Looking for Motivation. How to Keep Students' Software Projects from Ending Up on the Shelf?

Teresa Zawadzka
Faculty of Electronics, Telecommunications, and Informatics, Gdansk University of Technology
Gdansk, Poland *tegra@eti.pg.edu.pl*

Michał Zawadzki
Faculty of Electronics, Telecommunications, and Informatics, Gdansk University of Technology
Gdansk, Poland *michal.zawadzki@pg.edu.pl*

Agnieszka Landowska
Faculty of Electronics, Telecommunications, and Informatics, Gdansk University of Technology
Gdansk, Poland *nailie@eti.pg.edu.pl*

Abstract

IT specialists in the business environment work in teams according to the established methodology and using the established toolkit. From the university's point of view, preparing IT students to work in such an environment is a challenging task, as it requires either cooperation with business or the simulation of similar conditions in the university environment. Participation of students in real projects can provide them with the necessary practical skills. The aim of this paper is to present the experience gained in running real-life, long-term projects in academia, and to provide guidelines on how to involve students in running these projects to the benefit of students.

Keywords: teaching, IT, capstone project, teamwork, IS

1. Introduction

Capstone projects [11, 2] in computer science are an important element of students' educational path [5]. Their main aim is to teach students teamwork and to familiarise them with issues related to solving complex technical problems. They are conducted at various universities, still, many of the capstone projects are of limited scope and far from realistic. The expected outcomes of these projects are mainly PoCs (Proof of Concept) or prototypes, while making the product actually usable or deployment is described as future work never to be done. In rare cases, the outcome is a usable product, but still not published nor deployed to the final environment - just another thesis on the shelf. If one aims at the development of more complex or ready-to-use IT solutions, it usually is beyond the scope of a single capstone project. However, even if the expected outcomes should be production-ready or a minimal valuable product (MVP), capstone projects might contribute to these solutions - it can be a new functionality added or an improvement of the existing one made. It is expected that the outcomes of such projects are no longer prototypes, but working software that can be later deployed or developed by other students. In such a situation, capstone projects are treated as sub-projects of real-life, long-term projects called here **metaprojects**.

This paper aims at exploration of how real-life metaprojects can be conducted in academia and how to make students to contribute to them for the benefit of the students, the educators,

and perhaps also for external beneficiaries.

2. Related Work

The studies that are most relevant to our research are those that deal with project-based learning and the organization of so-called capstone courses [17]. The literature survey study found that the majority of such courses last 1 semester and intend to work for a client, but usually, no external client is involved. Moreover, students are expected to deliver only a proof of concept [17]. The review [5] points out that most studies focus on the challenges of group formation and evaluation, rather than on how to conduct such projects. Another review of over 500 undergraduate computer science projects found that among the four critical success factors for such a project are: the origin/idea of the project and the motivation of the students [14]. Among studies that deal with the project organization, there is a study on version control and how to support this activity in student projects [8], lifecycle of the project using agile approach [15], skills and process insight [9], planning and project difficulty as a risk factor [18], evaluation of the team projects [16, 3, 10], role rotation and document transfer [13], risk framework application to help students [4]. There are also works [1, 6, 7, 12] which analyse students' motivations showing that many students want to be involved in something beneficial for the others. Some of the studies reported here mentioned student projects that were part of a bigger, real-life projects, but none of them explored them in detail, focusing rather on sub-projects.

3. Research Methods

In the first step, we have analysed remarks made by students in capstone/teamwork projects' evaluation questionnaires. Then, a series of unstructured interviews was performed with the projects' supervisors and with IT business representatives. Based on those we have identified a set of requirements for capstone project outcomes (provided in Table 1).

In the next step, the two metaprojects that did very well in terms of the requirements, were analysed to look for the best practices and lessons learned. The first metaproject Graph Representation Integrating Signals for Emotion Recognition and Analysis (GRISERA) [19] is rooted in the field of affective computing. It concerns the development of the framework that provides the solution to create, share, and integrate data from different affect-related experiments in a way that provides unified access to datasets needed to build AI emotion recognition solutions. The second metaproject named Friendly Applications was established in 2014 and since then develops mobile applications for autism therapy. A family of applications was developed and launched for Google Play that supports both a child with autism and its caregiver. This is a real-life project and many children and therapists benefited from it already.

In the last step, through the analysis of lessons learned from the metaprojects, we identified a set of good practices, formulated in the form of guidelines.

4. Guidelines for leading metaprojects

The presented guidelines (see Table 2) are meant to support academic metaprojects in terms of student motivation and achievement of the expected educational outcomes, providing also a usable result. It is worth emphasizing that they should not be regarded as a sure recipe for the success of every metaproject, but rather as good practices to be adapted to a specific project. We found out, that tn many cases, guidelines, rules, and procedures from the project management in IT industry cannot be directly migrated to the academic environment, as it has its own specificity: (1) The work has to be broken down into chunks that can be fitted as capstone projects, engineering projects, or master theses. The range of these sub-projects, and methodology applied must guarantee that specific sub-projects do not interfere uncontrollably with the other sub-projects conducted in parallel. (2) Each sub-project must be tailored to specific, usually

Table 1. What we want the students to learn through capstone projects

Category	Expected results
Methodological knowledge	Students can work in the settled framework for project management.
	Students can play or at least practically understand the specific roles depending on the project management framework.
Toolkit	Student can work with the mainstream tools and strategies for code versioning and management.
	Students can work with the mainstream tools and strategies for issue tracking and management.
Testing and evaluation	Students can design, implement, and execute various-level tests within continuous development and deployment.
	Students have practical experience with product evaluation by the end user.
Architecture and technologies	Students are able to work in the settled architecture.
	Students can adapt to the technological requirements.
Environments	Students practically understand the difference between development, testing, and production environments, their purpose and how to work within them.
	Students practically understand and can apply modularisation, virtualisation, and containerization at least at the basic level.

variable requirements (such as the number of people 1-5, time frame - 1 or 2 semesters, etc.), as these projects are elements of the teaching path and must be conducted according to the settled rules. (3) The project supervisor is often the only person, who is engaged in the project longer than an academic year, so the exchange of people in the project is great. (4) Often there is no support for DevOps, technical, and managerial tasks. The educators must organize these aspects of the project by themselves. More detailed guidelines are provided in Table 2. Please note, that behind every guideline there is a lesson we had to learn the hard way.

5. Conclusions

The paper shows that conducting real-life metaprojects in academia, with usable outcomes, is possible, although challenging. We defined the expected educational results, concerning methodological knowledge, toolkit, testing, architecture, technologies, and environments and we observe an in-depth skills and insight being a benefit for the students involved. The analysis of our lessons learned has led to formulation of guidelines on how to conduct projects keeping students' software solutions from ending up on the shelf and providing them with the needed teamwork and engineering competence. The authors hope that their experience and gathered guidelines can help others to lead capstone projects that don't end up on the shelf.

The authors are aware that the list of guidelines is not complete and there are some issues that were addressed, but the direct solution was not given (e.g. how to guarantee the engagement of at least two educators, with one best with business experience). Still, the authors believe that the issues raised here are important and may become a voice in the discussion, on how to conduct capstone projects as sub-projects of metaproject at universities to make them beneficial for students, for educators, and maybe also real-life users.

Table 2. Guidelines for conducting academic projects.

Category	No.	Guideline
Metaproject methodology	1.	Define goals and milestones of the metaproject and align sub-projects with them.
	2.	Involve at least 2 educators, with wide technical knowledge and knowledge concerning teamwork issues to play different roles (eg. product owner, scrum master).
	3.	Involve at least one educator with business experience or search for external expert support.
	4.	Make the educator responsible for setting metaproject environment and toolkit (including DevOps tasks).
	5.	Set the licensing method at the very beginning and clearly communicate it to all stakeholders (incl. code and promotional materials).
Student project methodology	6.	Prepare and update an on-boarding procedure for newcomer students.
	7.	Make knowledge sharing a part of an off-boarding procedure for students leaving the metaproject.
	8.	Work according to agile methodology.
	9.	Make students' basic theoretical knowledge of agile projects a prerequisite for entering the project.
	10.	Monitor students' work weekly or biweekly.
	11.	Define Master thesis diplomas rather than engineering projects for more advanced tasks.
Toolkit	12.	Apply tools for code versioning, issue tracking, and metaproject knowledge repository.
	13.	Ensure that repositories or designs created in the supporting/external tools are owned and managed by an educator involved in the project at least by the end of student project.
	14.	Analyze requirements, limitations, and licensing before setting metaproject toolkit for code versioning, issue tracking, and knowledge repository.
Testing and evaluation	15.	Ensure students plan and perform tests on various levels, consider continuous development/continuous integration paradigm.
	16.	Have well-defined acceptance criteria (definition-of-done) and end each increment with acceptance tests/user evaluation.
Architecture and technologies	17.	Choose architecture and technology stack that allows for easy separation of sub-projects between groups.
	18.	Make centralized decisions on the technological stack, the environments, between-release consistency, and between-app consistency.
Environments	19.	Have multiple environments for development, testing, integration, and production defined.
	20.	Use modularisation, virtualisation, and containerization techniques for sub-projects to ensure that the outcomes work together as a product of the metaproject.

References

[1] Buckley, M., Kershner, H., Schindler, K., Alphonce, C., and Braswell, J.: Benefits of using socially-relevant projects in computer science and engineering education. In: *SIGCSE Bull.* 36.1 (Mar. 2004), pp. 482–486.

[2] Burge, J. E. and Gannod, G. C.: Dimensions for Categorizing Capstone Projects. In: *2009 22nd Conference on Software Engineering Education and Training*. 2009, pp. 166–173.

[3] Domínguez, C., Jaime, A., García-Izquierdo, F. J., and Olarte, J. J.: Factors Considered in the Assessment of Computer Science Engineering Capstone Projects and Their Influence on Discrepancies Between Assessors. In: *ACM Trans. Comput. Educ.* 20.2 (Mar. 2020).

[4] Kirk, D., Luxton-Reilly, A., Tempero, E., Crow, T., Denny, P., Fowler, A., Hooper, S., Meads, A., Shakil, A., Singh, P., Sutherland, C., Tsai, Y.-C. V., and Wuensche, B.: Educator Experiences of Low Overhead Student Project Risk Management. In: *26th Australasian Computing Education Conference*. Sydney, Australia: ACM, 2024, pp. 58–67.

[5] Kokkoniemi, M. and Isomöttönen, V.: A systematic mapping study on group work research in computing education projects. In: *JJ. Syst. Softw.* 204 (2023), p. 111795.

[6] Layman, L., Williams, L., and Slaten, K.: Note to self: make assignments meaningful. In: *Proceedings of the 38th SIGCSE Technical Symposium on Computer Science Education*. SIGCSE '07. Covington, Kentucky, USA: ACM, 2007, pp. 459–463.

[7] Liesaputra, V. and Ott, C.: Detecting Students' Affective States in Industry-focused Projects. In: *Proceedings of the 20th Koli Calling International Conference on Computing Education Research*. Koli Calling '20. Koli, Finland: ACM, 2020.

[8] Milentijevic, I., Ciric, V., and Vojinovic, O.: Version control in project-based learning. In: *Computers Education* 50.4 (2008), pp. 1331–1338.

[9] Motogna., S., Suciu., D. M., and Molnar., A.: Investigating Student Insight in Software Engineering Team Projects. In: *16th International Conference on Evaluation of Novel Approaches to Software Engineering*. INSTICC. SciTePress, 2021, pp. 362–371.

[10] Olarte, J. J., Domínguez, C., Jaime, A., and García-Izquierdo, F. J.: Student and Staff Perceptions of Key Aspects of Computer Science Engineering Capstone Projects. In: *IEEE Transactions on Education* 59.1 (2016), pp. 45–51.

[11] Ott, C. and Liesaputra, V.: Industry-Focused Projects in an Intense One-Year ICT Programme. In: *Proceedings of the Twenty-Second Australasian Computing Education Conference*. ACE'20. Melbourne, VIC, Australia: ACM, 2020, pp. 132–141.

[12] Ott, C. and Liesaputra, V.: Using Affective Learning Analytics in Industry-focused Projects: Experiences and Challenges. In: *Proceedings of the 24th Australasian Computing Education Conference*. ACE '22. Australia: ACM, 2022, pp. 153–162.

[13] Pérez, B. and Rubio, Á. L.: A Project-Based Learning Approach for Enhancing Learning Skills and Motivation in Software Engineering. In: *Proceedings of the 51st ACM Technical Symposium on Computer Science Education*. SIGCSE '20. Portland, OR, USA: Association for Computing Machinery, 2020, pp. 309–315.

[14] Pucher, R. and Lehner, M.: Project Based Learning in Computer Science – A Review of More than 500 Projects. In: *Procedia - Social and Behavioral Sciences* 29 (2011). The 2nd Int. Conf. on Education and Educational Psychology 2011, pp. 1561–1566.

[15] Subramaniam, S., Chua, F.-F., and Chan, G.-Y.: Project-based Learning for Software Engineering–An Implementation Framework. In: *Journal of Telecommunication, Electronic and Computer Engineering (JTEC)* 9.3-4 (Oct. 2017), pp. 81–85.

[16] Tafliovich, A., Petersen, A., and Campbell, J.: On the Evaluation of Student Team Software Development Projects. In: *Proc. of the 46th ACM Technical Symposium on Computer Science Education*. SIGCSE '15. Kansas City, USA: ACM, 2015, pp. 494–499.

[17] Tenhunen, S., Männistö, T., Luukkainen, M., and Ihantola, P.: A systematic literature review of capstone courses in software engineering. In: *Information and Software Technology* 159 (2023), p. 107191.

[18] Wikstrand, G. and Borstler, J.: Success Factors for Team Project Courses. In: *19th Conference on Software Engineering Education Training (CSEET'06)*. 2006, pp. 95–102.

[19] Zawadzka, T., Wierciński, T., Meller, G., Rock, M., Zwierzycki, R., and Wróbel, M. R.: Graph Representation Integrating Signals for Emotion Recognition and Analysis. In: *Sensors* 21.12 (2021).