Analyzing Challenges in Software Engineering Capstone Projects

Yvonne Sedelmaier, Dieter Landes
Faculty of Electrical Engineering and Informatics
Coburg University of Applied Sciences and Arts
96450 Coburg, Germany
e-mail: yvonne.sedelmaier@hs-coburg.de, dieter.landes@hs-coburg.de

***Abstract*—Engineering complex software systems is a very delicate and challenging task, which involves a variety of technical, general non-technical, and context-specific non-technical challenges. Getting better insight into the nature of these challenges is of paramount importance for aligning intended learning outcomes and didactical setup in software engineering capstone projects that aim at exercising and extending these competences. In order to obtain a fine-grained understanding of perceived challenges in capstone projects, this work presents results of a qualitative analysis of self-reports which students wrote as post-mortem documents after being part of such a capstone project. As a main contribution, the qualitative analysis substantiates results in earlier work that technical issues tend to be less challenging than non-technical ones, e.g., collaboration within the team and beyond, issues of project management and organisation, and methodological issues related to requirements engineering and effort estimation. In addition, the paper reveals challenges that might have been overlooked so far, e.g., project organisation (and not just planning), individual motivation, and individual deficiencies in setting or adhering to deadlines.**

Keywords-capstone project; software engineering; challenges; qualitative analysis.

I. INTRODUCTION

Software is a core ingredient of nearly any part of our everyday life. Software, however, requires highly skilled developers. Consequently, software engineering education plays an important role in higher education in order to acquire and exercise these skills. Traditionally, universities emphasized technical skills, such as, e.g., programming or testing skills, in software engineering education. Undoubtedly, software development requires profound technical knowledge [1]. Evidently, technical proficiency is not the only thing that matters. In recent years, it has become increasingly clear that non-technical, or soft, skills are equally important as software is developed in teams of individuals which need to interact with each other and various stakeholders such as, e.g., customers or users of their software. Software engineers need to analyze and understand complex situations and use a creative and solution-oriented approach. Various researchers emphasize the importance of non-technical skills in software engineering [2]–[6].

Software engineering requires a specific profile of competences that combines technical, general non-technical, and context-specific non-technical skills [7].

Internal surveys we conducted over the years indicate that students tend to overestimate their level of technical and non-technical competences. Many software engineering projects fail due to at least one of the following reasons: scheduling, specifications and/or average manufacturing costs [8]. Button and Sharrock [8] also state that software engineers tend to distinguish between two basic types of problems: "First, those that are due to deficiencies in the state of general engineering practice, and second, those that arise from the state of the project they were engaged in. Engineering work on any particular development thus does not involve only the resolution of the problems arising from the specific circumstances of the project itself, but also contends with problems that are recognized as generic problems of engineering work per se" [8]. Students hardly believe these facts. In their opinion they would do much better and lead the project to success if they were the actors.

As soft skills are core competences of a software engineer, they should be a core ingredient of software engineering education at universities. Yet, soft skills should not be exercised in isolation, but rather in a typical professional setting and in conjunction with technical skills.

Project work is one approach to bring complexity and problem awareness into university education. Project work fosters many soft skills, such as communication skills and the ability to work together in a team. Interpersonal skills cannot be trained without other people around, and project work combines these competences with the context in which they are needed. Furthermore, project work offers students opportunities to understand inter-relationships between technical knowledge and soft skills. Project work in a university context gives students the chance to prove that they can really succeed while understanding the difficulties of project work and the reasons for failure.

This contribution investigates these issues in more detail. More specifically, the research question that drives this work is identifying the (major) challenges that students face in software engineering projects during their university education. To that end, we performed a qualitative analysis of post-mortem reports after finishing a capstone software engineering project.

The next section discusses related work before Section III provides some details on the setup of the capstone project and its underlying intended learning outcomes. Section IV outlines the research design before Section V presents and discusses the results of the qualitative analysis. A summary and outlook on future work concludes the paper.

 ISBN: 978-1-61208-827-3

II. Related Work

A better understanding of the inner workings of (capstone) projects in software engineering has been addressed in earlier work under various perspectives.

Brereton and Lees [9] investigate four factors that arguably have some impact on the outcomes of student projects. In particular, they focus on team size, range of abilities within the teams, the presence of female team members, and the mix of expertise beyond computing in the team. Their findings indicate that these factors do not have significant impact except for the gender mix – teams with two or more female members performed better than purely male teams.

Wikstrand and Börstler [10] identified various correlations between structural aspects of team projects. Most importantly, the type of the project, i.e., Web project, editors / generators, or other projects, plays a major role for project success. In addition, the authors identified project planning as a crucial, but often underestimated issue in student projects, particularly as students tend to not take planning and other process issues seriously.

Bastarrica et al. [11] investigated the role of four major aspects in capstone software engineering projects, namely technical challenges, teamwork, planning, and requirements clarification. For each of these four aspects, the authors tried to figure out if they changed between project initiation and closure with respect to their perceived value and difficulty. Most prominently, they perceived a decrease in the value of addressing technical issues properly and an increase of perceived difficulty of negotiating requirements with clients. On the other hand, in this study students seemed to have a realistic impression of difficulties associated to proper project planning, while they found teamworking harder than expected.

In a similar vein, Paasivara et al. [12] investigate 15 hypotheses with respect to a change in attitude over the duration of a capstone project. They also substantiated that technical issues lose importance, while non-technical issues, e.g., communication within the team and with stakeholders, understanding requirements, or following a defined process gained in terms of perceived importance and difficulty.

All the mentioned research provides valuable insights by substantiating of refuting hypotheses, based on a statistical analysis of data gained in surveys or interviews. Nevertheless, the origin of the formulated hypotheses remains unclear. For that reason, our research takes a step back in order to identify potential challenges, technical as well as non-technical, in capstone projects, based on a qualitative research design. In other words, our work tries to lay the foundation for formulating hypotheses on relevant success factors and challenges on a sound basis. This seems to be an important contribution to avoid overlooking crucial aspects due to premature formulation of hypotheses.

III. Structure and Goals of the Capstone Project

A. Educational Context

Students in our bachelor program in informatics can enroll in a Software Engineering project (SE project) in their final year. Participants acquired solid programming skills during courses in their first and second years, and they already took a compulsory introduction to software engineering and two elective courses focusing on software requirements, architecture, and testing in more detail. The SE project is intended as a means to tie together what has been learned on software engineering so far and gain hands-on experience in a self-directed mode. Students are supposed to learn from their own experiences, rather than getting rigid instructions from instructors. Generally, the main task in the project consists of devising and implementing a (Web-based) information system that supports and automates some business process (i.e., belongs to type "Web project" in Wikstrand's and Börstler's terminology [10]). In most cases, development is from scratch, i.e., no enhancement or reengineering of existing systems.

The SE project is offered as an elective course, which typically runs for 14 weeks with 6 European Credit Transfer System (ECTS) credits, i.e., puts a workload of approximately 180 hours on each participant. This workload includes 4 contact hours per week in which the project teams physically meet at the university. During these physical meetings, instructors are present, but act as observers in the background unless explicitly asked for support. Teams also meet virtually, using tools such as Skype or social media to make agreements. So far, we have had nine iterations of the SE project from 2011 to 2020.

Since the course is an elective, the number of participants varies from year to year, ranging from 10 to 25 students. Participants are split in project teams of 4 to 6 members. Typically, project topics are contributed by real customers and differ between teams. Customers typically do not have an IT background, which brings issues of multidisciplinarity into the projects.

Organizing a team, tailoring a process model, and developing a software system at the same time overstrained bachelor students. Therefore, we mixed bachelor and master students in the same project, starting with the third iteration of the SE project. Bachelor students focus on technical issues, constitute the development team, and experience project management in a more passive fashion. In contrast, the master students are in charge of leading the project and in particular of adapting the process model to the specific situation. Each team is free to choose a process model. In the more recent offerings of the project, teams regularly embarked on agile approaches, in particular Scrum [13]. The project teams decide on which deliverables and which project roles are really important and how they will implement the chosen process model.

To enable them to fulfill their roles, instructors offer on-demand coaching for master students to reflect and improve their leadership skills. This individual coaching establishes a forum to discuss challenges and problems they face in their teams and obtain help by the instructors to master these challenges.

B. Intended Learning Outcomes

The teaching goals of the capstone project differ for bachelor and master students. The focus for bachelor students

lies on understanding and combining chunks of technical knowledge, which up to then have been isolated, into one big picture, and on integrating in a team, which includes fostering communication skills. Master students focus on organizing and leading a team. The difficulties for them are, e.g., communicating with team members, structuring tasks, and motivating team members for effective teamwork. Master students are responsible for the results and for meeting deadlines, as well as for assuring the quality of the software.

Intended learning outcomes are mainly competences and, consequently, assessment is competence-oriented as well. At least two instructors accompany/observe the project teams during the presence hours each week to get an idea of teamwork and individual contributions.

In particular, grading of the bachelor students is based on the following aspects:

- technical quality of results (completeness, complexity of the project topics) including artefacts, such as requirements specifications, software architecture documents, test specifications, etc.
- (customization of and) adherence to a process model,
- individual technical contribution,
- individual team-orientation,
- individual self-reflection, and
- final presentation.

Likewise, grading for the master students is based on

- adaptation of the process model including documentation of the tailored process,
- process quality and leadership,
- self-reflection, and
- final presentation.

A post-mortem reflection is conducted as an additional element to stimulate learning. To that end, students were asked to reflect on their own individual role in the project, as well as the performance of the entire team. Reflection and metacognition are advantages of project work and are didactical methods to foster soft skills and competences.

Self-reflection is stimulated in two steps. First, each of the students has to prepare a short individual self-report that addresses issues such as

- their roles and tasks in the project,
- their expectations with respect to the project and the degree to which these had been met,
- particular issues in the project that they personally would have handled differently and, from their personal point of view, more successfully,
- which role they would have liked in the project and what they would have done differently in that role, and
- how interaction and cooperation between team members evolved during the project, including their subjective explanation for these changes.

Secondly, one week after the project is complete, the project teams meet with instructors for a post-mortem analysis session of approximately two hours, which serves to reiterate any possible aspect that seems worth being discussed in the group.

The self-reports establish the data base that we analyze subsequently.

IV. Research Design

A. Qualitative Research Design

The research uses a mixed methods approach with focus on qualitative analyses applying the basic strategy of Grounded Theory (GT) [14] in combination with Mayring's content analysis [15] [16] . GT aims at developing middle range theories by generating codes in a multi-stage procedure [17]. One step of the analysis consists of going through the material carefully and assigning appropriate semantic codes to the text segments to which they apply. "Coding means categorizing of segments of data with a short name that simultaneously summarizes and accounts for each piece of data. Your codes show how you select, separate, and sort data to begin an analytic accounting of them" [18]. In particular, the generation of the code system is not accomplished up-front, but rather by inductive category formation while going through the material. Simultaneously, new or existing codes are added as tags to relevant portions of the material while reading, abstracting and interpreting the texts.

B. Research Questions

This paper focusses on the following research questions: What are the main challenges for students in SE projects? What are major issues they have to deal with?

C. Research Data

An SE project team consists of 5 members on average. The large majority of participants was male, with only four females taking part over the years. All females were enrolled in the bachelor program.

To get answers to the research question, we rely on students' post-mortem self-reports. Over nine years we collected 79 reports from 81 students in 13 teams. All teams were guided by a master student, so that 14 self-reports were written by master students. The reports have an average length of two pages of prose text. Self-reports were written anonymously.

The self-reports encompass lots of potentially interesting data, which may be analysed from various perspectives. At the current stage of our research, we focus on challenges in SE projects to answer our research questions.

D. Application of the Research Design

As outlined above, our approach develops a category system incrementally by first marking those text segments that refer to challenges that students had to face in SE projects. This was accomplished using the MAXQDA analysis tool [19]. In the first coding procedure, subcategories are developed by going through all self-reports and marking text passages. Doing so results in an initial category system with little structure, which possibly includes some duplications. In a second step, initial categories are merged, sorted, and

 ISBN: 978-1-61208-827-3

grouped according to their meaning. In this way, an unambiguous and structured category system arises.

E. Initial Results - Overview

As a result, our research process yielded 1,379 codes in 19 categories. One of these categories is a main category "challenges", which is of particular interest for this paper.

The main category "challenges" encompasses 3 unambiguous subcategories (professional & technical issues, human factors, and organizational matters). Furthermore, we found 3 categories that collect complex challenges. Challenges in this category (internal communication, complexity, leadership) combine at least 2 challenges of the unambiguous categories. Challenges concerning internal communication, for example, may have human and organizational causes. In addition, a category "other challenges" was built to sum up marginal problems of working together.

Focussing on these relevant categories, 732 coded text segments from 72 self-reports were evaluated and showed the top three challenges in SE projects (see Table I and Figure 1): Human factors are the biggest challenges for students when working in a team, followed by organizational matters and professional & technical issues.

It is worth noting that the main categories "Internal communication", "Big picture / Complexity", and "Leadership (-)" are atomic in the sense that they do not have any subcategories.

TABLE I. STUDENTS' CHALLENGES IN SE CAPSTONE PROJECTS

	Students' Challenges in SE Capstone Projects		
	Percent (valid)	*Percent*	*Documents*
Human factors	87.5	79.75	63
Organizational Matters	75	68.35	54
Professional & Technical Issues	72.22	65.82	52
Internal Communication	50	45.57	36
Big Picture / Complexity	47.22	43.04	34
Other Challenges	40.28	36.71	29
Leadership (-)	25	22.78	18
DOCUMENTS with Code(s)	100	91.14	72
DOCUMENTS without Code(s)	-	8.86	7
ANALYSED DOCUMENTS	-	100	79

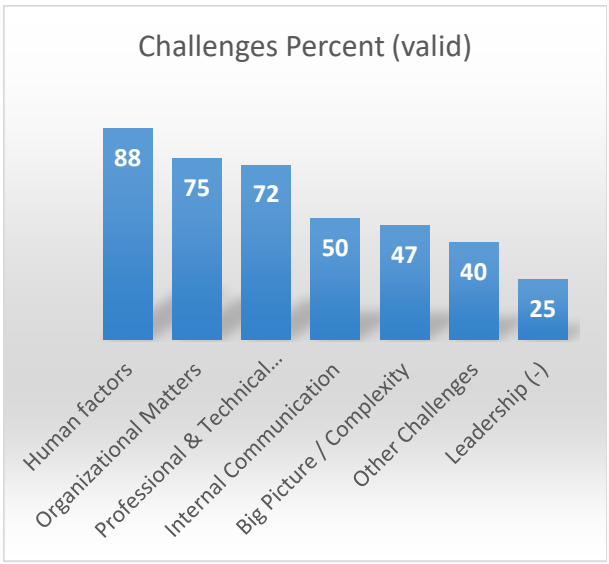

Figure 1. Students' Challenges in SE Capstone Projects (in percent)

F. Most Prominent Issues in Main Categories

A closer look at the main categories shows the following top issues within a specific category, as seen in Tables II, III, IV, and V.

TABLE II. TOP ISSUES IN HUMAN FACTORS

Top 4 Issues in Category "Human factors"		
Category	*Number of codes*	*Percent*
Collaboration bachelor and master students	32	10.49
Motivation	31	10.16
Collaboration	25	8.2
Communication with Third / Other Disciplines	20	6.56

The first subcategory refers to issues that relate to the interaction of bachelor and master students within a project team. The second subcategory reflects issues that are linked to a lack of individual motivation. The third subcategory refers to issues of how members of the project teams (excluding the master students) cooperated, while the last subcategory focusses on the communication with stakeholders outside the project team, possibly across disciplinary boundaries.

 ISBN: 978-1-61208-827-3

TABLE III. TOP ISSUES IN ORGANIZATIONAL MATTERS

Top 5 Issues in Category "Organizational Matters"		
Category	***Number of codes***	***Percent***
Time aspects / Timeliness	34	26.15
Management in general	33	25.38
Software Process Modell	16	12.31
Distribution of Tasks and Responsibilities	16	12.31
Communication	13	10.00

In terms of organisational matters, the first subcategory refers to issues related to stretching deadlines or skipping tasks due to time pressure or lack of time. The second subcategory collects issues related to organizing the project, e.g., developing a precise project plan, arrange meetings, facilitate meetings, etc. The third category refers to issues in the context of making the process model work properly. The fourth category addressed issues related to sharing the workload and assigning / accepting responsibilities in the project team, while the last one refers to (lack of) communication among team members.

TABLE IV. TOP ISSUES IN PROFESSIONAL & TECHNICAL ISSUES

Top 5 Issues in Category "Professional & Technical Issues"		
Category	***Number of codes***	***Percent***
Documentation	27	21.77
Software Requirements	25	20.16
Technical Knowledge	17	13.71
Effort Estimation	12	9.68
Tools	10	8.06

In the main category „Professional & Technical issues", the first subcategory deals with the deliverables beyond the actual code, e.g., requirements or architecture documents. The second subcategory refers to methodological issues related to clarifying requirements. The third and fifth subcategories deal with issues related to missing technical knowledge or tool deficiencies, while the fourth category refers to deficiencies related to time and effort estimations.

TABLE V. TOP ISSUES AMONG OTHER CHALLENGES

Top 3 Issues in Category "Other Challenges"		
Category	***Number of codes***	***Percent***
General Organisation	12	29.27
Shared Vision	9	21.95
Individual Situation	5	12.20

The main category „Other Challenges" relates to issues on a meta level, namely the organization of the project as a course and the individual situation of team members in the context of other subjects, but also a common understanding of priorities for the project, within the team or between team and instructors.

V. DISCUSSION

In contrast to the majority of earlier work on the subject, this work employs a well-founded qualitative approach to analysing educational data, in this case in the context of software engineering capstone projects.

Following this qualitative line of research, we arrived at 19 main categories of challenges that students face in capstone projects. These 19 main categories correspond to semantic clusters of issues raised in more than 70 textual post-mortem self-reports. Due to the chosen approach, categories and subcategories are subject to change whenever additional data become available.

The database of more than 70 textual self-reports is rich in the sense that it might provide insight from various diverse points of view. For now, we put a focus on identifying challenges students might face in a capstone project. Given that perspective, our result is closest in nature to the analysis by Paasivaara et al. [12]. Given our data, we can substantiate their findings that technical issues play only a minor role with respect to the "success" of a student project in comparison to other aspects, such as collaboration within the team and beyond, issues of project management and organisation, and methodological issues related to requirements engineering and effort estimation. In addition, we also found indications that, like stated by Wikstrand and Börstler [10], issues related to project planning are some challenge. Yet, our results are more fine-grained, thus allowing for more sophisticated hypotheses that might be tested subsequently. For instance, project organisation (and not just planning), individual motivation and individual deficiencies in setting or adhering to deadlines have not been mentioned as important issues in related research.

Furthermore, our findings are pretty well in line with the intended learning outcomes of the capstone project. As mentioned in Section III-B, developing problem-awareness with respect to issues related to a gross oversight and team formation and teamwork are among the most important goals of the capstone project. As these issues are mentioned frequently in the coded text segments (see Table I), students actually seem to realize that things look simpler as they are on closer inspection. As a consequence, we largely reached our intended learning outcomes.

VI. SUMMARY AND OUTLOOK

Providing students with an opportunity to tie together their knowledge on engineering (moderately) complex software systems and exercise and expand non-technical competences is paramount for well-educated graduates in software engineering. Capstone software engineering projects are very popular approach to that end. Yet, these capstone projects vary in terms of "success", both from the point of view of involved stakeholders and with respect to intended learning outcomes.

 ISBN: 978-1-61208-827-3

This paper aims at getting better insight into which challenges student face in software engineering capstone projects. To do so, self-reports of nine years were evaluated qualitatively with the MAXQDA analysis toolset. Our findings indicate that major challenges for students lie in human, organizational and professional. Furthermore, internal communication, complexity, and leadership are areas of potential difficulties in student projects.

As main results, our research identifies areas that pose difficulties of some sort or another to students when running a somewhat complex software engineering project. This establishes an opportunity to state more elaborate hypotheses on success or risk factors with respect to intended learning outcomes for software engineering outcomes.

In future studies, self-reports will be evaluated with other foci, e.g.: What are the learning outcomes from students` perspectives? What did students learn? Are there differences between bachelor and master students concerning the mentioned questions?

ACKNOWLEDGMENT

This work is funded by the German Federal Ministry of Education and Research (Bundesministerium für Bildung und Forschung) under grant number 01PL17022A as part of the EVELIN project. The authors are responsible for the content of this publication.

REFERENCES

[1] I. Sommerville, *Software Engineering,* 9th ed. Boston: Pearson, 2011.

[2] C. Gold-Veerkamp, *Erhebung von Soll-Kompetenzen im Software Engineering - Anforderungen an Hochschulabsolventen aus industrieller Perspektive*. Wiesbaden: Springer Vieweg, 2015.

[3] P. L. Li, A. J. Ko, and J. Zhu, "What Makes a Great Software Engineer?," in 37th International Conference on Software Engineering (ICSE), 2015, pp. 700–710.

[4] H.-K. Lu, C.-H. Lo, and P.-C. Lin, "Competence analysis of IT professionals involved in business services — Using a qualitative method," in 24th Conference on Software Engineering Education and Training (CSEE&T), 2011, pp. 61–70.

[5] I. Richardson, L. Reid, S. B. Seidman, B. Pattinson, and Y. Delaney, "Educating software engineers of the future: Software quality research through problem-based learning," in 24th Conference on Software Engineering Education and Training (CSEE&T), 2011, pp. 91–100.

[6] J. G. Rivera-Ibarra, J. Rodríguez-Jacobo, and M. A. Serrano-Vargas, "Competency Framework for Software Engineers," in 23rd Conference on Software Engineering Education and Training (CSEE&T), 2010, pp. 33–40.

[7] Y. Sedelmaier, *Basics of didactics for software engineering:* Research-based and application-oriented development and evaluation. Saarbrücken: LAP LAMBERT Academic Publishing, 2019.

[8] G. Button and W. Sharrock, "Project work: The organisation of collaborative design and development in software engineering," (en), *Comput Supported Coop Work*, vol. 5, no. 4, pp. 369–386, 1996.

[9] P. Brereton and S. Lees, "An Investigation of Factors Affecting Student Group Project Outcomes," in 18th Conference on Software Engineering Education & Training (CSEE&T), 2005, pp. 163–170.

[10] G. Wikstrand and J. Borstler, "Success Factors for Team Project Courses," in 19th Conference on Software Engineering Education & Training (CSEE&T), 2006, pp. 95–102.

[11] M. C. Bastarrica, D. Perovich, and M. M. Samary, "What Can Students Get from a Software Engineering Capstone Course?," in 2017 IEEE/ACM 39th International Conference on Software Engineering: Software Engineering Education and Training Track (ICSE-SEET), 2017, pp. 137–145.

[12] M. Paasivaara, D. Voda, V. T. Heikkilä, J. Vanhanen, and C. Lassenius, "How Does Participating in a Capstone Project with Industrial Customers Affect Student Attitudes?," in 2018 IEEE/ACM 40th International Conference on Software Engineering: Software Engineering Education and Training (ICSE-SEET), 2018, pp. 49–57.

[13] M. Klopp, C. Gold-Veerkamp, J. Abke, K. Borgeest, R. Reuter, S. Jahn, J. Mottok, Y. Sedelmaier, A. Lehmann, and D. Landes, "Totally Different and yet so Alike," in 4th European Conference on Software Engineering Education (ECSEE'20): ACM, 2020, pp. 12–21.

[14] B. G. Glaser and A. L. Strauss, *The Discovery of Grounded Theory: Strategies for Qualitative Research*. Chicago: Aldine Transaction, 2009.

[15] P. Mayring, *Qualitative Content Analysis.* Available: http://www.qualitative-research.net/index.php/fqs/article/view/1089/2385 (2020, Sep. 02).

[16] ____, *Qualitative Inhaltsanalyse:* Grundlagen und Techniken, 11th ed. Weinheim: Beltz, 2010.

[17] U. Kuckartz, *Qualitative Inhaltsanalyse. Methoden, Praxis, Computerunterstützung,* 4th ed. Weinheim, Basel: Beltz Juventa, 2018.

[18] K. Charmaz, *Constructing grounded theory,* 2nd ed. Los Angeles: SAGE, 2014.

[19] S. Rädiker and U. Kuckartz, *Analyse qualitativer Daten mit MAXQDA*. Wiesbaden: Springer Fachmedien Wiesbaden, 2019.

 ISBN: 978-1-61208-827-3