Improving a Model-Based Software Engineering Capstone Course

Michael J. May[1][0000−0003−4571−7972] and Amir Tomer[1][0000−0002−8387−2458]

Kinneret College on the Sea of Galilee, Jordan Valley, Israel 15132
{mjmay,tomera}@mx.kinneret.ac.il

Abstract. Capstone projects are a common feature of software engineering bachelor's degrees. We report on the experience and lessons learned from a decade of capstone projects at a small regional Israeli college. We first created a capstone process adapted to the department's model-based software design philosophy, cultural aspects of the student body, and the sparse industrial environment surrounding the college. After several years, we improved the process through the introduction of mandatory fill-in report templates. Analyses of ten years of project statistics and outcomes led us to an understanding of what capstone features led to better outcomes and how the report templates affected grading outcomes. Our templates are released under the Creative Commons Attribution-ShareAlike 4.0 International License.

Keywords: Software design engineering · Student assessment · Software engineering education · Capstone projects

1 Introduction

The Israeli Council for Higher Education (CHE) requires that all certified four-year software engineering degree programs include a capstone project, but leaves management details up to the institution [4]. Having managed software engineering capstone projects at a regional Israeli college for ten years and received numerous outstanding project awards at the national level, we amassed an archive of project reports, project metadata, and institutional knowledge of how to produce successful capstones. We performed a reflective qualitative and statistical analysis of our results, summarized here. We sought to answer three research questions about our capstone process:

1. What factors in the project team, customer, or topic can lead to improved evaluation (*i.e.,* grades) from academic and industrial advisors?
2. Did the introduction of the fill-in document templates improve project outcome and grades?
3. What did students think about their advisor, team, and project documents (with or without templates)?

We present two contributions related to our experience that ought to be of interest to the community.

First, we share our conclusions from the research questions and the process we used to reach them. They give guidance about what led to project success using our capstone process.

Second, we detail the design and motivation behind the set of fill-in templates for capstone project documentation that helped improve our process. All of the templates we describe can be accessed at the projects URL listed at the end of the paper and are released under the Creative Commons Attribution-ShareAlike 4.0 International License.

While the use of templates for software engineering is well known (see MIL-STD-498 DIDs (Data Item Descriptions) [13] and the IEEE Software Developers Toolkit), our templates improve on previous work in two major aspects:

1. They are tailor-made, *i.e.,* generated specifically to our institution's capstone process.
2. Unlike other templates that only describe the content of the documents and give guidelines, we provide color-coded text, detailed instructions, and formatting styles.

2 Methods

Our method to answer the research questions was influenced by the history of the capstone process in our department.

2.1 Initial Capstone Project Process

The capstone course began to operate in the 2009–10 academic year. It was a full year course (2 semesters) performed in the final year of the degree program.

We primarily sought projects at industrial customers, preferably high tech or software companies. We also allowed research-supporting capstones in which the customer was a researcher who needed software for research purposes. We did not allow theoretical or exploratory capstone projects that do not produce a concrete deliverable system.

Whether industrial or research-supporting, we required an active industrial advisor who met with the students weekly or biweekly (without the academic advisor). For research-supporting topics, the researcher took the role of the industrial advisor. The industrial advisor's job was to decide the requirements, guide planning and design, and ensure that development proceeded. Each customer chose the development methodology to be used (*e.g.,* agile methods, scrum, waterfall) and managed the team accordingly. We did not allow topics without an advisor who is the customer of or a significant stakeholder in the project's results. That

decision was meant to ensure that projects were guided steadily and that the advisor had a stake in the project's success.

Each capstone project was assigned an academic advisor whose job was to offer guidance on theoretical and technical matters, resolve team member conflicts, and ensure projects proceed in a timely manner.

The students' primary task was to create the working capstone deliverables. The deliverables were typically software systems that the project customer could use as part of its business needs.

Since a system without documentation is not maintainable, we also required them to create a set of documents that support the deliverables. To fulfill the model based design philosophy, we designed a set of three documents: the project charter, the specification and design, and the final report. Their contents and due dates[1] are summarized in Table 1. To ensure nothing was forgotten, we prepared a checklist of elements for all documents with required sections, technical required figures (*e.g.,* use case diagram), and typographic specifications.

The **charter document** is a managerial and planning document. It reflects the original plans for the system and is submitted early in the project lifecycle, after the topic and basic requirements are decided.

The **specification and design** is a technical document. It was designed to be built incrementally or in accordance with the chosen development methodology (*e.g.,* as user stories are reached or as modules are designed). It reflected the system as delivered.

The **final report** is a summary and reflective document. It summarized the project as performed from managerial and technical perspectives. Students wrote about the problems they faced in execution, what changes were made during the course of the project, and provided an after-the-fact timeline (Gantt).

Evaluation Capstone projects were evaluated by the industrial advisor and the academic advisor by filling in advisor-specific grading forms. The final grade was calculated as the average of the two grades. The forms can be found at the projects URL listed before.

The grading criteria for the industrial advisor are divided into three parts. Part A (60%) covered planning and performance in creating the deliverables. Since personal behavior and topics were covered, each student on the team was given a separate grade. Part B (30%) covered documentation, including elements for all required documents. Part C (10%) covered the capstone presentation defense.

The grading criteria for the academic advisor were similar. Part A (60%) was an overall grade per student for the capstone's planning, complexity, execution, and

[1] The academic year for our institution runs from late October to the end of June. July and August are the spring semester final exam period.

Table 1: Project document contents and due dates.

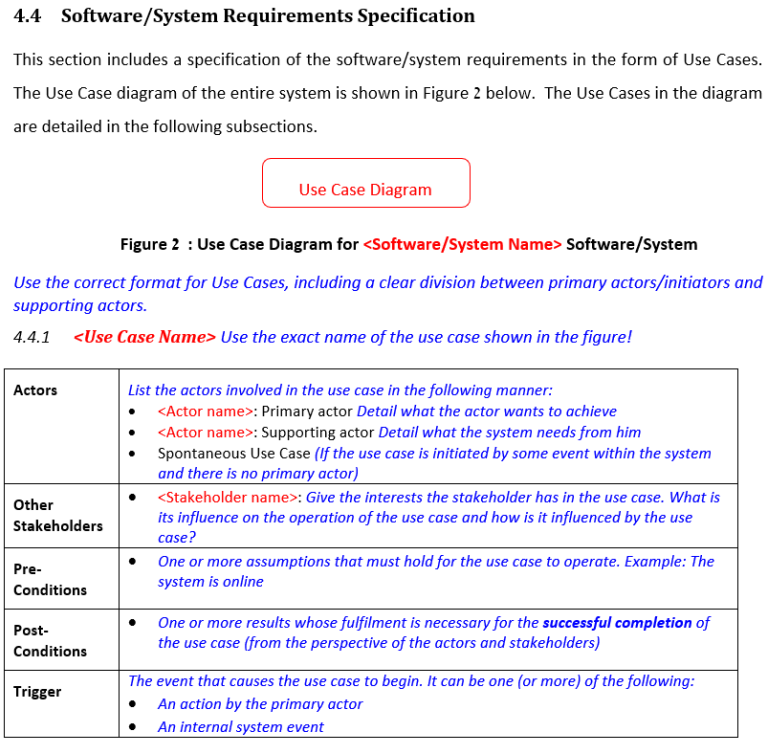
Document	Chapter	Contents
Charter	Project background	customer information, overview
(Due: 31 Dec)	Project properties	goals, deliverables, lifecycle, test plan, Gantt, stakeholders, risks
	Theoretical background	existing state, similar systems
	Technical attributes	high level req's, implementation & solution constraints, interfaces, development & implementation tools
Spec. & Design	Background	overview of project goals, background
(Due Partial: 1 Apr,	Organizational environment & business processes	customer organization, existing software state
Final: 1 Aug)	Glossary	problem domain object model, term list
	Software/system requirements	stakeholders, actors, sources, detailed req's, use case specifications
	Architectural specification	physical arch., logical arch., composite model, sequence diagrams
	Software specification	class diagrams, database diagram
	Implementation and build	developers guide
	Testing specification	tests and expected results
	Installation, use, validation	installation and maintenance manual, user manual, validation tests
Final Report (Due: 1 Aug)	Introduction	background, problems and challenges, solution approach, results, solution environment
	Theoretical background	sources, potential external customers
	Technical summary	design & impl. considerations, logical planning and execution, logical changes, screen shots, physical implementation and deployment
	Project execution	actual Gantt, hardware changes, integration & operation, test results, maintenance, support
	Self-evaluation, reflection	personal thoughts
	Conclusion	personal thoughts
	Second language project brief	brief in an alternate language (usually English in Hebrew documents)
	Appendices	code snippets, meeting summaries, extra tables

integration. Parts B (30%) and C (10%) for the academic advisor grade were identical to the industrial advisor's.

2.2 Process update: Introduction of fill-in templates

In 2014, we decided to improve the course by introducing the use of fill-in Microsoft Word document templates for the project reports. The templates gave the students a clear framework for their requirements, design, models, and tests, removing uncertainty about how to structure their first long documentary task. Initially, the templates were optional, but following positive feedback from students we made them mandatory in 2016.

The templates included in-place instructions with color-coded boilerplate, examples, instructions, and fields. A sample of the templates is shown in Figure 1.



4.4 Software/System Requirements Specification

This section includes a specification of the software/system requirements in the form of Use Cases. The Use Case diagram of the entire system is shown in Figure 2 below. The Use Cases in the diagram are detailed in the following subsections.

Use Case Diagram

Figure 2 : Use Case Diagram for <Software/System Name> Software/System

Use the correct format for Use Cases, including a clear division between primary actors/initiators and supporting actors.

4.4.1 ***<Use Case Name>*** *Use the exact name of the use case shown in the figure!*

Actors	*List the actors involved in the use case in the following manner:* • <Actor name>: Primary actor *Detail what the actor wants to achieve* • <Actor name>: Supporting actor *Detail what the system needs from him* • Spontaneous Use Case *(If the use case is initiated by some event within the system and there is no primary actor)*
Other Stakeholders	• <Stakeholder name>: *Give the interests the stakeholder has in the use case. What is its influence on the operation of the use case and how is it influenced by the use case?*
Pre-Conditions	• *One or more assumptions that must hold for the use case to operate. Example: The system is online*
Post-Conditions	• *One or more results whose fulfilment is necessary for the* ***successful completion*** *of the use case (from the perspective of the actors and stakeholders)*
Trigger	*The event that causes the use case to begin. It can be one (or more) of the following:* • *An action by the primary actor* • *An internal system event*

9

Fig. 1: Specification & Design template snippet. The use case specification continues on the page following the one shown. Black text is boilerplate to be left in. Blue text is instructions to be removed. Red text is to be replaced with appropriate values.

To manage student progress and detect personnel issues, we introduced the requirement for each team to submit a monthly one-page progress report (also a Microsoft Word fill-in template). The report must be signed by the industrial advisor and submitted to the academic advisor. It can also be found at the projects URL listed below.

2.3 Data Collection and Analysis Methodology

Throughout the history of the capstone course, we gathered yearly performance data. At the end of each year, we collected statistics about the course's performance, including basic information on the student team, the type of topic (industrial or research supporting), the customer, the industrial advisor, and the evaluation scores. At the end of 2020, having collected ten years of data, we performed an internal review of the capstone process and the gathered data.

We analyzed the data to search for answers to the research questions. For the first question, we used Pearson's and Spearman's correlation tests to find factors that correlated numerically with grades. For categorical data, we compared the distributions using the Kolmogorov-Smirnov test. For the second question, we examined distribution of grades for projects submitted without the templates and those submitted with (again using Kolmogorov-Smirnov). For the third question, we reviewed a reflection sections from final reports submitted before and after the introduction of the templates and noted comments and sentiments expressed about the process and templates.

The data analysis was performed by the capstone course staff with the help of a data analyst within our institution who was only given access to capstone statistics, removing student identifiers and other personal information. To preserve student and advisor privacy, only group statistics are reported in this work.

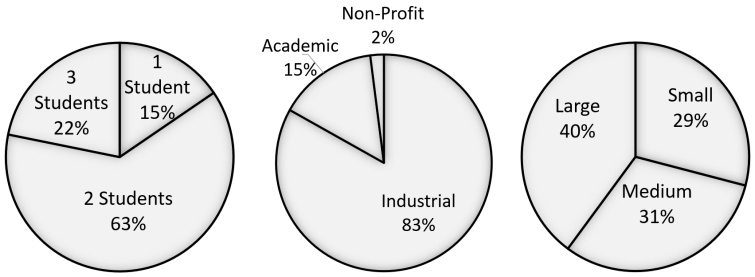
3 Results

For background, we first assembled a series of population statistics of the 2010–2020 projects and student teams, which we present first. We then present the results from the statistics analyses that we use to answer the research questions.

3.1 Capstone project population and topics

A total of 148 students completed 80 capstone projects. Team size (Figure 2(a)) was predominantly two, but about one third were lone students or teams of three.

Capstones were done for 38 distinct customers (Figure 2(b)), with the majority done for industries. Customers were about evenly distributed between small ($\leq$ 10 employees), medium (11 to 99 employees), and large ($\geq$ 100 employees) (Figure 2(c)).



(a) Team size (b) Customer type (c) Customer size

Fig. 2: Capstone project population distribution. Percentages reflect the number of students who performed a capstone in the team size, customer type, or customer size shown.

Table 2: Selection of capstone projects 2010–2020.

Customer	Capstone Topics
Industrial	Airline web services interface customer sandbox
	Remote fault management dashboard
	Water meter testing tracking tool
	Custom Wireshark file analyzer
	Water desalination plant energy optimization
	Flight recommender for travel agency website
Academic	Automated attendance and class participation app
	Conference management system
	Cemetery digitization system
	Life long learning knowledge discovery website
	Interactive co-op social game
	Tutor scheduling and hours reporting website

Topics varied widely (see Table 2 for example topics), but two themes dominated. For industrial topics, most were development projects not on the customer's product critical development path. Instead, they were productivity tools, simulators, or proofs-of-concept the customer wanted, but was unable to develop due to manpower or time limitations. The results echo ones reported by [9] at a Finnish institution who reported that industries sponsored capstones for recruitment, technological exploration, and getting the software developed. For research supporting topics, capstones developed rough but useful tools to advance research efforts. They were usually standalone tools that achieved some concrete research goal.

3.2 Question 1: Factors Influencing Evaluation

Grades include per-student evaluation, so team members may receive different final scores. Analysis showed grades to be high on the whole across all years, team

Table 3: Capstone grade statistics 2010–2020

	Final Grade	Industrial Grade	Academic Grade
Mean	91.47	93.04	89.85
Median	93.00	95.00	91.00
Max	100.00	100.00	100.00
Min	70.00	60.40	65.50
Std Dev	6.08	6.95	8.03
Confidence (p<0.05)	0.98	1.12	1.29

sizes, and customer sizes (see Table 3). There were no statistically significant differences based on student sex or advisor.

Small, but significant positive correlations were found between the academic advisor grade and team size[2] (*i.e.,* larger teams got slightly higher academic advisor grades) and between academic advisor grade and customer size[3] (*i.e.,* teams at larger customers got higher academic advisor grades). This points to the college being slightly more satisfied with work performed by larger teams at larger companies. This also gives support to the departments' discouraging of single-person capstone project teams, which are more likely to work at smaller organizations with simpler requirements.

Another positive predictive correlation was found between final grades and repeat industrial customers[4]. The implication is that capstones that receive higher grades are more likely to lead to a customer returning in following years for additional capstone projects. This points to the positive reinforcement effect a good result can have on the relationship between a customer and the college. Given the small number of industrial customers geographically near our institution, each such multi-year relationship is valuable and must be maintained.

3.3 Question 2: Impact of fill-in templates

The capstone course staff agreed that the introduction of the fill-in templates led to less student questions about how to prepare the report and to greater uniformity of report quality.

We analyzed whether the introduction of the capstone template had an effect on grades. Figure 3 graphs the average capstone project grades per year. The template was introduced in 2014, but was not mandatory until 2016, so we plot the average grades per year before the introduction of the template, during the

[2] Pearson correlation test, $[Rp = 0.213, p < 0.05]$

[3] Spearman correlation test, $[Rs = 0.204, p < 0.05]$

[4] We used binary logistic regression. The regression model was found to be statistically significant ($X^2(3) = 27.288, p < 0.001$). The predictor variables (independent variables, *i.e.,* grades) explain 28.6% of the variance in the dependent variable (*i.e.,* repeat customers). Prediction success overall was 66.7%.

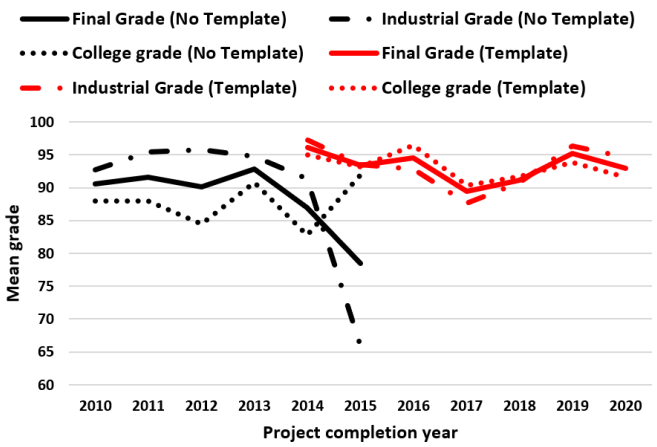


Fig. 3: Mean final capstone project grades with and without the fill-in template.

overlap, and after the template became mandatory. In 2015, only one student team chose to not use the template, so the data point is an outlier. Statistical analysis and the graphs show that the introduction of the templates led to an increase in the academic advisor grade, had no statistically significant impact on industrial grades, and closed the gap between the academic advisor and industrial advisor grades. This suggests that the templates led to the academic advisors being more satisfied with the capstone, but that the industrial advisor was more interested in the project deliverables than the reports' formatting.

3.4 Question 3: Student Attitudes

We reviewed a sample of 40 student project reflections (part of the final report), 20 from projects that used the templates and 20 from those that didn't. We marked the reflections for mentions of 3 aspects - the advisors, team dynamics, project documents - and whether the mention implied a positive, neutral, or critical attitude.

Of the 40 reflections, 36 mentioned the advisors. The majority (31, 86%) expressed a positive attitude, mostly gratitude for the energy and time they spent helping the team. A few (5, 14%) expressed critical attitudes, mostly about communication issues with the students (*e.g.,* "the advisors were disconnected from what was going on with the project"). There was no significant difference in the advisor grades between the projects with positive and critical attitudes (mean 92% positive, 91% critical, null hypothesis not rejected).

Of the 40 reflections, 20 mentioned the team. We cross referenced attitudes about the team with team size. As shown in Table 4a, the majority of reflections showed a positive or neutral attitude toward the team, but some were critical. The results clearly showed that students of larger teams tended to have more positive

Table 4: Attitudes in student reflections.

(a) Team Attitude vs Size

Team Size	Team Attitude: Critical	Neutral	Positive
1	3	0	0
2	2	5	9
3	0	0	1

(b) Document Attitude vs Template Use

Used Template?	Document Attitude: Critical	Neutral	Positive
Yes	3	0	17
No	1	0	11

attitudes about the team (χ^2 test $p < 0.01$). Positive mentions tended to be about how the students learned to divide up work (*e.g.,* "Cleanly dividing up work and planning meetings between the team make the work more efficient.") and give each other feedback (*e.g.,* "We were surprised to find that two minds working together as one made the work richer in information and reduced the chances of errors."). Critical attitudes toward the team reflected poor team chemistry (*e.g.,* "For future project developers we warmly recommend choosing an appropriate partner for doing the project."). For lone students, they were generally regret at working alone (*e.g.,* "It's better to work in pairs, the work is then divided in half and thinking will be more developed.").

Most of the reflections mentioned the documents (32). We cross referenced attitudes about the documents with template use. As shown in Table 4b, the majority of students (28, 87.5%) expressed positive attitudes, primarily about how the documents helped them remain organized and on track (*e.g.,* "Building the Charter document helped a lot to understand what the customer requires", "The Design document helped me stay organized during implementation (I implemented one use case at a time)"). Those with critical attitudes (4, 12.5%) criticized the documents as being inappropriate for agile development. In 3 of the critical reflections, the teams had received a detailed design document from the customer before the project began, so they felt the college documents were superfluous. Analysis did not show a conclusive dependency between document attitude and template use (χ^2 test $p > 0.05$, null hypothesis not rejected), so the templates may not have affected student attitudes toward the documentation.

4 Related Work

Capstone projects are common requirements in engineering departments. [6] give a taxonomy of engineering capstone courses, including a survey of common practices and a literature review relating to them. [11] present a model for evaluating the inclusion of external stakeholders in project courses in general and [7] describe results from industrial capstone projects, issues central to our model since

we directed the majority of capstone projects to external industrial customers. [5] describe two different capstone processes and their outcomes.

Software engineering capstones have been researched by many institutions. [8] and [3] describe project processes in computer science and software engineering at their institutions. [2] present a multi-institutional study of capstone projects in computing departments. They survey five institutions and the methodologies that they use in the capstone project courses. The emphasis of their work is on the integration of open source or free software in capstone projects and on the use of community-oriented topics in motivating student interest. [10] presents an industrial internship model for capstone projects used in the software engineering department of a large American institution. [1] discuss capstone project models in the computing departments in six universities on multiple continents. Each university summarizes the technical basics of the capstone course(s) at their institutions. Some of the institutions mention the use of large project teams, with one using globally distributed teams that must coordinate across time zones and continents. The context of the project (industrial or academic) and the documentation methods that each institution uses are not mentioned.

[12] presents a course in the spirit of our capstone project's process. It makes the jump from undergraduate software engineering courses to applied software management.

5 Conclusions

Capstone projects are the final step in a software engineering student's academic path at our institution. We encourage students to take on challenging projects in industrial settings that force them to work as a team, use the knowledge they acquired during their course work, and learn new technologies. Gathering data from ten years of capstones has given us insight into how project teams form, how their work progresses, and where process improvements are needed.

Our analyses found that most students preferred to work in teams and were satisfied with their project's software product and its documentation. Most students were happy with their industrial advisor's help during the project. Expressions of critical attitudes about the advisor and advisor grade were not shown to be dependent, so even students who were less than satisfied with their industrial advisor received grades similar to those who were satisfied with them. This points to the students successfully fulfilling their project obligations despite a less than ideal advisor relationship.

While the introduction of document templates for project reports was not associated with improved industrial grades or changes in student reflections, they led to improved academic outcomes and reduced headaches for students and academic advisors. We recommend their use at other institutions to formalize their documentation procedures and increase reporting uniformity and completeness.

Since Hebrew is our institution's language of instruction, the templates were created in Hebrew. Due to their success, we have translated the templates to English to make them usable by the wider community. They can be accessed at http://www2.kinneret.ac.il/mjmay/finalprojects.html.

References

1. Adams, L., Daniels, M., Goold, A., Hazzan, O., Lynch, K., Newman, I.: Challenges in teaching capstone courses. In: Proceedings of the 8th Annual Conference on Innovation and Technology in Computer Science Education. pp. 219–220. ITiCSE '03 (2003). https://doi.org/10.1145/961511.961575
2. Braught, G., Maccormick, J., Bowring, J., Burke, Q., Cutler, B., Goldschmidt, D., Krishnamoorthy, M., Turner, W., Huss-Lederman, S., Mackellar, B., Tucker, A.: A multi-institutional perspective on H/FOSS projects in the computing curriculum. ACM Trans. Comput. Educ. **18**(2), 7:1–7:31 (Jul 2018)
3. Conn, R.: A reusable, academic-strength, metrics-based software engineering process for capstone courses and projects. In: Proceedings of the 35th SIGCSE Technical Symposium on Computer Science Education. p. 492–496. SIGCSE '04 (2004)
4. Council for Higher Education: Kavim mankhim vi'hagdarot li'tokhniot limudim ba'tkhumim handasat makhshavim, handasat khashmal vi'elektronika, handasat tokhna u'mada'ey ha-makhshev - hakhlatat malag mi'yom 27.09.2016. Council for Higher Education, Jerusalem (27 Sept 2016), (Feitelson committee report)
5. Davis, H.G., Zilora, S.J.: A tale of two capstones. In: Proceedings of the 17th Annual Conference on Information Technology Education. p. 130–135. SIGITE '16, Association for Computing Machinery (2016)
6. Dutson, A.J., Todd, R.H., Magleby, S.P., Sorensen, C.D.: A review of literature on teaching engineering design through project-oriented capstone courses. Journal of Engineering Education **86**(1), 17–28 (1997)
7. Gorka, S., Miller, J.R., Howe, B.J.: Developing realistic capstone projects in conjunction with industry. In: Proceedings of the 8th ACM SIGITE Conference on Information Technology Education. SIGITE '07 (2007). https://doi.org/10.1145/1324302.1324309
8. Mohan, S., Chenoweth, S., Bohner, S.: Towards a better capstone experience. In: Proceedings of the 43rd ACM Technical Symposium on Computer Science Education. p. 111–116. SIGCSE '12 (2012). https://doi.org/10.1145/2157136.2157173
9. Paasivaara, M., Vanhanen, J., Lassenius, C.: Collaborating with industrial customers in a capstone project course: The customers' perspective. In: 2019 IEEE/ACM 41st International Conference on Software Engineering: Software Engineering Education and Training (ICSE-SEET). pp. 12–22 (2019)
10. Reichlmay, T.J.: Collaborating with industry: Strategies for an undergraduate software engineering program. In: Proceedings of the 2006 International Workshop on Summit on Software Engineering Education. pp. 13–16. SSEE '06 (2006)
11. Steghöfer, J.P., Burden, H., Hebig, R., Calikli, G., Feldt, R., Hammouda, I., Horkoff, J., Knauss, E., Liebel, G.: Involving external stakeholders in project courses. ACM Trans. Comput. Educ. **18**(2), 8:1–8:32 (Jul 2018)
12. Tomer, A.: Software mangineeringment: Teaching project management from software engineering perspective. In: 2015 IEEE Global Engineering Education Conference (EDUCON). pp. 5–11 (2015)
13. United States Department of Defense: MIL-STD-498. Standard, United States Department of Defense (Dec 1994)