term projects in cross-cultural environments for improving the soft and technical skills of the students. In papers [2], [3] researches related to increasing the level of student motivation and engagement in project-based learning are described. Papers [4], [5], focus on the development of real-world software by students as a vital aspect of engineering education. The research described in the paper [6] development management process, based on reflective weekly monitoring, for the context of student projects. Finally, studies [7] and [8] are related to using code review as a learning tool for software engineering students. Also, the studies [9], [10] were using parts of the software products, developed by PNUdev, as supportive tools.

During the research on the optimal development process two aspects were considered: delivering results in accordance to the timelines and ensuring the quality, both external and internal, of the resulting software update.

According to the proposed development process, development should be performed in one-week iterations. This iteration interval was defined as optimal for short-term projects. The first iteration starts with adding common implementation of common parts of the project: data model creation or update, creation of data transfer objects, and other common elements, which can be identified based on the architectural template. At the beginning of the iteration, the project technical lead should conduct a meeting where he or she would create the technical tasks, from the features, that are currently in development, time-estimate and discuss them with the team, and assign them to the team members. Tasks from one feature, most of the time, will be assigned to one person, during the development iteration to allow the members of the team to work in parallel efficiently. The number of features in development should be more or less equal to the number of developers on the project. The project technical lead has to track the progress of features implementation and ensure, that features are fully developed in a gradual manner, considering the original time estimations for the features, so the project timelines will be met successfully. To ensure the external quality, and, also, to allow project team members to have a deeper involvement in the project, it was recognized, that the best practice is to assign responsibility for the correct feature implementation to the members of the team. This role is called feature manager. Ideally, all the team members should perform this role in the project for different features. Feature managers should not take part in the implementation of the tasks related to the feature, for which he or she is the feature manager. The responsibility of the feature manager is to understand the requirements for the feature and to perform the acceptance testing of each task implementation, related to the feature. Also, at the planning meeting at the beginning of the development iteration, the feature manager should provide the technical lead with the list of all the requirements that are not implemented or not fully implemented for the feature, assigned to him, if this feature is, currently, in the development state. In this way, the feature manager is completely responsible for assuring the external quality and full completion of the feature. Apart from that, automated tests can be created to further ensure external and improve the internal quality of the system. To ensure the internal

quality of the product update, made in scope of the project, and, also, to provide useful experience and needed feedback for the project team members, all the results of the implementation of the technical tasks should be presented in form of merge requests and reviewed carefully by at least one core team member and one or couple of members of the project team. The code review from the core team member is done to ensure the internal quality of the changes, whereas reviews from the project team members are more focused on their learning from the experience of code reviews. Static code analysis tools and architecture tests might be also used to help ensure internal quality. For the Java ecosystem, ArchUnit, PMD, SonarQube, and others can be used.

General cycle of the task is following:

- Creation by technical lead.
- Implementation and testing by the developer.
- Running of the automated tests and automatic static code analysis.
- Acceptance by the feature manager.
- Code review.
- Merging of the code changes to the main code base and automatic deployment to a staging environment.

In the last week or two of the project development phase, the testing part is accomplished, when all team members would exchange the features, for which they were feature managers during the development, and would perform the acceptance testing for the features. Also, testing of the features from previous projects, related to the product, might happen in this phase. Usually, all the functional defects are fixed within a week. In case, some big discrepancy is found, project duration might be extended by the amount of time, required to implement needed updates. Usually, it takes no more than two weeks, for the projects of the considered size.

The research was performed by means of evaluating the experience of creation and leading of the PNUdev organization, experimentation, and path of trial and error. The organization functions for about 2 years, at the time of writing this paper. During this time, 5 software products were worked on. Some of them continue to evolve in the current projects. A couple of the training projects were conducted, where participants

were mostly students of the 2nd course. 80% of students found their first IT job in under 6 months, after completing the training project, and, part of them, stayed with the organization, where, till this time, they become members of the core team. The core team extended from 3 people to 10. Apart from the technical growth and production software creation, the organization establishes a perfect environment for learning, knowledge sharing, and networking for all the participants of the organization, beyond the organization projects.

REFERENCES

1. H. Akerlund, G. Audemard, H. Bollaert, V. Hayenne-Cuvillon, A. Hlobaz, M. Kozlenko, P. Milczarski, J.C. Monteiro, J. Morais, D. O'Reilly, P. Possemiers, and Z. Stawska, "Project GGULIVRR: generic game for ubiquitous learning in interactive virtual and real realities," in EDULEARN20 Proceedings of the 12th International Conference on Education and New Learning Technologies, July 6-7, 2020, pp. 5973-5979, doi: 10.21125/edulearn.2020.1566.

2. P. Morais, M. J. Ferreira and B. Veloso, "Improving Student Engagement With Project-Based Learning: A Case Study in Software Engineering," in IEEE Revista Iberoamericana de Tecnologias del Aprendizaje, vol. 16, no. 1, pp. 21-28, Feb. 2021, doi: 10.1109/RITA.2021.3052677.

3. R. K. Pucher, A. Mense, H. Wahl and F. Schmöllebeck, "Intrinsic motivation of students in project based learning," in Transactions of the South African Institute of Electrical Engineers, vol. 94, no. 3, pp. 6-9, Sept. 2003.

4. R. J. Machado, P. Guerreiro, E. Johnston, M. Delimar and M. A. Brito, "Work in progress - IEEEXtreme: From a student competition to the promotion of real-world programming education," 2009 39th IEEE Frontiers in Education Conference, 2009, pp. 1-2, doi: 10.1109/FIE.2009.5350540.

5. Buhari, S. Valloo and H. Hashim, "A Streamlined Approach to Enhance the Capacity of Undergraduate IT Students to Deliver High Quality and Demand-Driven Final Year Project: A Conceptual Framework on Collaboration between Industry and

University," 2017 7th World Engineering Education Forum (WEEF), 2017, pp. 910-914, doi: 10.1109/WEEF.2017.8467126.

6. M. Marques, S. F. Ochoa, M. C. Bastarrica and F. J. Gutierrez, "Enhancing the Student Learning Experience in Software Engineering Project Courses," in IEEE Transactions on Education, vol. 61, no. 1, pp. 63-73, Feb. 2018, doi: 10.1109/TE.2017.2742989.

7. Z. Kubincová and I. Csicsolová, "Code Review in High School Programming," 2018 17th International Conference on Information Technology Based Higher Education and Training (ITHET), 2018, pp. 1-4, doi: 10.1109/ITHET.2018.8424617.

8. T. Brown, M. R. Narasareddygari, M. Singh and G. Walia, "Using Peer Code Review to Support Pedagogy in an Introductory Computer Programming Course," 2019 IEEE Frontiers in Education Conference (FIE), 2019, pp. 1-7, doi: 10.1109/FIE43999.2019.9028509.

9. Lazarovych et al., "Software Implemented Enhanced Efficiency BPSK Demodulator Based on Perceptron Model with Randomization," 2021 IEEE 3rd Ukraine Conference on Electrical and Computer Engineering (UKRCON), 2021, pp. 221-225, doi: 10.1109/UKRCON53503.2021.9575458.

10. M. Kozlenko, O. Zamikhovska, V. Tkachuk and L. Zamikhovskyi, "Deep Learning Based Fault Detection of Natural Gas Pumping Unit," 2021 IEEE 12th International Conference on Electronics and Information Technologies (ELIT), 2021, pp. 71-75, doi: 10.1109/ELIT53502.2021.9501066.