

**A FRAMEWORK FOR GAMIFICATION OF  
PROJECT-BASED SOFTWARE ENGINEERING  
EDUCATION**







MAURÍCIO RONNY DE ALMEIDA SOUZA

**A FRAMEWORK FOR GAMIFICATION OF  
PROJECT-BASED SOFTWARE ENGINEERING  
EDUCATION**

Tese apresentada ao Programa de Pós-Graduação em Ciência da Computação do Instituto de Ciências Exatas da Universidade Federal de Minas Gerais como requisito parcial para a obtenção do grau de Doutor em Ciência da Computação.

ORIENTADOR: EDUARDO FIGUEIREDO

Belo Horizonte  
Setembro de 2019







MAURÍCIO RONNY DE ALMEIDA SOUZA

**A FRAMEWORK FOR GAMIFICATION OF  
PROJECT-BASED SOFTWARE ENGINEERING  
EDUCATION**

Thesis presented to the Graduate Program  
in Ciência da Computação of the Univer-  
sidade Federal de Minas Gerais in partial  
fulfillment of the requirements for the de-  
gree of Doctor in Ciência da Computação.

ADVISOR: EDUARDO FIGUEIREDO

Belo Horizonte

September 2019



© 2019, Maurício Ronny de Almeida Souza.  
Todos os direitos reservados

Ficha catalográfica elaborada pela Biblioteca do ICEx - UFMG

Souza, Maurício Ronny de Almeida.

S729f      A Framework for gamification of project-based  
software engineering education / Maurício Ronny de  
Almeida Souza. — Belo Horizonte, 2019.  
xxvi, 176 f. il.; 29 cm.

Tese (doutorado) - Universidade Federal de Minas  
Gerais – Departamento de Ciência da Computação.

Orientador: Eduardo Magno Lages Figueiredo

1. Computação – Teses. 2. Engenharia de software -  
ensino e aprendizagem. 3. Teoria dos jogos.  
4. Aprendizagem por atividades. I. Orientador. II. Título.

CDU 519.6\*32(043)





UNIVERSIDADE FEDERAL DE MINAS GERAIS  
INSTITUTO DE CIÊNCIAS EXATAS  
PROGRAMA DE PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

## FOLHA DE APROVAÇÃO

A Framework for Gamification of Project-based Software Engineering  
Education

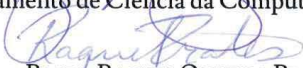
**MAURÍCIO RONNY DE ALMEIDA SOUZA**

Tese defendida e aprovada pela banca examinadora constituída pelos Senhores:

  
PROF. EDUARDO MAGNO LAGES FIGUEIREDO - Orientador  
Departamento de Ciência da Computação - UFMG

  
PROFA. CLAUDIA MARIA LIMA WERNER  
Instituto Alberto Luiz Coimbra de Pós Graduação e Pesquisa de Engenharia, Programa de  
Engenharia de Sistemas. - UFRJ

  
PROF. MARCO TULLIO DE OLIVEIRA VALENTE  
Departamento de Ciência da Computação - UFMG

  
PROFA. RAQUEL OLIVEIRA PRATES  
Departamento de Ciência da Computação - UFMG

  
PROFA. CHRISTIANE ANNELIESE GRESSE VON WANGENHEIM  
Departamento de Informática e Estatística - UFSC

Belo Horizonte, 13 de Setembro de 2019.







*To Mara and Rosa (in memoriam), with love.*







# Acknowledgments

Many friends, relatives, and researchers took part in the execution of this research and the fulfilment of the requirements of the phd program. However, my mother (Mara), my grandmother (Rosa), and my eternal friend Ariane, who were always supporting me, could not be physically present to see my accomplishment. I would like to thank them for their continuous watch over my steps.

I would not be able to even start this journey if I did not have the support of my beloved wife, Renata. As a final gift, Renata provided me with the most beautiful flower I've ever seen: Liana. Since Liana's birth, my life definitely changed for better, with many blessings everyday.

I would also like to devote a special acknowledgment to my advisor, Eduardo Figueiredo. Eduardo supported me not only as an advisor, but also as friend. In the troubled times, he could always provide a comforting word, and gave me all the freedom I needed to execute the research I wanted to.

Special thanks to my family, including the friends that are brothers and sisters: Mauricia, Olivar, William, Victor, Julieta, Sebastião, Pato, Ponei, Luizinho, Yoshio, Portela, Carol, Nath, Marcel, Marina, Madalena, João, Daniel, Adriele, and Patricia.

I would like to thank all the friends from Lab-Soft, who fought the battles shoulder by shoulder with me at UFMG: J-John, Katt, Fischão, Cleitão, João, Igor, Markos, Allan, Adriano, Daenerys, and Lucas. Their friendship made the phd-student life a lot easier with all the laughs and stories. The new friends I made in Lavras, who supported me in my new life, also have to be mentioned: Du, Mayron (the terrible), Jadson, Paulo Afonso, Valéria, Victor, Paula, Heitor, Luiza, Lala, Gui, the people from MaUFLA, Otacílio, Cida, Chris, Bruno, Marlon, and Vinicius.







*“No. Not even in the face of Armageddon. Never compromise.”*  
(Alan Moore, Watchmen)







# Resumo

Balancear teoria e prática é um desafio recorrente no ensino de engenharia de software. No entanto, as diretrizes curriculares da ACM / IEEE e da Sociedade Brasileira de Computação (SBC) enfatizam a necessidade de proporcionar aos alunos experiências práticas suficientes para o desenvolvimento das competências esperadas para os profissionais de engenharia de software. As abordagens baseadas em projetos e jogos têm sido amplamente utilizadas para atender a essa necessidade. Portanto, o objetivo desta tese é a proposta de um framework conceitual para apoiar a adoção conjunta de Aprendizagem Baseada em Projetos (PBL) e gamificação para introduzir a prática na educação engenharia de software. Seguindo o paradigma de design science, realizamos uma série de estudos empíricos e de literatura para entender o uso desses métodos educacionais para apoiar o ensino de engenharia de software. Com base nas lições aprendidas desses estudos, propomos e avaliamos o GaPSEE, um framework para apoiar professores no planejamento e execução de tarefas práticas usando os princípios de PBL e gamificação. O GaPSEE foi avaliado em cinco estudos de caso, executados em três universidades federais no Brasil. O resultado de entrevistas com 4 professores e as respostas de uma pesquisa com 76 alunos são indicativos de uma percepção positiva sobre o uso do GaPSEE para introduzir a prática no ensino de engenharia de software. Os principais benefícios observados com a aplicação do GaPSEE estão relacionados ao aumento da participação e interação dos alunos com os professores, maior engajamento e motivação dos alunos, contextualização significativa da prática e um roteiro de atividades para orientar e acompanhar o progresso dos alunos.

**Palavras-chave:** Educação em engenharia de software, aprendizagem baseada em projetos, gamificação.







# Abstract

Balancing theory and practice is a recurring challenge in software engineering (SE) education. However, the curriculum guidelines of the ACM/IEEE and Brazilian Computer Society (SBC) emphasize the need of providing students with sufficient practical experiences, for the development of the competences expected for SE professional. Project and game based approaches have been largely used to address this necessity. Therefore, the goal of this thesis is the proposal of a framework to support the joint adoption of Project-Based Learning (PBL) and gamification to introduce practice in SE education. Following the design science paradigm, we conducted a series of literature and empirical studies for understanding the use of these educational methods to support software engineering education. Based on the lessons learned from these studies, we propose and evaluate GaPSEE, a framework to support lecturers in the planning and execution of practical assignments using principles of PBL and gamification. GaPSEE was evaluated in five case studies from three federal universities in Brazil. The result from interviews with 4 lecturers and responses of a survey with 76 students are indicative of positive perception about the use of GaPSEE to introduce practice in software engineering education. The main benefits noticed from the application of GaPSEE are related to increased students participation and interaction with lecturers, increased engagement and motivation of students, meaningful contextualization of practice, and having a roadmap of activities to guide and track the progress of students.

**Keywords:** Software engineering education, project-based learning, gamification.







# List of Figures

1.1	Study design . . . . .	7
2.1	Timeline of primary studies [Souza et al., 2018]. . . . .	23
3.1	Examples of four badges in the SE course. . . . .	35
3.2	Badges exhibited in the Hall of Fame for the top 3 students of all time. . .	36
3.3	Results for the survey background questions BQ1 to BQ3. . . . .	40
3.4	Results for the survey background question BQ4. . . . .	40
3.5	Survey results on the students perception on the use of badges. . . . .	41
3.6	Survey results on the students perception of the “Hall of Fame”. . . . .	42
4.1	Action Research cycle, adapted from Davison et al. [2004]. . . . .	51
4.2	Academic period of the participants (Q2). . . . .	63
4.3	First Contact with software engineering in academia (Q4). . . . .	64
4.4	Professional experience with software development or software engineering (Q5). . . . .	64
4.5	Evaluation of the use of software projects as practical assignment in software engineering education (Q6). . . . .	65
4.6	Evaluation of the use of traditional lectures and punctual assignments in software engineering education (Q7). . . . .	66
4.7	Contribution of the project in learning specific software engineering topics (Q8) for the PBL sample. . . . .	67
4.8	Contribution of the project in learning specific software engineering topics (Q8) for the Non-PBL sample. . . . .	67
4.9	Comparison of the results for Q8. . . . .	68
4.10	Positive aspects stated in the responses of Q9. . . . .	69
4.11	Negative aspects stated in the responses of Q10. . . . .	71
5.1	GaPSEE project layout . . . . .	86



5.2	Expected actions for each component of GaPSEE . . . . .	92
5.3	GaPSEE process . . . . .	93
5.4	First level of the assignment in PLT case. . . . .	104
5.5	Leaderboard in the PLT case. . . . .	106
6.1	Study design. . . . .	113
6.2	A level from the SQM case study . . . . .	118
6.3	The leaderboard from SQM case study . . . . .	118
6.4	Higher education program attended by the participants (N=76). . . . .	120
6.5	Participants' (a) age,(b) gender, (c) professional experience (in years), and (d) frequency of playing games (N=76). . . . .	121
6.6	Importance of practice for software engineering education and adequacy of the assignment for the courses. . . . .	122
6.7	Results of the Survey for the Aesthetics of the gamification approach. . . .	123
6.8	Participants' perception on aesthetic aspects of the gamification approach.	123
6.9	Participants' perception on development of specific skills. . . . .	124
6.10	Participants' perception on development of general skills. . . . .	125
6.11	Distribution of occurrences of positive and negative codes for the case studies.	128
6.12	Ratio of Codes per Participants for positive and negative aspects (# occur- rence of positive or negative codes) / (size of the sample). . . . .	129



# List of Tables

2.1	SE education Knowledge Areas [IEEE/ACM, 2015]	12
2.2	SWECOM Skill Areas and Skills	13
2.3	Game elements used in SE education context[Souza et al., 2018]	28
2.4	Game elements to support software engineering practice and education	30
3.1	Hall of Fame with the top ten students of all time in the SE Course	36
3.2	Questionnaire	38
3.3	Interview script	39
4.1	Learning goals of the assignment	54
4.2	Overview of the installments of the SE course	55
4.3	Questionnaire Structure	62
4.4	Population sampling	63
4.5	Positive aspects identified in the responses of the PBL and Non-PBL samples for Q9.	73
4.6	Categorization of the positive aspects identified in the responses of the PBL and Non-PBL samples for Q9	73
4.7	Negative aspects identified in the responses of the PBL and Non-PBL samples for Q10.	74
4.8	Categorization of the negative aspects identified in the responses of the PBL and Non-PBL samples for Q10	75
5.1	GaPSEE Guidelines	88
5.2	Learning goals of the assignment in PLT case.	98
5.3	Iterations of the PLT case	101
5.4	Tasks of the second iteration of the PLT case	101
5.5	GaPSEE suggestion of game elements to achieve gamification goals.	103
6.1	Background of the participants	115



6.2	Use of practical assignments, PBL, and Gamification in previous iterations of each course . . . . .	115
6.3	Organization of the case studies . . . . .	117
6.4	Population sample for the survey study. . . . .	120
6.5	Positive codes and categories mapped from the responses of the participants.	126
6.6	Negative codes and categories mapped from the responses of the participants.	127
6.7	Occurrences of positive and negative codes for each category. . . . .	129
6.8	Script for interviewing lecturers . . . . .	131
6.9	Codes and categories related to positive aspects mapped from the interviews.	137
6.10	Categories of codes mapped from the interviews. . . . .	138



# Contents

<b>Acknowledgments</b>	<b>xi</b>
<b>Resumo</b>	<b>xv</b>
<b>Abstract</b>	<b>xvii</b>
<b>List of Figures</b>	<b>xix</b>
<b>List of Tables</b>	<b>xxi</b>
<b>1 Introduction</b>	<b>1</b>
1.1 Problem and motivation . . . . .	3
1.2 Goals . . . . .	5
1.3 Method . . . . .	6
1.4 Contribution and relevance . . . . .	7
1.5 Thesis project outline . . . . .	9
<b>2 Literature Review</b>	<b>11</b>
2.1 Software engineering education . . . . .	11
2.1.1 Practice in software engineering education . . . . .	14
2.2 Project-Based Learning (PBL) . . . . .	15
2.2.1 Characteristics of PBL . . . . .	16
2.2.2 Challenges and issues related to the use of PBL in software engineering education . . . . .	18
2.3 Gamification . . . . .	21
2.3.1 Game elements . . . . .	23
2.3.2 Gamification in software engineering education . . . . .	24
2.3.3 Cases of use of gamification in software engineering education . . . . .	27
2.3.4 Game elements used in the gamification of software engineering . . . . .	28



2.4	Discussion of literature gaps . . . . .	29
2.5	Final remarks . . . . .	31
<b>3</b>	<b>Empirical Study on the Use of Gamification in Software Engineering Education</b>	<b>33</b>
3.1	Course setup . . . . .	34
3.2	Study settings . . . . .	36
3.2.1	Study goals and research questions . . . . .	37
3.2.2	Study design and research methods . . . . .	37
3.2.3	Planning of the study phase I - Survey . . . . .	37
3.2.4	Planning of the study phase II - Interviews . . . . .	38
3.3	Results . . . . .	39
3.3.1	Study phase I – Survey results . . . . .	39
3.3.2	Study phase II – Interviews results . . . . .	43
3.4	Discussion . . . . .	45
3.4.1	RQ1 – Badges in a software engineering course . . . . .	45
3.4.2	RQ2 – Leaderboards in a software engineering course . . . . .	45
3.5	Threats to validity . . . . .	46
3.6	Final remarks . . . . .	47
<b>4</b>	<b>Empirical Study on the Use of PBL in Software Engineering Education</b>	<b>49</b>
4.1	Study settings . . . . .	49
4.1.1	Study goals and research questions . . . . .	50
4.1.2	Research method . . . . .	50
4.1.3	Study design . . . . .	51
4.2	Course setup . . . . .	52
4.2.1	PBL in the software engineering course . . . . .	52
4.2.2	Learning goals . . . . .	54
4.3	Observation from the Action Research cycles . . . . .	55
4.3.1	Type of project and realism . . . . .	55
4.3.2	Guidance, freedom of choice, and evaluation . . . . .	57
4.3.3	Teamwork and scalability . . . . .	60
4.4	Questionnaire analysis . . . . .	61
4.4.1	Population sample . . . . .	62
4.4.2	Participants background . . . . .	63
4.4.3	Evaluation of the learning method . . . . .	64



4.4.4	Evaluation of the project contribution to learning software engineering topics . . . . .	66
4.4.5	Positive and negative aspects of the PBL course . . . . .	68
4.4.6	Comparison of positive and negative aspects between PBL and Non-PBL courses . . . . .	72
4.5	Discussion . . . . .	76
4.5.1	RQ1 – The challenges of using PBL in an introductory software engineering course . . . . .	76
4.5.2	RQ2 – Students’ perception on the use of PBL in an introductory software engineering course . . . . .	78
4.6	Threats to validity . . . . .	79
4.7	Final remarks . . . . .	80
<b>5</b>	<b>A Framework for the Gamification of Project Based Software Engineering Education</b>	<b>83</b>
5.1	Goal and scope . . . . .	84
5.2	Target audience . . . . .	85
5.3	Components . . . . .	85
5.4	Structure and roles in GaPSEE practical assignments . . . . .	85
5.5	GaPSEE Guidelines . . . . .	87
5.6	GaPSEE Process . . . . .	91
5.6.1	Setup phase . . . . .	92
5.6.2	Execution phase . . . . .	94
5.6.3	Evaluation phase . . . . .	95
5.6.4	Refinement phase . . . . .	96
5.7	Suggestions for implementation . . . . .	96
5.7.1	Planning the assignment . . . . .	97
5.7.2	Planning the project . . . . .	99
5.7.3	Planning gamification . . . . .	102
5.7.4	Executing the assignment . . . . .	105
5.8	Final remarks . . . . .	108
<b>6</b>	<b>Evaluation of the Proposed Framework</b>	<b>111</b>
6.1	Study settings . . . . .	112
6.1.1	Study goal and research questions . . . . .	112
6.1.2	Study design and research methods . . . . .	112
6.2	Case Studies . . . . .	113



6.2.1	Selection of case studies . . . . .	114
6.2.2	Preparation and execution of the case studies . . . . .	116
6.3	Survey with students . . . . .	119
6.3.1	Population Sample . . . . .	119
6.3.2	Results . . . . .	120
6.4	Interviews with lecturers . . . . .	131
6.4.1	Previous installments of the courses . . . . .	132
6.4.2	Changes in students' attitudes . . . . .	133
6.4.3	Changes in the management and preparation of the assignment	135
6.4.4	Relevance and positive aspects of GaPSEE approach . . . . .	136
6.4.5	Negative aspects of GaPSEE approach and improvements for replications of the case studies . . . . .	138
6.4.6	Perspective on the use and recommendation of GaPSEE . . . . .	140
6.5	Discussion . . . . .	141
6.6	Threats to validity . . . . .	144
6.7	Final Remarks . . . . .	145
<b>7</b>	<b>Conclusion</b>	<b>147</b>
7.1	Summary . . . . .	147
7.2	Contribution . . . . .	149
7.3	Future Work . . . . .	150
	<b>Bibliography</b>	<b>151</b>
	<b>Appendix A Primary Studies used in the Systematic Mapping</b>	<b>163</b>
	<b>Appendix B Questionnaire for the Evaluation of the Framework</b>	<b>169</b>



# Chapter 1

## Introduction

Software engineering is the application of a systematic, disciplined, quantifiable approach to the development, operation and maintenance of software [IEEE/ACM, 2015]. The challenges of educating new software engineers are more than just programming, they include attention to details, such as quality, schedule, and economic goals [IEEE/ACM, 2015]. For instance, an important challenge in software engineering education arises from the dual nature of the software engineering discipline: it has roots in Computer Science and has emerged as an engineering discipline. Hence, it affects both theory and practice [IEEE/ACM, 2015]. This characteristic has a direct impact on the amount of material instructors must cover in software engineering classrooms. In addition, software professionals are required not only to understand technical challenges but also to be up-to-date with nontechnical issues, including management, communication, and teamwork.

In software engineering higher education, besides learning theory and acquiring technical skills, students need to develop the ability to apply, evolve, and practice those skills throughout their lifetime [Gary, 2015]. Additionally, soft skills, such as leadership, teamwork, decision-making, negotiation, and self-reflection, are important abilities for software engineering practice, since software development also involves several human and social aspects [Marques et al., 2014]. Nevertheless, the development of these crosscutting capabilities is usually less supported in Computer Science programs [Marques et al., 2014].

There is no consensus on how to teach software engineering, since each institution adopts its own methods based on the experience of its professors [Marques et al., 2014]. Traditional approaches (expository lectures, exams, and complimentary assignments) are still largely used by lecturers [Sancho-Thomas et al., 2009; Bessa et al., 2012; Marques et al., 2014; Fioravanti et al., 2018] . A possible cause is the difficulty in



changing the instructional process used by lecturers, and that it is a common pattern in computer science and engineering courses [Marques et al., 2014]. However, it may lead to demotivating students [Barnes et al., 2008; Prikladnicki et al., 2009; Bessa et al., 2012]. In addition, teacher-centered educational methods may not support the practical development of competences [Barnes et al., 2008; Sancho-Thomas et al., 2009] and may have limited learning efficiency [Prikladnicki et al., 2009]. Therefore, student-centered approaches may be more suited for allowing the development of competences as the students learn-by-doing, with a higher motivation from the learner, a more active role in learning process, and better learning in the application level [Prikladnicki et al., 2009; Fioravanti et al., 2018; Kuhrmann and Münch, 2018].

Curricular guidelines, such as the ACM/IEEE Curriculum Guidelines for software engineering programs (SE 2014 [IEEE/ACM, 2015]), recommend including team-based projects in the curriculum of the software engineering and computer science undergraduate programs. The necessity of providing real world experience of software development to students is a recurring theme in SE 2014, and several of its guidelines address this matter [IEEE/ACM, 2015]. For instance, Curriculum Guideline 5 suggests that “students also need practical material to be taught early so they can gain maturity by participating in real-world development experiences (...)” [IEEE/ACM, 2015]. Curriculum Guideline 10 also discusses the multiple dimensions of the problem-solving aspect of software engineering, and suggests that “problem solving is better learning through practice and taught by example” [IEEE/ACM, 2015]. Furthermore, Curriculum Guideline 17 suggests the need of using interesting, concrete and convincing examples to motivate students. Finally, Curriculum Guideline 14 objectively declares “the curriculum should have a significant real-world basis” [IEEE/ACM, 2015].

Several learning approaches have been proposed and applied to introduce practical aspects in software engineering education [Marques et al., 2014], including: game-based learning [Peixoto et al., 2014], case studies [Razali and Chitsaz, 2010], simulation [Blake, 2003], inverted classrooms [Herold et al., 2012], maintenance projects [Andrews and Lutfiyya, 2000], service learning [Chao and Randles, 2009], and open source development [Ellis et al., 2007]. The applied nature of software engineering has also motivated the adoption of game-related approaches for software engineering education [von Wangenheim and Shull, 2009; Souza et al., 2018].



## 1.1 Problem and motivation

A recurring challenge in software engineering education is engaging students to experience the professional practices of software engineering in such a way that they can understand which practices and techniques are useful in various different situations [IEEE/ACM, 2015]. However, it is difficult to achieve the appropriate balance between theory and practice. This leads to a gap between the skills of recent graduates and the expectations of the software industry with the level of preparation of the recently graduated professionals [Radermacher et al., 2014], specially regarding the lack of necessary competences to start performing their activities efficiently [von Wangenheim and da Silva, 2009; Moreno et al., 2012; Meira, 2015]. Therefore, there is a gap between learn by studying (in academia) and learn by doing (at work) [Moreno et al., 2012].

Software engineering courses in Computer Science or Information Systems departments usually provide limited opportunities for understanding the details related to practices, such as project management, quality assurance, and clients requirements understanding [Peixoto et al., 2011]. Software processes, for instance, play a key role in software engineering education, both as a focal and as a crosscutting topic to reinforce students' understanding of software engineering practice [IEEE/ACM, 2015]. However, students practice of software process in academia is limited to the practical assignments they are exposed during academic life (e.g., project-based activities, capstone projects, and practical exercises). In addition, the nature of these assignments and projects proposed in the classroom is limited in scope and time. Therefore, incorporating real-world elements into the curriculum is a crucial challenge to enable effective learning of software engineering skills and concepts [IEEE/ACM, 2015].

The curriculum guidelines of the ACM/IEEE [IEEE/ACM, 2015] emphasize that the professional competences emerge through the theoretical study of knowledge units and the practical application of their concepts. As a consequence, it is necessary to move beyond the expository classroom format, since it does not favor effective student learning [IEEE/ACM, 2015]. These guidelines also suggest the importance of introducing real world problems, related to software engineering, in the learning process, and the inclusion of knowledge units that allow the development of the competences expected for professionals in the area.

Similarly, the National Curricular Guidelines for Computer Science Bachelor Programs (DCN16) [MEC, 2016] and the Formation References for Undergraduate Computer Courses (RR-CC17) [Zorzo et al., 2017] suggest a shift from curricular structure based on learning topics to a structured focused in the development of competences. DCN16 [MEC, 2016] recommends the use of student-centered educational methods,



supported by lecturers in a role of facilitator. RR-CC17 [Zorzo et al., 2017], states that “the pedagogical project should adopt teaching and learning methods that promote the explicit relation between contents covered and competences expected for graduates”.

From the instructor perspective, however, developing professional competencies in students using practical assignments is challenging, because it requires that: (i) instructors understand and plan the expected outcomes in terms of skills the students are supposed to develop; (ii) instructors specify processes, activities, policies or procedures that allow and induce students to develop specific skills; (iii) students are properly trained or mentored for executing the specified process, activities or procedures in order to develop the expected skills; (iv) instructors evaluate the outputs of students activities for assessing the development of skills. Additionally, it is important that students are well motivated to perform these activities.

One strategy largely used to overcome these challenges is the introduction of software projects in software engineering education (e.g., capstone and project-based courses) [Delgado et al., 2017; Marques et al., 2018]. In this context, Project-Based Learning (PBL) is one of the main successful student-centered educational approach broadly used in computing science, information systems and engineering courses [Delgado et al., 2017; Marques et al., 2018; Macias, 2012; Jazayeri, 2015; Shuto et al., 2016; Warin et al., 2016; Yamada et al., 2014]. However, there is a shortage of comprehensive methodological frameworks and tools for PBL [Warin et al., 2016; Macias, 2012]. As a consequence, it may aggravate other problems related to PBL adoption, such as: the effort required to run PBL courses [Harms and Hastings, 2016; Hanakawa, 2015; Nguyen et al., 2013; Marques et al., 2018; Rupakheti et al., 2017; Daun et al., 2016; Gary, 2015; Mäkiö et al., 2017]; scalability [Harms and Hastings, 2016; Gary, 2015]; and the difficulty to track students progress through the project [Fukuyasu et al., 2013; Harms and Hastings, 2016].

Gamification, on the other hand, has been used in software engineering education as a strategy to engage and motivate students in performing specific behaviours, such as the more frequent use of specific tools, acquiring the habit of applying specific techniques, or being more participative in the classroom [Singer and Schneider, 2012]. Gamification has also been used as a strategy to induce learners to use specific software engineering abilities or practices, by promoting competition or systematically rewarding learners as they perform expected actions or show expected behaviors [Laskowski, 2015]. Therefore, it is a relevant strategy to support students in developing an appreciation for continued learning and in acquiring habits for professional software development [Singer and Schneider, 2012; Laskowski, 2015; Souza et al., 2018]. Similar to PBL, a problem related to gamification is the difficulty to adapt it to each context, as there are



few studies providing general guidelines to use this technique for software engineering education.

Therefore, the motivation for using specific methods and approaches in learning process is influenced by several criteria, such as the flexibility and ease of using the approaches, their suitability for being used by most instructors, and the effort, restrictions and skills involved in the use of these approaches [Marques et al., 2014]. For instance, in a survey with 89 lecturers about the adoption of games and game elements in software engineering education [Rodrigues et al., 2018], the most recurring cause of not using these approaches were related to: the lack of knowledge (21.3%); not knowing appropriate games for software engineering education (15.7%); lack of time (13.5%); not believing in the method (6.7%); lack of interest in the method (5.6%); lack of materials to support the adoption of this method (4.5%); and lack of resources (2.2%). Even considering that the use of games in software engineering education is not new [Souza et al., 2018], the lack of approachable models for introducing alternative learning methods is still a barrier. Thus, providing educators with appropriate resources to support the adoption of alternative learning methods may contribute to addressing the previously mentioned problems.

As a consequence, the motivation of this thesis lies on: (i) the necessity of introducing practice in software engineering education and the related challenges; (ii) the gap between what and how software engineering is taught in university and the competences needed from professionals entering the industry; (iii) the necessity of moving beyond expository lectures; and (iv) the need of approachable materials and resources to support educators in the introduction of alternative student-centered educational methods.

## 1.2 Goals

The goal of the thesis is to propose a conceptual framework to support the joint adoption of gamification and Project-Based Learning for practical software engineering education. This framework is intended to provide guidelines on how to setup educational software development projects, using gamification to guide and motivate students on performing specific software engineering practices. To achieve this goal, the following specifics goals are defined:

- **SG1** Investigate how gamification can be used to support software engineering education.
- **SG2** Investigate how PBL can be used to support software engineering education.



- **SG3** Investigate the benefits and drawbacks of the joint use of gamification and PBL to support software engineering education.

The goal of this research is not only defining a novel educational method for software engineering education, but also to integrate existing approaches (PBL and gamification) in an unified approach.

The scope of this thesis is limited to the investigation of PBL and gamification in the context of software engineering education. Although we acknowledge the existence and relevance of other methods and techniques for this context, it is out of the scope of this thesis the comparison of the approach proposed in this thesis project to other approaches.

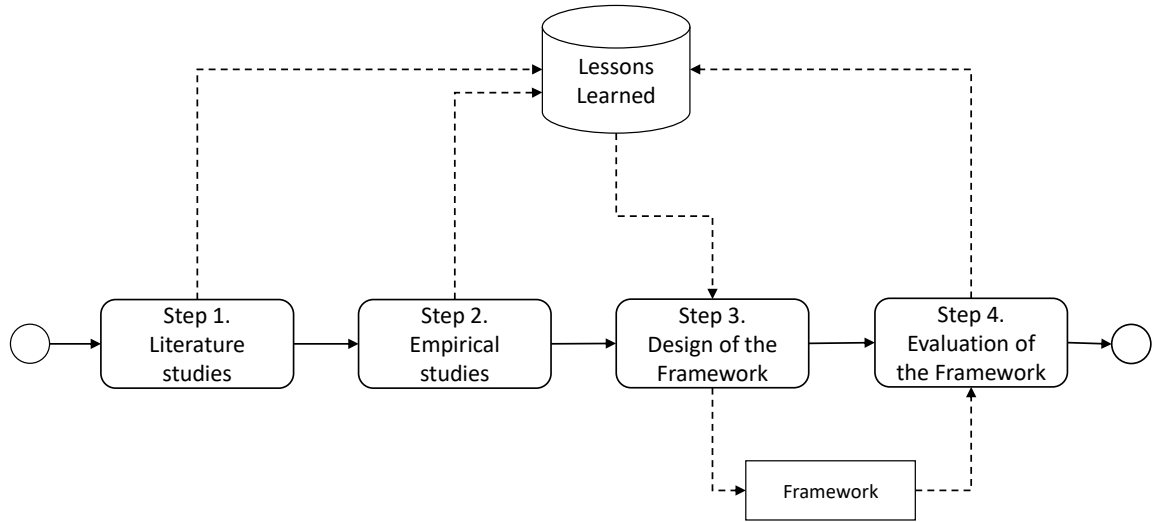
## 1.3 Method

This thesis adopts the design science paradigm. In the design-science paradigm, knowledge and understanding of a problem domain and its solution are achieved in the building and application of new and innovative artifacts [Hevner and Chatterjee, 2010]. Therefore, the design of the proposed framework (the target artifact) is based on a series of investigations and the gradual construction of knowledge.

The study design of this thesis is based in a multi-method approach that combines two or more quantitative or qualitative methods [Hesse-Biber, 2010]. In this thesis, we adopt secondary literature studies, Action-Research, and other empirical approaches. In this approach, data triangulation is used to consolidate results from different methods, in order to answer research questions, and collect data from different sources to increase the confidence on the results [Easterbrook et al., 2008].

Figure 1.1 presents the study design adopted for this thesis. The research was organized in a process with four steps: “Literature studies” (step 1), “Empirical studies” (step 2), “Design of the Framework” (step 3), and “Evaluation of the Framework” (step 4). The first step consists in a series of studies on the literature on PBL and gamification, in order to understand the principles and characteristics of each method, and their application in software engineering education (Chapter 2). The second step consists of studies to understand the practical implications of using gamification and PBL (individually and in conjunction) in software engineering education (Chapters 3 and 4). The third step is the design of the proposed framework (Chapter 5), based on the lessons learned from previous steps. Finally, the fourth step is the evaluation (and subsequent refinement) of the proposed framework (Chapter 6).





**Figure 1.1.** Study design

Steps 1 and 2 contribute directly to the specific goals SG1 (*Investigate how gamification can be used to support software engineering education*) and SG2 (*Investigate how PBL can be used to support software engineering education.*). Steps 2 and 4 contribute to the specific goal SG3 (*Investigate the benefits and drawbacks of the joint use of gamification and PBL to support software engineering education.*).

## 1.4 Contribution and relevance

The main expected contribution of this thesis is the documentation of a framework for gamification of project-based software engineering education. This product is expected to provide educators with a reusable method and guidelines to support the introduction of educational software engineering projects as practical assignments in software engineering related courses, grounded in lessons learned from theory and practice on the use of gamification and PBL.

The relevance of this study relies not only on the growing demand for development of professional competences for undergraduate software engineering students, but also on the recurring challenge of balancing theory and practice in software engineering education. An expected contribution of this research is providing additional options for the introduction of practice in software engineering education, using gamification and PBL. Additionally, a recurrent problem stated in the literature on gamification in education is the shortage of empirical evidences of the use of this technique. Therefore, the relevance and contribution of this thesis also resides in providing empirical data



regarding the use of PBL and gamification.

To the date of production of this document, the following publications were byproducts of this thesis, and contain parts of the thesis results.

- Maurício R. A. Souza, Lucas Veadó, Renata Moreira, Heitor Costa, Eduardo Figueiredo. Games for Learning: Bridging Game-related Education Methods to Software Engineering Knowledge Areas. In 39th International Conference on Software Engineering (ICSE), Software Engineering Education and Training (SEET) track. Buenos Aires, Argentina, 2017. [Souza et al., 2017b]
- Maurício R. A. Souza, Kattiana Constantino, Lucas Veadó, Eduardo Figueiredo. Gamification in Software Engineering Education: An Empirical Study. In 30th International Conference on Software Engineering Education and Training (CSEE&T). Savannah, GA, USA, 2017. [Souza et al., 2017a]
- Maurício R A Souza, Renata Moreira, Eduardo Figueiredo. A Framework for the Gamification of Practical Assignments in SE Education (Poster). In 30th International Conference on Software Engineering Education and Training (CSEE&T), Poster Sessions. Savannah, GA, USA, 2017.
- Mauricio R. de A. Souza, Lucas Veadó, Renata Teles Moreira, Eduardo Figueiredo, Heitor Costa. A Systematic Mapping Study on Game-related Methods for Software Engineering Education. In Information and Software Technology (IST), 2018. [Souza et al., 2018]
- Pedro Rodrigues, Maurício R A Souza, Eduardo Figueiredo. Games and Gamification in Software Engineering Education: A Survey with Educators. in 2018 IEEE Frontiers in Education Conference (FIE). San Jose, CA, USA, 2018. [Rodrigues et al., 2018]
- Maurício R A Souza, Renata Moreira, Eduardo Figueiredo. Playing the project: Incorporating gamification into PBL approaches for software engineering education. In 27º Workshop sobre Educação em Computação (WEI), Belém, PA, Brazil, 2019. [Souza et al., 2019a]
- Maurício Souza, Renata Moreira, Eduardo Figueiredo. Students Perception on the use of Project-Based Learning in Software Engineering Education. In: XXXIII SBES - Education Track. Salvador, Brazil, 2019. [Souza et al., 2019b]



## 1.5 Thesis project outline

In addition to this introductory chapter, this thesis is organized as follows. Chapter 2 describes literature studies about PBL and gamification, and the relevant background and related work. Chapters 3 and 4 present empirical studies about the use of gamification and PBL in software engineering education. Chapter 5 presents the framework proposed in this thesis project. Chapter 6 describes the evaluation of the framework. Further details on each chapter is discussed as follows.

**Chapter 2** presents results of systematic and non-systematic reviews of the literature in order to establish the theoretical foundation for the themes addressed in this research, namely: software engineering education, Project-Based Learning, and gamification. This chapter also discusses related work, and summarizes a set of lessons learned from the literature regarding the use of PBL and gamification in the context for software engineering education. The results of this chapter contribute to the specific goals SG1 and SG2 of this thesis.

**Chapter 3** describes an empirical study on the adoption of gamification in software engineering education. The goal is to understand the perception of students on the use of two game elements (badges and leaderboards) in an introductory software engineering course. The results of this study contribute to the specific goal SG1 of this thesis.

**Chapter 4** presents an Action-Research study describing the adoption of PBL in an introductory software engineering course and reports the lessons learned from this study. To evaluate the approach, the perceptions of 32 students were collected (using questionnaires) and compared to 17 students who also participated in similar practical assignments in a traditional course. The results of this study contribute to the specific goals SG2 and SG3 of this thesis.

**Chapter 5** presents GaPSEE, a framework for the gamification of project based software engineering education. The framework is composed of two elements: a set of guidelines and a process for the definition of practical assignments in the context of software engineering education, using principles of both PBL and gamification.

**Chapter 6** presents the results of five case studies executed in three educational institutions for the evaluation of GaPSEE. The evaluation involves four software engi-



neering lecturers using GaPSEE in five different courses, with a total of 113 students. In addition to the observation of the researcher, the lecturers were interviewed and the students answered questionnaires. The results of these studies contribute to the specific goal SG3 of this thesis.

**Chapter 7** presents the conclusion of this thesis, summarizing the results in relation to the specific goals. We discuss the contributions and implications of this research. We also propose future works as consequence of this study.



# Chapter 2

## Literature Review

This chapter describes literature review studies that aim at defining relevant theoretical foundation for the understanding of this research. Section 2.1 presents the theoretical foundation on software engineering education. Section 2.2 describes the concepts of Project-Based Learning and its challenges in software engineering education. Section 2.3 describes the theoretical foundation on gamification and game elements (based on the results of previous publications [Souza et al., 2017a, 2018]). Finally, Section 2.4 presents related work.

### 2.1 Software engineering education

The “Curriculum Guidelines for Undergraduate Degree Programs in Software Engineering” (or “SE 2014”) [IEEE/ACM, 2015] provides guidance to academic institutions and accreditation agencies about what should constitute an undergraduate course on software engineering. This document defines a body of knowledge suggesting what every software engineering graduate must know. In this thesis, this document is used as the main reference regarding knowledge areas and learning topics of software engineering. According to this document, graduates in software engineering programs should be able to demonstrate the following qualities: “Professional Knowledge”, “Technical Knowledge”, “Teamwork”, “Design Solutions in Context”, “Perform Trade-Offs”, “End-User Awareness”, and “Continuing Professional Development”.

However, the software engineering courses do not seem to succeed in helping students to acquire this sort of skills [Wilhelm et al., 2002; Sancho-Thomas et al., 2009]. Most of them are mainly focused on teaching technical contents [Sancho-Thomas et al., 2009]. One reason is the amount of content that instructors have to cover. For instance, SE 2014 is organized in ten “knowledge areas”, representing particular sub-disciplines of



software engineering that are generally recognized as a significant part of the software engineering knowledge that a graduate should know [IEEE/ACM, 2015].

Table 2.1 provides the acronym, name and a brief description of the seven knowledge areas considered in this thesis, namely Software Modeling and Analysis (MAA), Requirements Analysis and Specification (REQ), Software Design (DES), Software Verification and Validation (VAV), Software Process (PRO), Software Quality (QUA), and Professional Practice (PRF). For instance, Software Process is concerned with software engineering practices used to develop and maintain software components and systems at the individual, team, and organizational levels.

**Table 2.1.** SE education Knowledge Areas [IEEE/ACM, 2015]

Acronym	Name	Description
MAA	Software Modeling and Analysis	Modeling and analysis can be considered core concepts in any engineering discipline because they are essential to documenting and evaluating design decisions and alternatives.
REQ	Requirements Analysis and Specification	The construction of requirements includes elicitation and analysis of stakeholders' needs and the creation of an appropriate description of desired system behavior and qualities, along with relevant constraints and assumptions.
DES	Software Design	Software design is concerned with issues, techniques, strategies, representations, and patterns used to determine how to implement a component or a system.
VAV	Software Verification and Validation	Software verification and validation uses a variety of techniques to ensure that a software component or system satisfies its requirements and meets stakeholder expectations.
PRO	Software Process	Software process is concerned with providing appropriate and effective structures for the software engineering practices used to develop and maintain software components and systems at the individual, team, and organizational levels.
QUA	Software Quality	Software quality is a crosscutting concern, identified as a separate entity to recognize its importance and provide a context for achieving and ensuring quality in all aspects of software engineering practice and process.
PRF	Professional Practice	Professional practice is concerned with the knowledge, skills, and attitudes that software engineers must possess to practice software engineering professionally, responsibly, and ethically.

Although we acknowledge the relevance of the knowledge areas “Computing Essentials”, “Mathematical and Engineering Fundamentals” and “Security”, we did not consider them in this thesis because they are not specific to the context of software engineering education. These knowledge areas cover basic/introductory topics for computing/IT and engineering courses that may be studied in more details in other domains of computer science or information systems degrees. For example, Computing Essentials cover a range of topics from algorithms, data structures, and complexity



(commonly associated to CS1/CS2 courses) to computer organization and operational system basics.

Meira [2015] warns that the undergraduate curriculum is currently fragmented in topics, rather than unified around problems and projects. Therefore, another perspective that can be used to drive software engineering education is toward the expected competences for software engineering professionals. The Software Engineering Competency Model, SWECOM [Ardis et al., 2014], describes competences for software engineers who participate in developing and modifying software-intensive systems. The model is organized in skill areas, skills, and work activities. SWECOM defines that skill differs from knowledge [Ardis et al., 2014]: knowledge is what one knows, while skill is what one can do.

In this model, skills are divided into “Cognitive Skills”, “Behavioral Attributes and Skills”, and “Technical Skills”. However, the “Technical Skills” are the primary focus of SWECOM. Therefore, in this thesis, SWECOM is adopted as a reference model for the identification of relevant skills to be addressed in software engineering education.

SWECOM describes five “software engineering life cycle skill areas”. These skill areas include skills needed to accomplish various work activities within a phase of software development or sustainment. Table 2.2 summarizes the life cycle skill areas and their respective skills. SWECOM also describes eight “software engineering crosscutting skill areas”. These are applied across all life cycle skill areas. Examples of crosscutting skill areas include “Software Measurement” and “Software Configuration Management” [Ardis et al., 2014].

**Table 2.2.** SWECOM Skill Areas and Skills

Life Cycle Skill Areas	Skills
Software Requirements Skills	Software Requirements Elicitation
	Software Requirements Analysis
	Software Requirements Specification
	Software Requirements Verification and Validation
	Software Requirements Process and Product Management
Software Design Skills	Software Design Fundamentals
	Software Design Strategies and Methods
	Software Architectural Design
	Software Design Quality Analysis and Evaluation
Software Construction Skills	Software Construction Planning
	Managing Software Construction
	Detailed Design and Coding
	Debugging and Testing
	Integrating and Collaborating
Software Testing Skills	Software Test Planning
	Software Testing Infrastructure
	Software Testing Techniques
	Software Testing Measurement and Defect Tracking
Software Sustainment Skills	Software Transition
	Software Support
	Software Maintenance



### 2.1.1 Practice in software engineering education

In software engineering education, besides learning theory and acquiring technical skills, students need to develop the ability to apply, evolve, and practice those skills throughout their academic life (Gary, 2015). Additionally, soft skills, such as leadership, teamwork, decision-making, negotiation, and self-reflection, are important abilities for software engineering practice, since software development also involves several human and social aspects [Marques et al., 2014]. Nevertheless, the development of these cross-cutting capabilities is usually less supported in Computer Science programs [Marques et al., 2014].

Traditional approaches (e.g., expository lectures, exams, and complimentary assignments) are still largely used by lecturers (Marques et al., 2014; Bessa et al., 2012; Sancho-Thomas et al., 2009). The teaching of basic concepts and theoretical foundations with no link to their practical applications or no examples in the students' contexts is usual in computing courses [Fioravanti et al., 2018]. However, these teacher-centered educational methods may not support the practical development of competences (Barnes et al., 2008; Sancho-Thomas et al., 2009) and may have limited learning efficiency (Prikladnicki et al., 2009). Therefore, student-centered approaches have shown to be more suited for allowing the development of competences, in learn-by-doing, with a higher motivation from the learner, a more active role in the learning process, and better learning in the application level (Prikladnicki et al., 2009).

The software engineering education community has been striving for the proposal and adoption of alternative learning methods, that support the development of skills, rather than just focusing on theory on technical knowledge. For instance, Marques et al. [2014] performed a systematic mapping study to identify practical approaches for teaching software engineering. The authors identified 173 primary studies describing practical approaches for software engineering education. These approaches include: game learning, case studies, simulation, inverted classrooms, maintenance projects, service learning, and open source development. The authors conclude that there is a clear concern for teaching software engineering involving practical experiences. Bridging the gap between theory and practice is still a major challenge in software engineering education.

SE 2014 IEEE/ACM [2015] recommends including team-based projects into the software engineering and computer science curriculum. The needs of providing real world experience of software development to students are a recurring theme in SE 2014, and several of its guidelines address this matter IEEE/ACM [2015]. For instance, Curriculum Guideline 5 suggests that “students also need practical material to be taught



early so they can gain maturity by participating in real-world development experiences (...)” IEEE/ACM [2015]. Curriculum Guideline 10 also discusses the multiple dimensions of the problem-solving aspect of software engineering, and suggests that “problem solving is better learning through practice and taught by example” IEEE/ACM [2015]. Furthermore, Curriculum Guideline 17 suggests the need of using interesting, concrete and convincing examples to motivate students. Finally, Curriculum Guideline 14 objectively declares “the curriculum should have a significant real-world basis” IEEE/ACM [2015]. In this context, the introduction of software projects in software engineering education (eg.: capstone and project-based courses) have been largely used to address these issues [Delgado et al., 2017; Marques et al., 2018].

## 2.2 Project-Based Learning (PBL)

Project-Based Learning (PBL) is one of the main successful student-centered educational methods broadly used in computing science, information systems and engineering courses [Marques et al., 2018; Macias, 2012; Jazayeri, 2015; Delgado et al., 2017; Shuto et al., 2016; Warin et al., 2016; Yamada et al., 2014]. PBL aims to better prepare students for real-life by getting them to come up with their own solutions through proactive and collaborative means, in a contextualist, collaborative and constructivist learning environment [Thevathayan, 2018]. According to Thevathayan [2018], constructivism is a learning theory in which students actively construct their own mental models from sensory data inputs and existing beliefs, rather than act as passive receivers of knowledge. Fioravanti et al. [2018] characterize PBL as an active learning approach. According to the authors, the goal of active learning is to “provide opportunities for learners to critically think about content through a range of activities that help preparing learners for the challenges of professional situations” [Fioravanti et al., 2018].

In PBL, students are engaged in the investigation of realistic problems, and they learn by working on an open-ended project, discovering problems and finding solutions as they go along [Blumenfeld et al., 1991; Jazayeri, 2015]. In this instructional model, students confront real-world issues and problems that they find meaningful, determine how to address them, and act collaboratively to create solutions [Bender, 2012]. The instructor has a less central role, acting as a guide, and students take more responsibility for their own learning, which results in higher student involvement [Martin et al., 2014; Jazayeri, 2015].

Among the main benefits accredited to PBL, the literature suggests: it facili-



tates authentic learning [Thevathayan, 2018]; provides durable benefits, regarding engagement, integration of methods and techniques learned in different courses, and the development of teamwork [Gary, 2015; Warin et al., 2016]; and ensures that students continuously apply and evaluate sustained interaction experiences [Jazayeri, 2015].

PBL presents many opportunities and challenges when used earlier in the degree program. Students exposed to standard industry processes and tools, such as agile and version control, earlier in the program are more likely to be industry-ready by the time they graduate [Thevathayan, 2018]. However, newcomers may become overwhelmed when asked to make complex design decisions prematurely.

### 2.2.1 Characteristics of PBL

Many software engineering courses use projects as assignments to give students a chance to experience practical problems. However, Jazayeri [2015] states that this is not enough to implement PBL, as these projects are not typically open-ended, and students are expected to follow a very specific path to the solution. Therefore, students are not in charge of learning in these scenarios. In PBL, projects drive the learning process, and learners should face meaningful problems, where they can investigate, apply and reflect on knowledge and skills useful to solve it. The educator moves from the role of knowledge provider, to the role of facilitator, providing meaningful feedback on students' actions, helping students to reflect, and providing sufficient guidance for students to achieve learning outcomes.

Blumenfeld et al. [1991] describe five characteristics that define PBL projects: (1) the project should be driven by a question or problem without a predetermined solution; (2) the project should result in a series of artifacts that culminate in a tangible final product that addresses the driving question/problem; (3) the project should be realistic, grounded in real world problems; (4) the project should allow students to work collaboratively with peers and instructors to construct knowledge; and (5) the project should allow students to have an active voice - i.e., room to decide over how to achieve their goals and negotiate some aspects of the project.

Similarly, Thevathayan [2018] proposes that the main features of PBL are: learning is driven by open-ended ill-structured problems; students are expected to work collaboratively in teams; teacher is expected to play the facilitator role. According to the author, the project component promotes teamwork through discussion of design options, division of tasks, writing of shared reports and team presentations. Additionally, teamwork also allows students to learn firsthand the need to plan and manage the project within cost and time constraints.



Fioravanti et al. [2018] suggest that PBL focuses on real-world problems and challenges and relies on problem solving, decision making and investigative skills. For Bender [2012], PBL projects are focused on authentic problems or issues from the real world and require extensive collaborative work. There are five key elements in PBL [Bender, 2012]:

- Anchor: This is the basis for posing a question;
- Artifacts: Items created within the course of a project that represent possible solutions to the problem or aspects of the solution to the problem;
- Authentic achievement: Represents the emphasis that the learning stemming from these projects should stem from real-world scenarios;
- Driving question: The primary question that provides the overall task or stated goal for the PBL project; and
- Student voice and choice: Students should have voice in project selection and statement of the essential question.

Delgado et al. [2017] organize a list of lessons learned from the evolution of a software engineering course over six semesters. Their recommendations include [Delgado et al., 2017]:

- Team size: teams should have at least 4 members to generate the dynamics and issues that are common on collaborative software endeavors. However, big groups may face coordination problems;
- Project selection: students should be free to select their own projects, rather than imposing one or restricting their selection to a list. It increases students' commitment and excitement levels, and creates a stronger sense of ownership. However, it may be more difficult to ensure that the projects are of similar complexity;
- Project execution: Projects should have identical timelines and iteration schedules. This practice improves monitoring and grading, and keeps students better informed about the progress of the other projects, which promotes a sense of competence throughout the iterations, and encourages the sharing of technical knowledge among teams;
- Tools and technologies: The use of common project management tools across all teams allow for better monitoring and grading. In respect to technologies for the



project development, instructors should define a set of mandatory and optional technologies. The first should be covered in classroom to give students basic conditions for developing their projects;

- **Grading:** The project should have the highest weight in the final course grade, and part of that grade should evaluate the individual contributions of each student. Defining a mechanism to provide individual grades is essential to maintain fairness;
- **Stakeholders:** When it is not possible to have an industry client, the role of the product owner should be performed by someone external to the team, ideally a TA (Teaching Assistant) or the course instructor. The product owner should meet the team periodically and help them to plan the iteration tasks and solve organizational problems;
- **Supporting students:** The technical topics included in the syllabus should support the development of the project directly and be covered early in the semester.

### 2.2.2 Challenges and issues related to the use of PBL in software engineering education

The literature suggests that there are some challenges related to the use of PBL in software engineering education. We grouped these challenges in five categories: “PBL as an educational method”; “setup of PBL courses”; “selection of meaningful projects”; “tracking students progress and learning outcomes”; and “teamwork and different types of learning”. These challenges are further discussed in the following paragraphs.

**PBL as an educational method** suggests a paradigm change in the roles of lecturers and students. For the lecturers, the main challenge is related to the new teachers’ role, where they have to change from knowledge transmitter to facilitator, losing control on the student work [Martin et al., 2014]. Regarding the learner role, Martin et al. [2014] point that “in some academic contexts, students are not used to dealing with this kind of problems, therefore they feel lost and end up rejecting the methodology”.

Additionally, Yamada et al. [2014] discuss the issues related to the assessment of the educational effectiveness of this learning method. Yamada et al. [2014] break down this issue as four problems: obscurity of educational effectiveness, quantitative measurement of the education process, difficulty in quantifying personal characteristics, and difficulty in determining the learning process.



Regarding the **setup of PBL courses**, Martin et al. [2014] argue that teachers find difficulties in designing activities that fulfill the main characteristics of this methodology. This is aggravated by the lack of comprehensive frameworks to support educators in setting up courses using this learning method. Warin et al. [2016] compare the situation of Project-based Learning and Problem Based Learning: for the first, they found no papers in the literature proposing complete methods; for the later, there are well-established generic methods that are broadly applicable. Warin et al. [2016] and Macias [2012] also point the need of tools to support this learning method.

Moreover, preparing and running a PBL course consume time, resources, and effort, both for educators and learners [Harms and Hastings, 2016; Hanakawa, 2015; Nguyen et al., 2013; Marques et al., 2018; Rupakheti et al., 2017; Daun et al., 2016; Gary, 2015; Mäkiö et al., 2017]. For instructors, PBL requires considerable effort to: (i) supervise, guide and mentor students over a significant period of time [Gary, 2015; Daun et al., 2016]; (ii) setup appropriate projects [Rupakheti et al., 2017]; (iii) make preparations for the course, in terms of lecture material, academic examples, or project milestones [Daun et al., 2016]; and (iv) setup of processes, development environments, physical space, and state of the art tools. For students, PBL ensures sustained, long-term participation, what is contrary to burst nature of students [Gary, 2015]. However, they may not be able to devote the necessary time and effort because they may have other subjects to study at the university [Hanakawa, 2015].

All those issues impact on scalability. Scaling PBL is difficult both in terms of student head count and integration across the degree program [Gary, 2015]. For instance, Harms and Hastings [2016] claim that there is a limit to the number of student-led projects that an instructor can manage, and in their experience, using student-led projects with a class of more than 30 students is hard to manage.

The **selection of meaningful projects** is a crucial step in the setup of PBL courses, projects play a central role in the learning process. Therefore the first challenge is the selection of projects that provide good balance in size, complexity and realism. Harms and Hastings [2016] state that “projects need to not be too shallow and yet not be too idealistic either”.

Prioritizing realism, by means of selection of real projects from external stakeholders may lead to specific challenges. Educators must face the problems of the difficulty in establishing partnerships with the industry [Daun et al., 2016]. Additionally, the participation of external people as real customers may be problematic, as their availability and expectations are not controlled. On the other hand, the option of having the educator acting as stakeholder is also challenging [Daun et al., 2016]. It requires a high sensitivity and experience to foresee student challenges, maintain the



acted role throughout the semester, carefully guide the knowledge discovery process in such a way that the students achieve teaching goals and perform satisfactorily. Additionally, it requires appropriate background industry experience. Finally, students have to clearly separate instructions from the lecturer role from statements given by the role of stakeholder.

The choice of projects may impact in the coverage of expected learning outcomes [Nguyen et al., 2013]. Additionally, criteria such as relevancy of the application domain must balance student excitement and relevancy for industry, however, this is a hard choice, as relevancy in this context may be volatile trends [Delgado et al., 2017].

**Tracking students progress and learning outcomes** is also challenging. First, PBL deals with ill-structured problems as central activities [Martin et al., 2014]. In ill-structured problems, one or more of the problem elements are unknown or not known with any degree of confidence. The goals are vague or unclear. There are multiple solutions and solution paths (or even no consensual solution). They present uncertainty about which concepts, rules and principles are necessary. And learners are required to express personal opinion, beliefs or judgments [Martin et al., 2014]. As a result, students may feel lost, and require the right amount of guidance towards learning outcomes. Instructors are required to spend considerable effort in tracking students progress and providing meaningful feedback for students to ensure learning.

Tracking the progress of students projects is not easy, as instructors are required to scrutinize the report by students or participate in projects in order to understand progress of the projects [Fukuyasu et al., 2013]. Consequently, the definition of a strategy to evaluate students is also difficult, as it may require considering qualitative and quantitative data on their performance. If not clearly defined, students may become confused and concerned regarding their assessment. As students direct their activities based on the given assessment criteria, the assessment design plays a key role in what students will focus on [Fagerholm and Vihavainen, 2013].

Ensuring the alignment between the project and learning outcomes requires dealing with complex theoretical relationships in a sound fashion [Daun et al., 2016]. In software engineering, for instance, software process is a crosscutting knowledge, and each of the phases of software development life cycle, is closely related to the other, however each one has a multitude of learning topics. It is difficult to impart the importance of each topic and the impact they have on each other. Additionally, students may become absorbed by a single facet of the project, such as programming [Hanakawa, 2015; Winterfeldt and Hahne, 2014].

Regarding **Teamwork and different types of learning**, some authors [Yamada et al., 2014; Sunaga et al., 2016, 2017] suggest that instructors should plan carefully



about team composition. Teams composed by members with the right complementary skills may improve the learning experience, while others compositions may compromise it. In contrast, students working in teams may be affected by their lack of teamwork experience. Chen et al. [2014] state that “if not equipped with the necessary teamwork skills, these students are like blind explorers trying to find the proper direction in which to take their project”. The authors support that the lack of team experience impedes learning and makes it difficult to obtain quality results.

Kizaki et al. [2014] mention other problems related to teamwork: shortage of communication between members and difference of a member’s technical capabilities. The later, may also impact in the problem of a heterogeneous effort distribution among team members [Kizaki et al., 2014; Nguyen et al., 2013], what may lead to knowledge not being equally distributed [Nguyen et al., 2013]. As a consequence, it becomes even harder to individually assess students progression toward learning outcomes, due “to the ingenuity some students show in hiding behind others’ work” [Gary, 2015].

Finally, students are impacted differently by the teaching approach, as they may have different learning styles [Zhi, 2016]. Therefore, a challenge using PBL is to address different learning styles of students, which may require mixing different learning environments, resources, modes and/or contents.

## 2.3 Gamification

This section describes the results of literature review studies conducted to understand the research on the use of games and related methods in software engineering education, and to further analyze the role of gamification in this specific context. Therefore, this section relies on the results of a systematic review study [Souza et al., 2017a, 2018], and a subsequent study with the specific focus on gamification used to support software engineering activities, both in the educational and professional environments.

Gamification is a relatively new term that has been used to denote the use of game elements and game-design techniques in non-gaming contexts [Deterding et al., 2011]. The goal of gamification is to use the philosophy, elements, and mechanisms of game design in non-game environments to induce certain behavior in people, as well as to improve their motivation and engagement in a particular task [Pedreira et al., 2015].

In the context of education, gamification is one category of game-related methods used to support software engineering education. In the context of this study, the term “game-related methods” refers to any approach that uses games or game elements



for supporting teaching and learning processes in the scope of software engineering education. In our systematic mapping study [Souza et al., 2017a, 2018], we mapped three categories of game-related methods for software engineering education: (i) Game-Based Learning (GBL); (ii) Game Development Based Learning (GDBL); and (iii) gamification.

The term “Game Based Learning” (GBL) has been used to refer to any approach using games for learning purposes. This thesis adopts the definition of GBL used by von Wangenheim and Shull [2009]: the use of game applications for defining learning outcomes. Games are any contest (play) among adversaries (players) operating under constraints (rules) for a goal (winning, victory, or pay-off) [von Wangenheim and Shull, 2009]. Hence, GBL approaches apply games with the purpose of learning specific skills and concepts, usually named as “serious games” (games with purposes), edutainment, or educational games. In this thesis, we did not limit our concept to digital games or to the use of games designed specifically for learning purposes (serious games), we considered any game used in educational context. Problems and Programmers [Baker et al., 2005], SimSE [Navarro and van der Hoek, 2009] and “The Incredible Manager” [Dantas et al., 2004] are examples of GBL approaches for software engineering education.

The key difference between gamification and GBL is that the former deals with creating a game like experience, by incorporating elements of games, in real life contexts or applications, while the latter is the use of full-fledged games for educational purposes.

Wu and Wang [2012] define Game Development Based Learning (GDBL) as an approach where students are required to modify or develop a game as a part of a course using a game development framework to learn skills within computer science and software engineering. In the context of software engineering education, game development brings the fun factor to courses and provides pedagogical aspects of problem-based learning, cooperative learning, blended learning, and experiential learning [Krusche et al., 2016]. When developing a game, students have hands-on experience on software process, design, and other skills related to software engineering.

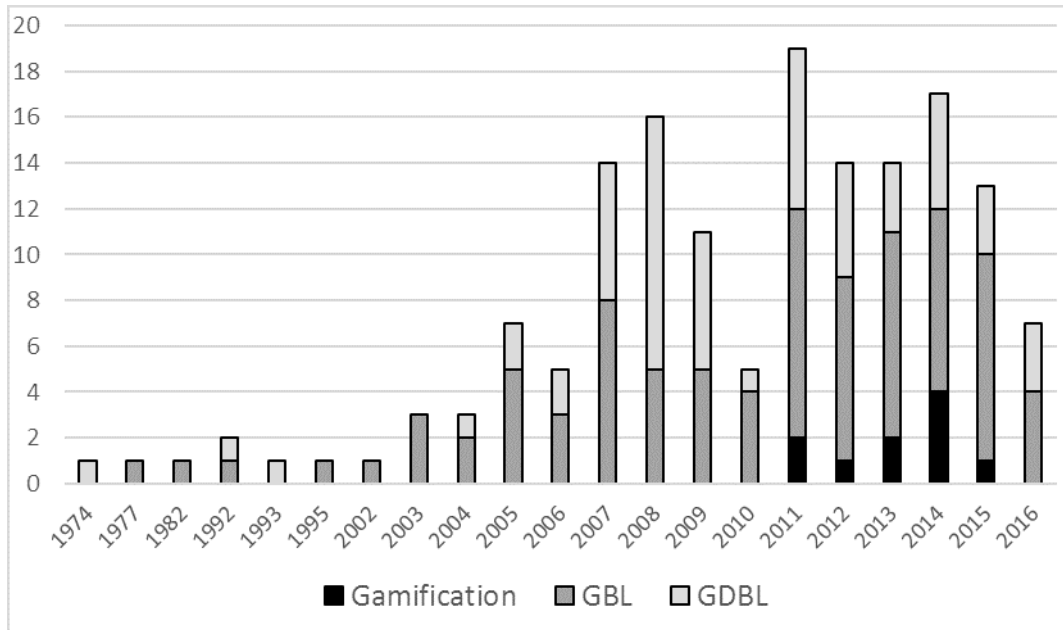
GDBL differs from gamification because the first uses the practice of game development as a practical instrument to expose students to specific practices, while gamification uses elements and philosophy of games to create game like experiences.

Different from GBL and GDBL approaches, by definition, gamification is a technique that requires a non-game context (e.g., the classroom activities, capstone projects, or the use of a tool) in which game elements are introduced. Therefore, in the context of software engineering education, it is not a “stand-alone” educational tool. Gamification is usually used as a device to motivate students in conforming to desired behaviors, such as the more frequent use of specific tools, acquiring the habit



of applying specific techniques, or to increased participation in the classroom.

Figure 2.1 presents a histogram with the frequency of primary studies discussing the use of game-related methods in software engineering education, from 1970 to 2016. While the use of game-related methods to support software engineering education is not a novelty, gamification is a recent trend.



**Figure 2.1.** Timeline of primary studies [Souza et al., 2018].

### 2.3.1 Game elements

Game elements are characteristics of games that can be found in most (but not all) games [Deterding et al., 2011]. They are a key part for any gamification strategy and are present in all gamification strategy. Game elements are also described as a set of components that compose a game [Bedwell et al., 2012]. In some studies, game elements are also called game attributes [Bedwell et al., 2012]. However, there is no standard taxonomy for the terminology and description of game elements. For instance, emblem [Garcia et al., 2017] and badge [Hamari, 2017] are two names for the same game element, which are visual rewards given to the user and identify user achievements in the game.

Dicheva et al. [2015] state that there is not a commonly agreed classification of game design elements. The analysis of the primary studies in our systematic mapping study also supported this claim, as we observed a lack of standard definitions of game elements [Souza et al., 2017a, 2018]. Deterding et al. [2011] identify game elements



in varying levels of abstraction and proposed a classification scheme based on 5 levels (from concrete to abstract): (i) “Game interface design patterns”; (ii) “Game design patterns and mechanics”; (iii) “Game design principles and heuristics”; (iv) “Game models”; and (v) “Game design methods”. Zichermann and Cunningham [2011] categorize game elements into mechanics, dynamics, and aesthetics. Dicheva et al. [2015] propose a classification of game elements in gamified educational contexts organized in two levels: (i) “Game Mechanics” and (ii) “Design Principles”. The former is a combination of the first two levels of Deterding’s classification and refers to the more concrete representation of game elements, such as leaderboards, badges, point, and levels. The latter is a combination of the third and fourth levels of Deterding’s classification and it is concerned with abstract elements used in games, such as Competition and Status. Bedwell et al. [2012] present a taxonomy to define game elements for educational purposes. Werbach and Hunter [2012] propose a pyramid that organizes game elements in three categories: dynamics, mechanics, and components. Dynamics contain the main concepts of a game. Examples of elements in this group are: Constraints, Emotions, Narrative, Progression, and Relationships. Mechanics contain the basic process that directs users to engage with content and continue to drive the action forward. Examples of mechanics are: Challenges, Feedback, Competition, and Cooperation. Components are more concrete elements that are used to implement the former, and are tools that can be employed to motivate user in the environment of interest. Examples are Achievements, Avatars, Badges, Combat, Leaderboards, and Levels.

Several frameworks have been proposed to support the selection of game elements to define a gamification strategy. Examples are: “Six Steps to Gamification” or 6D Werbach and Hunter [2012], 5H2W [Klock et al., 2016], Octalysis [Chou, 2015], and game design frameworks such as MDA [Hunicke et al., 2004].

### 2.3.2 Gamification in software engineering education

The research on gamification to support software engineering is recent. Results of Pedreira et al. [2015] showed that software requirements, software development and software testing are the areas which attracted the greatest interest in the field of gamification. The authors suggest that the existing research on gamification applied to software engineering is very preliminary or even immature, with a majority of publications in workshops or conferences, and few of them offer empirical evidence on the impact of their proposals on user engagement and performance. The authors point out the necessity of further research providing empirical results about the effect of gamification. Examples of studies showcasing the adoption of gamification in profes-



sional activities include gamification of the software development life cycle [Dubois and Tamburrelli, 2013] and software process improvement initiatives [Herranz et al., 2014].

Dal Sasso et al. [2017] and Garcia et al. [2017] propose frameworks for the gamification of software engineering activities. The work of Dal Sasso et al. [2017] provides a set of basic building blocks to apply gamification techniques, supported by a conceptual framework. Garcia et al. [2017] propose a complete framework for the introduction of gamification in software engineering environments. This framework is composed of an ontology, a methodology guiding the process, and a support gamification engine.

In the context of software engineering education, gamification approaches are usually applied to create innovative learning experiences, more focused on engaging and motivating learners to perform desired behaviors [Souza et al., 2017a, 2018]. We observed two different strategies for the use of gamification in software engineering education: (i) gamification of the classroom experience and (ii) gamification of specific software engineering activities. Gamification of the classroom experience refers to the use of game elements to engage and motivate students in performing learning activities [Uskov and Sekar, 2014; Laskowski, 2015]. For instance, Laskowski [2015] describes the experiment of implementing gamification techniques into software engineering and service-oriented architecture courses using Points, Leaderboards and Badges to promote competition and, consequently, to motivate students in the class activities. The strategy is focused on the gamification of the classroom activities, and not specific software engineering related activities.

Gamification of specific software engineering activities, on the other hand, applies game elements to motivate learners in practicing specific skills or performing specific practices [Akpilat and Slany, 2014; Singer and Schneider, 2012; Long et al., 2011]. For instance, Singer and Schneider [2012] describe an experiment in which students are encouraged to make more frequent commits to a version control system in a software project course. The authors proposed a leaderboard based on the number of commits from each student and established milestones/thresholds that would trigger messages congratulating students and teams that reached the specified number of commits. Akpilat and Slany [2014] uses weekly challenges to motivate students on applying eXtreme Programming practices to their project. In this scenario, students compete for a “challenge cup” award.

In comparison to GBL and GDBL approaches, few researchers explored gamification in software engineering education, which still is an open field for more experiments and proposals. This is supported by the small number of primary studies found in our systematic mapping study [Souza et al., 2017a, 2018] and in the results of Alhammad and Moreno [2018], which have found 10 and 21 primary studies respectively, with an



intersection of 8 primary studies.

In software engineering education, gamification is used as a strategy to induce learners to use specific software engineering abilities or practices, by promoting competition or systematically rewarding learners as they perform expected actions or show expected behaviors Souza et al. [2018]. Therefore, gamification is a relevant strategy to support students in developing an appreciation of the importance of continued learning and in acquiring habits for professional software development [Souza et al., 2018].

Findings of Alhammad and Moreno [2018] show that the most positively affected aspect was student engagement. Nonetheless, the authors point that the primary studies indicate that students' overall performance and learning outcomes were also found to have more positive than negative results, and it also improved the adoption of software engineering best practices. The authors also discuss that gamification does not necessarily require the implementation of costly tools and frameworks.

Regarding the challenges of gamification in software engineering education, a recurring concern is the difficulty of adapting gamification to each context [Souza et al., 2018]. Each context requires great effort from the educators to setup game elements appropriately, and still there is a chance of failure. This issue becomes even more challenging because of the lack of a systematic approach for gamifying software engineering education. That is, we need a set of systematic steps that software engineering educators can follow to gamify their courses [Alhammad and Moreno, 2018]. Another related issue is the lack of detailed justification for the selection of specific game elements in the published reports [Alhammad and Moreno, 2018], what makes difficult to systematically replicate and adapt the approaches described. These problems may lead to another issue: the sub-utilization of gamification, such as believing that the technique is simply creating a “pointsification” system, i.e., turning gamification into simply using points, badges and leaderboards as the core of the experience Fuchs and Wolff [2016]; Souza et al. [2018]; Alhammad and Moreno [2018].

Assessing the impact of gamification is also difficult [Souza et al., 2018]. There is still a relatively small quantity of empirical data to support generalizations, which leads to the need for more empirical data. The studies in the field usually lack important details that may support replication of studies and generalization of results [Alhammad and Moreno, 2018], such as the context of the courses, background of students and so on.

Finally, if not carefully planned, gamification may have negative impacts on learning Souza et al. [2018]; Alhammad and Moreno [2018]; Dal Sasso et al. [2017]. If not gradually introduced, gamification may lead to extra cognitive workload for students, which in turn may cause confusion or frustration, preventing students from fully un-



derstanding and enjoying the “game” Alhammad and Moreno [2018]. Similarly, there is the risk of students trying to “game the system”, i.e., students might become more engaged in exploiting the rules to “win the game”, than following the expected flow of activities or achieving learning goals Souza et al. [2018]. Dal Sasso et al. [2017] supports this observation adding that “gamifying an already interesting activity may move the focus from the activity itself to the reward system” [Dal Sasso et al., 2017]. The approach proposed by Grant and Betts [2013] showed that many new users work intensively to acquire the easiest badges as quickly as possible, with increased user activity immediately before the awarding of a badge and a strong activity decrease in the period afterwards.

### 2.3.3 Cases of use of gamification in software engineering education

Diniz et al. [2017] use gamification to orient and motivate undergraduate students to contribute to open source software projects. The authors used four game elements: quests, points, ranking and levels. Their approach was assessed with 17 students contributing to a real open source project. Their results show the approach succeeded in motivating and orienting newcomers to collaborate to open source projects. They observed that the “quest” element was specially useful in guiding students and keeping them invested, while points helped by providing feedback to students.

Akpolat and Slany [2014] use weekly challenges to motivate students on applying eXtreme Programming practices to their project. The students had to compete for a “challenge cup” award. Bell et al. [2011] expose students to software testing using a game-like environment, HALO (Highly Addictive, socialLly Optimized) Software Engineering. HALO uses MMORPG (Massively Multiplayer Online Role-Playing Game) motifs to create an engaging and collaborative development environment. Students performed software testing tasks as “quests” contextualized in a fictional storyline. When students complete “quests”, they gain “experience points” and they “level up”. Players gain social rewards (titles and levels) when they complete achievements (such as successfully closing over 500 bugs). Students can track their progress and choose the quests they want to perform in any order. Quests can be achieved alone or requiring a developer to form a team and work collaboratively towards their goal.

Long et al. [2011] describe eight MMORPG elements incorporated in a project based software engineering capstone course: (i) Narrative Context (Epic Story); (ii) Feedback; (iii) Reputations; (iv) Rank, and Levels; (v) Marketplaces and Economies; (vi) Competition under explicit and enforced rules; (vii) Teams; and (viii) Time Pres-



sure.

Laskowski [2015] described an experiment of implementing gamification techniques into software engineering and service-oriented architecture courses. In the first course, authors used Points and Leaderboards and promoted competition. In the second course, the authors used Points, Leaderboards, and Badges. Additionally, the authors adopted a physical representation for Points, in the form of Poker Chips.

Uskov and Sekar [2014] propose the incorporation of over 20 gamification elements in modern software engineering courses. The authors organize gamification elements in three categories: (i) Progression Gamification Techniques; (ii) Feedback Gamification Techniques; and (iii) Behavior Gamification Techniques. However, the authors did not provide enough detail about which elements they used in a pilot study briefly described in the study.

### 2.3.4 Game elements used in the gamification of software engineering

In order to understand how gamification is being used in the context of software engineering education, we analyzed which game elements are used in each primary study (Table 2.3). Leaderboards, Points, and Levels are the most recurrent game mechanics found (5, 4, and 4 primary studies, respectively). Competition is the most recurring game design principle identified (4 primary studies). These results are further explained in our literature review [Souza et al., 2018].

**Table 2.3.** Game elements used in SE education context[Souza et al., 2018]

Game Element	Quantity
Leaderboards	5
Points	5
Milestones, Levels, Paths and Progress	5
Competition	4
Collaboration, Altruism, Teams	4
Rewards	3
Challenges	2
Quests	2
Awards	1
Time Pressure	1
Gifts & Sharing	1
Status	1
Badges	1
Feedback	1

The study of Alhammad and Moreno [2018] supports our findings, stating that leaderboards, points and levels were found to be the most frequently used game elements with around 73% of the primary studies examined adopting one or more of



these elements in their gamified framework. Their results also show that challenges, feedback and rewards were the most adopted mechanics, and progression was the most used dynamic. The authors go further suggesting that the approach used to select which gamification elements to use for which purpose appears to be unclear. Finally, Alhammad and Moreno [2018] analyze patterns in the joint use of game elements, observing that points and leaderboards are not only the most used game components, but they are also the most often combined gamification components. Out of the 12 papers that implemented points in their gamified solution, 10 papers accompanied points with a leaderboard [Alhammad and Moreno, 2018]. The combination of quests and achievements were also a common pattern of components, and challenges and feedback were the two game mechanics most often used together.

In a subsequent investigation, we combined the primary studies found in our study [Souza et al., 2018] and in the study of Pedreira et al. [2015], and surveyed the literature on gamification in software engineering (both in educational and professional contexts), for a total of 58 primary studies (listed in Appendix A). Table 2.4 shows the game elements identified in these studies. Points, badges and leaderboards were the most recurring elements. Dal Sasso et al. [2017] refer to these elements as the “*PBL Triad*”, and explain they are the most common form of feedback used in games. Dal Sasso et al. [2017] suggest that these elements are widely used in gamification systems, because they appear to work moderately well as extrinsic motivators.

However, Dal Sasso et al. [2017] warn that creating meaningful gamification environments is “a far from trivial endeavour”. According to the authors, selecting and adapting game elements in a thematic environment is a strongly iterative process. The authors also mention the risk of falling into “pointsification”, and go further alerting of the risk of “*stalling*”, i.e. “if the gamification layer is not constantly revisited, maintained, and evolved, it risks to quickly become obsolete, and therefore will not only be ignored by the users, but it might even cause decreased participation” [Dal Sasso et al., 2017].

## 2.4 Discussion of literature gaps

To the best of our knowledge, no other work has proposed a framework for the joint adoption of gamification and PBL in the context of software engineering education. Alhammad and Moreno [2018] explain that there is a lack of systematic approaches to support the use of gamification in this context. However, we found related work in the specific areas of gamification and PBL in software engineering [Long et al., 2011;



**Table 2.4.** Game elements to support software engineering practice and education

Element	Description	Count
Points / Experience Points (XP)	Feedback mechanics and track progress. Award based on achievement or desired behavior.	32
Badges / Achievements	Given when users complete some specific goal.	22
Leaderboards / Ladders	Show users how they compare to others and others can see them.	19
Quests / Tasks / Missions	Set specific objectives that users must complete in order to obtain points and rewards.	18
Progress / Feedback	Necessary to give users a measure of progress or feedback. It can have many forms and have many mechanics.	17
Levels / Progression	Show users where to go next, allowing them to prepare for what is coming.	17
Rewards / Prices	Something to give when a task is completed successfully.	16
Challenges	Set specific challenges that users can try. Usually more difficult than the normal tasks, but they give better awards.	11
Guilds / Teams	Allow users to form groups to help each other.	10
Ranking / Scores	Show users the top users that have most points or scores.	9
Social Status	Give users a greater visibility. Often used with leaderboards.	9
Competition	Allow users to compete with each other and challenge themselves, winning rewards.	8
Customization / Avatars	Allow users to create and customize their avatar or the environment.	8
Narrative / Story	Linked with the theme, it is a story to strengthen the understanding of your gamification.	6
Branching Choices	Give users the opportunity to choose their own path.	5

Singer and Schneider, 2012; Akpolat and Slany, 2014; Laskowski, 2015; Delgado et al., 2017; Diniz et al., 2017; Rupakheti et al., 2017].

In the scope of gamification, there are several frameworks to support the creation of gamified experiences for general purposes [Mora et al., 2015; Chou, 2015; Hunnicke et al., 2004; Klock et al., 2016]. We have also found proposals of frameworks to support gamification of software engineering activities [Dal Sasso et al., 2017; Garcia et al., 2017]. Nevertheless, these frameworks do not focus on software engineering education activities. In the specific context of software engineering education, we have only found reports describing experiences of using gamification [Laskowski, 2015; Long et al., 2011; Singer and Schneider, 2012; Akpolat and Slany, 2014; Diniz et al., 2017]. However, as stated by Alhammad and Moreno [2018], many of these reports do not provide sufficient details to allow a systematic replication of their proposals.

Similarly, the related work on the use of PBL are focused on describing the implications and benefits of the use of this educational method in software engineering education, but do not provide objective instructions on how to replicate their approaches. For instance, Rupakheti et al. [2017] describe their lessons learned in experimenting with PBL for a software requirement course, and Delgado et al. [2017] describe their lessons while evolving a software engineering course over six semesters.



## 2.5 Final remarks

This chapter described the fundamental concepts used in this thesis, and provided an overview of the literature on Project-Based Learning and gamification in the context of software engineering education.

Our main findings from the literature review on the use of PBL are:

- PBL is based on constructivist and active learning theories. Therefore, its focus is on learn-by-doing, and have students taking an active role in the learning process. Therefore, instructors should step aside to a supportive role;
- PBL projects should be realistic and culminate in tangible products. The project should be central to the learning experience;
- PBL should encourage teamwork and collaboration between students and instructors. However, tracking individual performance is important;
- PBL should give students sufficient room to decide and to negotiate aspects of the projects;
- There are several issues and challenges related to PBL as a learning method, to the setup of PBL courses, to the selection of meaningful projects, to tracking students progress and learning outcomes, and to manage teamwork and consider different profiles of students.

Our main findings from the literature review on the use of gamification are:

- Research on gamification in software engineering education is still preliminary and more empirical studies are necessary to assess its relevancy for this area.
- Gamification is a promising area for further research, and software engineering and development activities are suitable scenarios for gamification.
- Gamification is not a “stand-alone” educational tool. Gamification is more used as a device to motivate and engage students in conforming to desired behaviors, than to objectively make students learn.
- We identified two approaches for using gamification in software engineering education: the gamification of the classroom experience, and the gamification of specific software engineering practices. While the former is concerned with motivating and engaging students in classroom activities, the latter is more concerned



with promoting the use of specific software engineering abilities and practices, or with the development of specific skills.

- Points, Badges, Levels, and Leaderboards are the most recurring game elements used for the gamification of software engineering. However, relying solely on these elements may turn a gamification approach into a mere “pointsification” system.
- Designing a gamification strategy requires effort. Game elements need to be selected and adapted appropriately for each context. However, there is a lack of systematic approaches for gamification of software engineering education, and the reports do not provide enough details to support replication of their approaches.
- When designing gamification approaches for software engineering education, designers should be cautious about: not creating excessive additional cognitive workload; gamification of meaningful aspects of the learning process, to mitigate the risk of students deviating their focus from the learning goals; not stalling.

These findings are inputs for empirical investigations on the use of these methods in software engineering courses (Chapters 3 and 4) and for the design of the GaPSEE framework (Chapter 5).



## Chapter 3

# Empirical Study on the Use of Gamification in Software Engineering Education

This chapter describes our experience and provides an evaluation of the adoption of gamification in a 60-hour introductory course on software engineering at Federal University of Minas Gerais (UFMG), in Brazil, during the second semester of 2016. The goal of this study is to provide an initial understanding of how gamification fits in the software engineering education context. In the context of this thesis, this study aims to provide better understanding on the use of gamification in software engineering education. Therefore, this study supports the specific goal SG1 of this thesis (*Investigate how gamification can be used to support software engineering education*).

In this study, we introduced two elements of games, namely badges and leaderboards, to engage students and promote a safe competition environment by rewarding students for their achievements. While badges award students by specific achievements, a leaderboard aims to indicate the overall performance of every student with respect to their peers. We introduced these game elements using the course webpage, where students could have regular feedback. Therefore, considering the two main categories of gamification approaches found in Chapter 2, gamification of the classroom experience and gamification of specific software engineering activities, we opted for the first, as it demanded less effort and preparation to implement in the course.

We evaluated the proposed course features in two steps. First, we applied a survey to 18 volunteer students to collect their feedback about the impact of badges and leaderboards during their course experience. Second, we interviewed six randomly selected students for a deeper investigation about their perceptions about the course



and the use of game elements. Our results show that badges had a greater impact on the motivation of students than leaderboards. With respect to the number of participants interviewed, it is important to highlight that the focus of this study is to report our experiences and observations, rather than validating hypotheses.

The remainder of this chapter is organized as follows. Section 3.1 describes the gamification elements introduced in the software engineering course under investigation. In Section 3.2, we describe the design of this study. Section 3.3 presents the results of the study. In Section 3.4, we discuss our findings regarding the research questions defined for this study. Section 3.5 discusses the threats to the validity of the study. Section 3.6 concludes this chapter.

### 3.1 Course setup

The introductory software engineering course (“SE Course”, henceforth) aims to introduce students to the concepts and methods required for the development of large software intensive systems. Its prerequisite is familiarity with object-oriented programming, demonstrated through a successful completion of the Modular Programming course at UFMG (or equivalent course in another university).

This software engineering course is mainly based on two textbooks: Software Engineering by Sommerville [2010] and The UML User Guide by Booch et al. [2005]. The course syllabus includes: software development process, agile methods, software requirements analysis and specification, software design, system implementation and testing, software reuse, and software quality. Previous instances of this course had already been the target of experimental studies on alternative educational methods [Figueiredo et al., 2014; Fernandes et al., 2016]. In the second semester of 2016, we planned the inclusion of game elements in the course format. Specifically, we adopted badges and leaderboards. A badge is a common element in games to recognize specific achievements of players [dos Santos et al., 2018b]. Leaderboard is recurring element in games to foster competition, allowing players to compare their progress or performance against other players [Dicheva et al., 2015; Zichermann and Cunningham, 2011]. Both elements capitalize on principles of social status and reputation.

In the context of the SE Course, we used badges to recognize specific actions of students. We established eight badges in this first iteration of a gamified course. Examples of badges are listed below.





- **Agility and Precision:** Awarded to the student who first submitted a correct solution for a specific practical task, such as, code implementations, homework,



or practical assignment.

- **Clean Code:** Awarded to students who used code standards explained in the classroom to document the code and keep it easy to read.
- **Performance Improvement:** Awarded to the student who had the greater improvement in grade from the first to the second exam.
- **Online Participation:** Awarded to students who accessed all online material of this course.

In the first day of class, the instructors communicated students about the existence of badges, but they did not provide details on how to obtain them. Whenever a student met the criteria to receive a badge, all students were communicated in the classroom and the badge was revealed in the course website with the name of students who obtained it. Figure 3.1 shows examples of badges awarded to students during the course. Additionally, these badges did not provide direct bonus grades or any advantage in the course, apart from achievement recognition. However, the attendance to the badges criteria would positively contribute to a better performance on the course assignments and activities.

Badge	Achievement	Student Name
	Agility and Precision	Jane Roe
	Performance Improvement	John Doe
	Clean Code	Jane Doe
	Online Participation	John Roe

**Figure 3.1.** Examples of four badges in the SE course.

Leaderboard was incorporated with two resources: (i) a chart with partial grades and (ii) a Hall of Fame. The chart with partial grades was a digital document that was updated periodically and accessible in the course website, where students could track their grades and compare their performance against other students. To preserve the anonymity, the students were identified only by their university registration number. The “Hall of Fame”, on the other hand, is a special page in the course website that






acknowledges the names of the top three best students of each course semester and the top ten students of all times.

Table 3.1 shows the Hall of Fame with the top-10 best grades of all time in the SE course. The second column of this table indicates the student grade in 100 points. The third and fourth columns indicate the semester and the student name, respectively. Although Table 3.1 presents the best scores since 2011, this table was only made available in the second semester of 2016 based on historical data. The best students of each class, and the best students of all times were also awarded with badges. Figure 3.2 shows the badges exhibited in the Hall of Fame for the top 3 students of all time.

**Table 3.1.** Hall of Fame with the top ten students of all time in the SE Course

Position	Grade	Semester	Student
1	96.50	2016-2	Student A
2	93.70	2011-1	Student B
3	93.50	2016-2	Student C
4	91.55	2016-1	Student D
5	91.25	2013-2	Student E
6	90.75	2013-2	Student F
7	87.04	2012-2	Student G
8	86.79	2014-1	Student H
9	86.50	2015-2	Student I
10	85.97	2014-1	Student J

Position	Grade	Semester	Student
	96.50	2016-2	Student A
	93.70	2011-1	Student B
	93.50	2016-2	Student C

**Figure 3.2.** Badges exhibited in the Hall of Fame for the top 3 students of all time.

## 3.2 Study settings

This section explains how we planned and executed this study. Section 3.2.1 presents the study goal and research questions while Section 3.2.2 discusses the research method we adopted. Section 3.2.3 presents the survey and Section 3.2.4 explains the structure of an interview with 6 randomly selected students.



### 3.2.1 Study goals and research questions

The goal of this study is to investigate how the use of gamification could contribute to motivate students in software engineering education. To achieve this goal, we formulated two Research Questions (RQ) presented below.

**RQ1.** What are the student perceptions on the use of badges in the SE Course?

**RQ2.** What are the student perceptions on the use of leaderboards in the SE Course?

### 3.2.2 Study design and research methods

To answer the research questions, we adopted two techniques. First, we conducted a survey with the students to collect general impressions on the course (Phase I). Second, we conducted interviews to further understand the perception of the students about the gamification techniques used in the course (Phase II). The interviews and questionnaire for the survey were in Portuguese. Therefore, any transcriptions presented in this document were translated by the authors.

For both phases, the target population was all 36 students enrolled in the SE Course. They were invited to participate in both studies by e-mail. To reduce possible bias, the students were instructed that the participation in the survey and in the interviews was not compulsory and this participation did not provide any benefits in grades. Besides that, the student names were not revealed to the course professor during the data analysis, to ensure that students would not be embarrassed for giving negative responses.

### 3.2.3 Planning of the study phase I - Survey

Survey is an empirical strategy for collecting information from or about people to describe, compare or explain their knowledge, attitudes, and behavior using questionnaire or checklist [Pfleeger and Kitchenham, 2001]. In the first phase of our study, a survey was applied to collect a quantitative perspective of the students' perception on the use of badges and leaderboards in the SE course.

We created a questionnaire on Google Forms <sup>1</sup> with two parts: the first one was composed of 4 questions about the background of the students; the second part had 8 questions about the perception of the students about the gamification elements used in the course. Table 3.2 summarizes the items of the questionnaire. The background questions were named BQ1 to BQ4, while the questions of the second part of our survey

---

<sup>1</sup><https://www.google.com/forms/about/>



were named SQ1 to SQ8 (“ID” column). The second column (Questions) describes the questions and the third column (Type of answer) describes the possible answers for each question. Invitations were sent by e-mail to all 36 students formally enrolled in the course, as described in Section 3.2.2.

**Table 3.2.** Questionnaire

ID	Questions	Type of answer
BQ1	Are you familiar with the term “Gamification”?	Likert Scale: (1) Definitely not; (2) Not; (3) Indifferent; (4) Yes; (5) Definitely yes
BQ2	How often do you play games?	Likert Scale: (1) Never; (2) Few times a year; (3) Few times a month; (4) Few times a week; (5) Everyday
BQ3	Do you like playing games?	Likert Scale: (1) Definitely not; (2) Not; (3) Indifferent; (4) Yes; (5) Definitely yes
BQ4	If you play games, what are the main reasons you play games? (multiple options allowed)	Multiple Choices: <input type="checkbox"/> For skill development; <input type="checkbox"/> For the challenges; <input type="checkbox"/> For the fun; <input type="checkbox"/> For the competition; <input type="checkbox"/> To enjoy spare time.
SQ1	Did you find relevant the use badges to reward individual achievements of students during the course?	Likert Scale: (1) Definitely not; (2) Not; (3) Indifferent; (4) Yes; (5) Definitely yes
SQ2	Did you feel motivated to perform better in order to be awarded a badge?	Likert Scale: (1) Definitely not; (2) Not; (3) Indifferent; (4) Yes; (5) Definitely yes
SQ3	Would you like to see more badges for other achievements in this course?	Likert Scale: (1) Definitely not; (2) Not; (3) Indifferent; (4) Yes; (5) Definitely yes
SQ4	Would you like to see the use of badges to reward individual achievements in other courses?	Likert Scale: (1) Definitely not; (2) Not; (3) Indifferent; (4) Yes; (5) Definitely yes
SQ5	Did you find relevant the use the “Hall of Fame” method?	Likert Scale: (1) Definitely not; (2) Not; (3) Indifferent; (4) Yes; (5) Definitely yes
SQ6	Did you feel motivated to improve your performance in the course due to the “Hall of Fame”?	Likert Scale: (1) Definitely not; (2) Not; (3) Indifferent; (4) Yes; (5) Definitely yes
SQ7	Would you like to see similar “Hall of Fame” method in other courses?	Likert Scale: (1) Definitely not; (2) Not; (3) Indifferent; (4) Yes; (5) Definitely yes
SQ8	Do you have any suggestions or criticisms regarding the badges and “Hall of Fame” methods used during the course?	Open answer

### 3.2.4 Planning of the study phase II - Interviews

An interview is a research method defined by a conversation where questions are asked and answers are given [Wohlin et al., 2012]. In this study, we used interviews to complement and deepen the results observed in the survey (Phase I), providing a qualitative perspective on the students’ perception of the gamification elements introduced in the SE Course.



As described in Section 3.2.2, we sent invitation e-mails to all 36 students formally enrolled in the course. In the invitation email, we made it clear that the participant would have their personal data kept anonymously. We interviewed participants individually, face-to-face or by video-conference, as they preferred in order to make the situation more comfortable and natural for them. The interviews were executed after the conclusion of the course, and students were informed that the course instructor would not have access to the names of the participants, to reduce possible bias. Table 3.3 describes the interview script. This script is composed of 9 questions (“Questions” column), named IQ1 to IQ9 (“ID” column). For instance, the first question (IQ1 in Table 3.3) asks students if they track their partial grades during the course.

**Table 3.3.** Interview script

ID	Questions
IQ1	Did you follow up on your partial grades during the course?
IQ2	Do you usually compare your grades in the course with the grades of your colleagues? Why?
IQ3	What is your opinion about comparing your grades with the grades of your colleagues? Is it positive or negative? Why?
IQ4	Do you feel motivated to perform better when comparing your performance with other colleagues?
IQ5	What did you think of the “Hall of Fame” feature to keep track of the top performers in the course in each semester?
IQ6	Did you receive any badges during the course? Do you think this kind of recognition of student achievement is relevant?
IQ7	Did/would you feel motivated for receiving badges for your actions?
IQ8	Do you believe that if the criteria for badges were known, would you work harder to obtain them?
IQ9	Do you believe that there should be more badges during the course? Give examples of other kind of badges.

### 3.3 Results

In this section, we discuss the results of the study. Section 3.3.1 presents the descriptive analysis of the results from the survey (Phase I). In Section 3.3.2, we discuss the qualitative results of the interviews (Phase II).

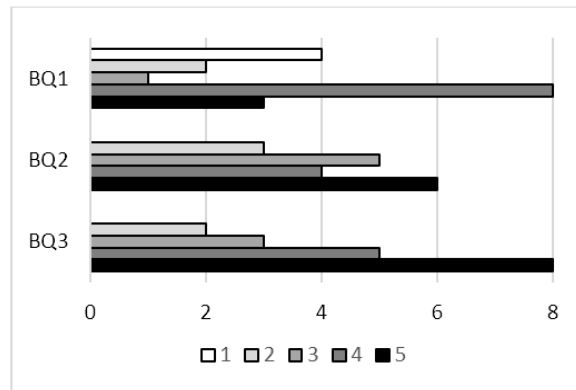
#### 3.3.1 Study phase I – Survey results

From the 36 students in the SE course, 18 participants answered the survey. Of these students, 17 were undergraduate students of the Information System course and only 1 was an undergraduate student in the course of Computer Science.

Figure 3.3 presents the results for the questions BQ1 to BQ3 regarding the background of the participants. The first question (BQ1) was about the participant fa-

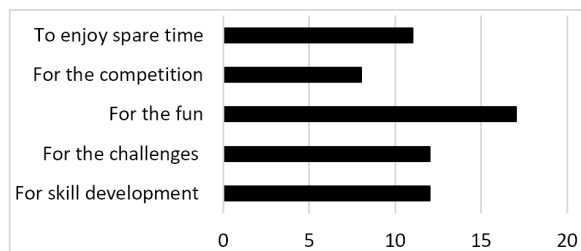


miliarity with the term “gamification”. 11 participants (61.1%) answered that they were familiar with this term (8 were somewhat familiar and 3 were definitely familiar). Only 4 participants (22.2%) claimed that they were definitely not familiar with the term. Our second question (BQ2) was about the frequency in which participants play games. Six participants (33.3%) claimed that they play games every day, 4 participants (22.2%) play a few times a week and 5 participants (27.7%) play sometimes a month. Only 3 participants (16.6%) claimed they play games only a few times a year. No participant claimed to never plays games. In the third question (BQ3), we inquired the participants about their appreciation for games. The results show that most of participants (13 – 72.2%) confirmed that they like playing games.



**Figure 3.3.** Results for the survey background questions BQ1 to BQ3.

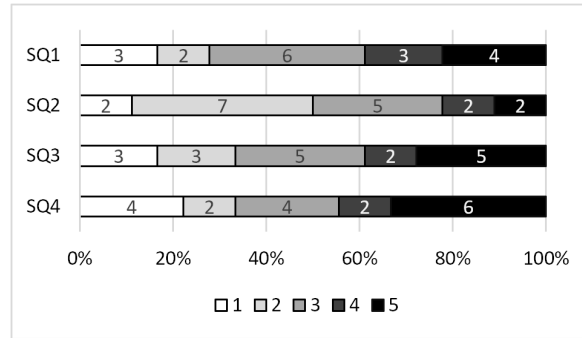
Figure 3.4 shows the results for the last background question (BQ4) in this pre-questionnaire. BQ4 aims to understand the reasons why the participants like playing games. In total, 17 participants (94.4%) stated that they aim to have fun. In addition, 12 participants (66.6%) claimed that one reason is to develop skills and the same number of students said they like to face challenges when play games. Less frequently, 11 participants (61.1%) aimed to enjoy spare time and 8 participants (44.4%) aimed for competition.



**Figure 3.4.** Results for the survey background question BQ4.



Figure 3.5 presents the responses on the perception of students about the gamification elements introduced in the course. Questions SQ1 to SQ4 (described in Section 3.2.3) were defined to investigate the students' perception on the implementation of badges in the course.



**Figure 3.5.** Survey results on the students perception on the use of badges.

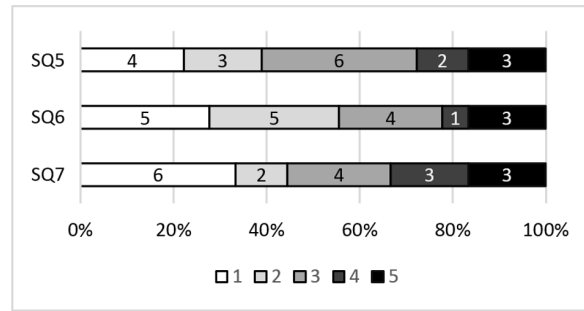
As described in Section 3.1, eight badges were implemented in the SE course. When participants were asked if they found the use of badges relevant (SQ1), the responses were slightly positive (i.e., 7 positive, 6 neutral, and 5 negative responses as presented in Figure 3.5). However, when asked if these badges motivated them towards a better performance in the course (SQ2), Figure 3.5 shows that the responses were negative (9 negative, 5 neutral, and 4 positive responses). When asked if they would appreciate the existence of more badges in the course (SQ3), or the existence of similar resource in other courses (SQ4), the responses were positive in both cases. This data indicates that badges were well received by students, but they were not seen as a key factor of motivation by the majority of them. Further investigation about these results was explored in the interviews.

Regarding the leaderboards, specifically the “Hall of Fame”, the survey results were generally negative, as seen in Figure 3.6. The questions SQ5 to SQ7 inquired participants about the relevance of such resource (SQ5), about how it motivated them to achieve better performance in the course (SQ6), and about the relevance of such resource in other courses (SQ7). Although there were very positive responses for the three questions, the negative responses were dominant. In the Phase II of this study, we investigated the reasons of such negative perception.

Finally, we received 8 responses for the open question SQ8, regarding suggestions and criticism on the use of the game elements introduced in the SE course. Two participants suggested the addition of more badges as transcribed below.

*“There should be more badges throughout the course”*





**Figure 3.6.** Survey results on the students perception of the “Hall of Fame”.

*“In general, I always work hard on the courses I’m enrolled. Thus, in the software engineering course I felt that my dedication was recognized. However, there should be more badges in the course for different types of activities in order to try to reach as many students as possible. In my opinion, the students who have also worked hard, but have not earned any badges, could be discouraged and feel that their effort was not recognized.”*

Three participants suggested that badges could be converted to bonus grades:

*“It was not mentioned by the professor whether anyone who won badges or appeared in the ‘Hall of Fame’ would earn more grades for this. If I knew this could happen, I would be more motivated.”*

*“Badges could be converted into extra grades. In my opinion, perhaps it is the only way to really motivate students.”*

*“(...) Why there are no awards in grades? (...)”*

Five students suggested that the course instructor should provide more details on how to obtain the badges, so students could pursue them proactively:

*“Maybe knowing what badges I could have obtained, it would make me motivated to “work” to get them. We did not have much incentive.”*

*“I think if we knew the titles or how to get some badges, we could be more competitive to get them. Since we did not know which badges could be won, it was difficult to focus on something specific in order to earn them. It would be interesting to get at least some badges for the students in the next course. Of course, they could have some ‘extras’ grades. But if the students knew that it was possible to win the ‘Clean Code’ badges, for example, more students would try to make a better code. The proposal is super interesting.”*



*“It would be interesting to report about the badges, at the beginning of the course, so students could know about it in advance. And, what are the criteria for choosing the badges; it is also interesting to know. In this course, we had the badge ‘Clean Code’, for example, what would be ‘Clean Code’? What specific characteristics the code should have, so that it is chosen? I believe that extra information would be interesting to add awareness for students who could not win the badges.”*

*“It would be interesting to disclosure the activities and rules to earn badges. Thus, we could work focusing on them since the beginning of the course.”*

*“This method could be more interesting if the means of evaluation were clearer and more objective. As ‘Online Participation’, which online actions are rewarded? Log in every day at the platform (online course), or see all classes or post questions? In which case would any of these actions be more accounted than another? Is the ordering for better online participation updated every week? What do you get with your name in badge list or Hall of Fame? Are there awards in grades? Is it visible among colleagues? And, is this visibility positive?”*

### 3.3.2 Study phase II – Interviews results

Based on the results of the survey, we planned and conducted interviews to obtain a better understanding on the student perception on the gamification elements used. Six students accepted the invitation for interviews. The interviews followed the script described in Section 3.2.4. In order to identify the interviewee responses, we adopt the identifiers “Participant A”, “Participant B”, “Participant C”, “Participant D”, “Participant E”, and “Participant F” to refer to the participants, while preserving their anonymity.

Regarding the leaderboards, we first investigated how students used the information about their progress that we provided. As described in 3.1, the leaderboards were introduced in the course with two resources: (i) a chart with partial grades updated regularly, where students could compare their progress against other students; and (ii) a hall of fame. First, we investigated if the students kept track of their progress (IQ1) and if they used the partial grades to compare their performance against their colleagues (IQ2). Except for the Participant A, all participants stated they tracked their progress in the course. Moreover, all participants, except for the participant C, stated they used the grade chart to compare their performance against their colleagues. Participant B, Participant D, Participant E, and Participant F stated that it was useful to



understand the overall performance of the classmates in order to know what their real performance was. Participant A stated that this comparison was a consequence of the competitiveness in the classroom. However, Participant C claimed that he always tried to achieve the best grades, and he was not interested in comparing his performance to the others.

Regarding the positive aspects of this comparison (IQ3), students mentioned three main positive feedbacks: (i) students with lower performance than the average performance of the class felt motivated to perform better, (ii) students tried to understand how they could improve their learning strategy, and (iii) they talked to classmates with better performance to exchange knowledge. One possible issue pointed by the participants was the risk of students acting only in response to the general performance of the class, i.e., if everyone is getting poor grades, there is no reason to try to perform better. Participant D also warned about the risk of creating ego conflicts in the class. With respect to motivation of improving because of this comparison (IQ4), only Participant A and Participant D had negative responses. Participant A would only feel inclined to try to perform better in case the class had a better performance as a whole.

Question IQ5 inquired the participants about their opinion on the “Hall of Fame” resource. Participant A did not like the strategy to implement the Hall of Fame, and claimed that it did not capture the essence of software engineering, and it would be better to acknowledge other aspects such as the best product developed instead of grades. The other participants had positive perceptions about the recognition provided by the Hall of Fame.

From the six participants in this study phase, only one (Participant C) received badges (two) during the course. Participant C found the experience positive and rewarding. Even without receiving badges, Participants B, D, and F found this element rewarding and beneficial as a form of recognition for specific actions. Except for Participant A, all others responded question IQ7 positively.

When we asked participants about the strategy of not revealing the criteria for receiving each badge, the opinions were mixed. Participants B, C, and E defended the idea of specifying the criteria for each badge as soon as possible, because students would establish additional goals besides the grades and, from an educational perspective, it would become an opportunity to pay attention to aspects that they would not normally observe. For Participants C, D, F, the criteria for obtaining badges could make students focus too much attention on the game aspect and, somehow, they could have a counterproductive effect on learning. Except for Participant A, all students were positive when asked if they would like to see more badges in the course.



## 3.4 Discussion

This section discusses the results presented in Section 3.3 and our general findings regarding the research questions defined in Section 3.2.1.

### 3.4.1 RQ1 – Badges in a software engineering course

Our results showed a general positive perception of the students towards the use of badges during the SE Course. Considering that we did not explore this resource to the full potential, by having only eight badges, the students showed interest on them.

Student perception on the role of badges was twofold: (i) they served as a social reward, a public recognition of the student skill or effort; and (ii) they served as a secondary goal, besides the grades and approval, to strive for when performing the course activities. These two elements can be further explored to motivate students not only in performing better in the course, but also as a motivation to further explore software engineering good practices. For instance, it can make students aware of good practices related to the use of tools, of good practices for coding, and so on. In this initial study, we opted for keeping the requirements for earning badges in secret to avoid the students expectation of grades, but we are aware of the motivation it can generate.

One of the students participating in the interviews was particularly unsatisfied about the gamification elements in the course, because he wanted them to explore the practical nature of software engineering and professional practices. This is an important feedback, considering that another approach would be the gamification of specific software engineering activities, as discussed in Chapter 2.

Another issue we observed about the gamification strategy is that most students think of grades as the only reward they can achieve in a course. The feedback received in the survey reflects this rationale: students often asked how the badges would translate into higher grades. It was a surprise to see that they also perceived some value in the social recognition aspect of badges.

### 3.4.2 RQ2 – Leaderboards in a software engineering course

The use of leaderboards in the course was received with mixed opinions. The results of Phase I of the study had more negative responses than positive ones. In the survey, we focused specifically in the Hall of Fame resource. The results obtained from Phase II gave us additional perspectives on this issue.



First, we observed that students used the partial results to regularly compare their performance against each other, and felt motivated to perform better when they had lower grades than the others. These periodic updates were seen as a baseline to understanding the overall performance of the class and to assess their own performance. The interviews showed that the Hall of Fame was seen as a social recognition of the efforts of the students, and it was also seen in a good light from this perspective. We believe that the negative aspect of this strategy may be related to the exclusive focus on grades. While grades are a direct measure of the student performance in the course, it could be complemented by other measures. For instance, we could use additional badges to assess the student performance by their number of achievements.

### 3.5 Threats to validity

In this section, we document potential threats to the study validity and discuss some bias that may have affected the study results. We also explain our actions to mitigate them.

**Results:** The results presented in the study are first and foremost observations, suggestions and lessons learned for further research. We have obviously presented our own interpretation on the analysis of the surveys and interviews. However, there may be several other important issues in the data collected, not yet discovered or reported by us.

**Interviews:** In order to avoid the risk of bias and misinterpretations of the six interviews in our study (and also to avoid depending on good memory of interviewers), we decided to carefully record all interviews and shared with another researcher of this study. After that, one of them was responsible for transcription of all interviews. Therefore, audio and text were available for analyses. Moreover, some meetings were necessary in which the researchers discussed about each answer and extracted all positive and negative impressions about each question. Thereby, we could increase the chance of obtaining an unbiased interview analysis.

**Number of Participants:** The data collected only captures the subjective opinion of each student. A larger number of participants should be interviewed to capture the general view of a broader audience. However, it was our first experience with gamification on software engineering education, and we had a good number of volunteers to participate in our study, without any concrete benefits (i.e., grades). About 50% of all students of the course participated in the survey, but less than 20% took part in the interviews. However, we do not attempt to generalize to a larger population, but



merely discuss some interesting issues discovered during this study (survey and interview). We then presented some discussions, suggestions, lessons learned, and insights for future research. Additionally, this study is an experience report. Therefore, we are concerned in reporting our observations in this scenario, rather than validating any hypothesis.

## 3.6 Final remarks

In this chapter, we described an experience of introducing gamification elements (namely, badges and leaderboards) in a software engineering course. We are aware that the gamification technique has more to offer, but in this first experience, we relied on the most basic and popular elements. Our study was focused on the student perception and the motivational aspect of the approach, rather than on their impact in grades.

Our results showed a positive perception of the use of badges in the course. Students showed interest in badges and saw them as both (i) a social reward and (ii) secondary goals to strive for in the course, besides grades and approval. Regarding the use of leaderboards, our quantitative results showed a negative perception of the resource. However, in the interviews, most of the students mentioned that they like to compare their performance against each other, and, when their performance is lower than the rest of the class, they feel motivated to try to perform better or to rethink their learning strategy. In addition, students liked the possibility of being recognized for their efforts.

We believe that gamification has a motivational role in software engineering education that requires further exploration and evaluation. Although our results cannot be generalized, we provided some evidences on the relevance of this technique in an educational environment. A major drawback is that gamification requires significant effort from instructors to setup and to maintain their elements during a course. Significant lessons learned from this study, that support the specific goal SG1 (*Investigate how Gamification can be used to support software engineering education*) of this thesis, are:

- Social recognition is relevant for students motivation. However, grades are still the most common reward they expect.
- Badges were perceived as secondary goals for student to strive for in addition to approval in the course. Therefore, it is relevant to evaluate the role of badges



as motivators for the adoption of specific software engineering skills, such as following a process.

- One student explicitly claimed that the gamification should promote the practical aspect of software engineering education. However, as a first contact with gamification, we opted for a more simplistic approach, that was positively received.
- The Hall of Fame (leaderboard) was received with mixed opinions. While the students said they would not compare their grades with other students for the purpose of competition, they compare grades with the purpose of assessing their performance in the course in comparison with their colleagues.
- Students would like the existence of more badges and that everyone could try to earn them, instead of having badges awarded for a single student. Again, it is an indication that students are more interested in personal progress than competing with colleagues.

The next chapter describes an investigation on the use of PBL in software engineering education. To achieve that, an Action Research study was performed to systematically understand the issues of using PBL in an introductory software engineering course, and identify possible solutions to address these issues. The next chapter briefly describes the introduction of game elements in this PBL course, as an instrument to motivate students in performing specific software engineering practices.



## Chapter 4

# Empirical Study on the Use of PBL in Software Engineering Education

This chapter describes the experience of adopting Project-Based Learning (PBL) in an introductory software engineering course at Federal University of Lavras (UFLA). An Action Research study was carried with the purpose of gradually introducing PBL in a software engineering course. The action research cycles spanned over four academic periods of the software engineering course offered in the curriculum of the undergraduate program in Information Systems at UFLA. Thirty-two students responded a questionnaire to collect data about the students perceptions on the use of PBL in software engineering education. We compared their responses to the perceptions of 17 students that conducted a similar project in a practical assignment in a similar software engineering course using traditional teaching methods.

This study is directly related to the specific goal SG2 (*Investigate how PBL can be used to support software engineering education*) of this thesis. The remainder of this chapter is organized as follows. Section 4.1 describes the goals, method, and the study design applied in the execution of this study. Section 4.2 describes the course subject to investigation. Section 4.3 describes the Action Research study and its results. Section 4.4 presents the results of a survey with students. Section 4.5 presents the discussion of the results. Section 4.6 discusses the main threats to the validity of this study. Finally, Section 4.7 concludes this chapter.

### 4.1 Study settings

This section explains how we planned and executed this study. Section 4.1.1 presents the goal and research questions of this study. Section 4.1.2 discusses the research



strategy to answer the research questions. Section 4.1.3 describes the process for the execution of this study.

### 4.1.1 Study goals and research questions

The goal of this study is to understand the challenges and lessons learned from the use of Project-Based Learning in software engineering education. To achieve this goal, we propose the following Research Questions (RQ):

**RQ1.** What are the challenges of using PBL in an introductory software engineering course?

**RQ2.** What is the perception of students on the use of PBL in an introductory software engineering course?

### 4.1.2 Research method

To answer the research questions, we conducted an Action-Research study in an introductory software engineering course with the purpose of incrementally refining a PBL approach to introduce a practice-oriented learning method.

Action Research is a research approach that advocates the intervention in a problem, the proposal of solutions and their application, for the purpose of solving the problem and creating theory regarding the action [Coughlan and Coghlan, 2002]. In Action Research, the researchers attempt to solve a real-world problem while simultaneously studying the experience of solving the problem [Davison et al., 2004]. While most empirical research methods have researchers attempting to observe the world as it currently exists, in Action Research, the researchers aim to intervene in the studied situations for the explicit purpose of improving the situation [Easterbrook et al., 2008].

According to Easterbrook et al. [2008], Action Research has been pioneered in fields such as education, where major changes in educational strategies cannot be studied without implementing them, and where implementation implies a long-term commitment, because the effects may take years to emerge. Similarly, dos Santos and Travassos [2009] suggest that this method seems to be a useful research methodology when considering the social challenges involved in software engineering research, and the long history of success in similar domains, in terms of research practice and challenges, such as education and nursing.



### 4.1.3 Study design

In Action Research, activities are organized in a structured cyclic process, called “Action Research Cycle”. Figure 4.1 presents the Action Research Cycle adopted in this study. This process usually includes the following activities: “Diagnosis and Planning”, “Intervention”, “Evaluation”, and “Reflection and Learning” [Davison et al., 2004].



**Figure 4.1.** Action Research cycle, adapted from Davison et al. [2004].

The study was carried in four iterations of an introductory software engineering course, from 2016 to 2017. Each course is considered a cycle of the Action Research cycle. In each cycle, we executed the following activities in each phase.

**Diagnosis and Planning Phase:** Prioritization of problems to address in the current cycle and definition of actions to be taken. This phase was executed before the beginning of each course, defining a Course Plan describing how the actions would be translated to teaching strategy.

**Intervention Phase:** Execution of the Course Plan and data collection. During this phase, we assigned, monitored, and assessed Practical Projects.

**Evaluation Phase:** Evaluation of the course outcomes, from the perspective of students and instructors. In this phase, we analyzed the data collected during the course.

**Reflection and Learning Phase:** Identification and documentation of lessons learned. During this phase, we reflect on the outcomes of the cycle, identifying positive outcomes from the actions taken in the current cycle, and possible issues to be addressed in following iterations.



## 4.2 Course setup

The software engineering course (“SE course”, henceforth) is a 60 hours introductory course offered every semester at Federal University of Lavras, included in the curriculum of the Information System Bachelor undergraduate program. The SE course aims to introduce students to the concepts and methods required for the development of large software intensive systems. The prerequisite for taking this course is the approval in the Object-oriented Programming course.

This software engineering course is mainly based on two textbooks: Software Engineering by Sommerville [2010] and The UML User Guide [Booch et al., 2005]. The course syllabus includes: software development process, agile methods, software requirements analysis and specification, software design, system implementation and testing, configuration management, and software quality.

In previous iterations (before 2016), the SE course was heavily theoretical. Its main activities were mostly based on traditional lectures, exams and practical exercises where students had to design a fictitious software by using UML diagrams. From 2016 to 2018, we adapted the course in order to introduce a more practical approach. We shifted from the teacher-centered method to PBL and introduced a practical assignment in the format of a project, equivalent to 60% of the course grades. Two exams and punctual exercises were responsible for the remaining grades.

### 4.2.1 PBL in the software engineering course

PBL was introduced gradually in the course, as it was the first experience of the course lecturer with this educational method. The following characteristics (based on PBL) were introduced over these four course installments under investigation.

- **Project-based:** The software development project is a central part of the course. All classroom activities and lectures are driven by the progression of the project.
- **Driving questions:** All activities of the project are driven by meaningful questions that direct students into investigating and applying software engineering theory for the project.
- **Realistic:** The project and its process are based on real world problems and methods. The development of the project is organized in iterations, inspired by the Scrum sprints and ceremonies [Schwaber and Sutherland, 2016], and the students are encouraged to use up-to-date tools to support the execution of their projects.



- **Tangible Product:** All projects should result in a minimum working software adequate to the project needs. This product is developed iteratively, creating partial artifacts along the process (documentation and prototypes).
- **Balance between guidance and freedom of choice:** The lecturer act as knowledge facilitators and mentors in the course. The role of the lecturer is to provide meaningful tasks that provide directions for the project progress, and to support students in performing these tasks. However, these tasks should give students room to make decisions on how to execute them. Therefore, the lecturer provides a general roadmap of meaningful activities, and the students decide the order and the means to accomplish these activities. This process results in students having to investigate tools, methods and techniques in the theory or real-world cases.
- **Evidence based:** The students have to provide evidences of performing tasks (mostly related to software development life cycle). Therefore, in order to monitor progress, the students have to not only deliver artifacts but also describe how they performed each task with evidences (e.g., logs of tools, minutes of meetings, specific sections of a document).
- **Teamwork:** The students work in teams, simulating small software developing companies.

The general organization of the course is described as follows. In the first weeks of the course, an anchoring problem of the project is discussed with the students, and the assignment project is presented to the class. Students organize themselves in teams. The project is organized in three iterations, with the first devoted to understanding the project needs, and the two others focusing on the design and development of the project. During each iteration, the lecturer and teaching assistant dedicate some classroom time to support students in the execution of the project, by providing directions for the progress of the project, and performing hands-on classroom activities to mentor students in the practice of specific software engineering activities (e.g., designing use case scenarios, designing architectural design documentation). By the end of each iteration, students deliver artifacts (documentation and increments of the product) and present their results to the lecturer in the format of a seminar (similar to a Sprint Review ceremony from Scrum [Schwaber and Sutherland, 2016]). After presentations, students discuss lessons learned, challenges, and the relations of theory and practice (similar to a Sprint Retrospective ceremony from Scrum [Schwaber and Sutherland, 2016]).



## 4.2.2 Learning goals

The assignment is structured in accordance to the learning goals described in Table 4.1. The “Question” is the driving question of the assignment, i.e., the main inquiry that drives all activities related to the project. The “topics of interest” reflects what topics from the course syllabus should be addressed in the project execution, and the respective skill areas from SWECOM [Ardis et al., 2014]. The “specific skills” are the specific competencies related to the topics of interest that should be developed by students during the project execution. The specific skills were based on SWECOM technical skills [Ardis et al., 2014]. Finally, general skills are relevant crosscutting personal skills, described in SE 2014 [IEEE/ACM, 2015], to any software engineering professional in formation.

**Table 4.1.** Learning goals of the assignment

<b>Driving Question</b>	"How can we systematically design and develop a software product to meet customer needs?"
<b>Topics of Interest</b>	<p><b>Course Syllabus:</b> {software requirements analysis and specification, software design, system implementation and testing, configuration management}</p> <p><b>SWECOM Skill Areas:</b> {Software Requirements; Software Design; Software Construction; Software Testing. }</p>
<b>Specific SE Skills</b>	<p>Elicit customer needs and describe system requirements</p> <p>Design software in accordance to functional and non-functional requirements</p> <p>Develop software in accordance to design decisions</p> <p>Test software using appropriate methods</p> <p>Use appropriate version control tools to manage the evolution of the software.</p>
<b>General SE Skills</b>	<p><b>Professional Knowledge:</b> Show mastery of software engineering knowledge and skills and of the professional standards necessary to begin practice as a software engineer.</p> <p><b>Teamwork:</b> Work both individually and as part of a team to develop and deliver quality software artifacts.</p> <p><b>Design Solutions in Context:</b> Design appropriate solutions in one or more application domains using software engineering approaches that integrate ethical, social, legal, and economic concerns.</p> <p><b>Perform Trade-Offs:</b> Reconcile conflicting project objectives, finding acceptable compromises within the limitations of cost, time, knowledge, existing systems, and organizations.</p> <p><b>End-User Awareness:</b> Demonstrate an understanding and appreciation of the importance of negotiation, effective work habits, leadership, and good communication with stakeholders in a typical software development environment.</p> <p><b>Continuing Professional Development:</b> Investigate and use appropriate up-to-date tools and technologies suitable for the execution of the project.</p>



## 4.3 Observation from the Action Research cycles

This section summarizes the main results observed from the action research cycles. We defined a set of aspects that present the main characteristics that evolved over the course installments as consequences of issues identified in the “Evaluation” and “Reflection and Learning” phase of the cycles, and subsequent actions proposed and implemented in the “Diagnosis and Planning” and “Intervention” phases. Table 4.2 presents an overview of the four installments of the SE course (the action research cycles), describing the number of students enrolled in the course, the distribution of teams, the type of project used in the assignment, and the criteria for evaluating students. The following subsections describe how the course evolved and the lessons we learned in the context of each aspect.

**Table 4.2.** Overview of the installments of the SE course

Cycle	# Stud.	Teams	Type of Project	Evaluation Criteria
2016.1	13	1 team (13 stud.)	Real project External client	Subjective, based on artifacts produced, presentation
2016.2	18	5 teams (3-5 stud.)	Chosen by students Lecturer validates projects	Predefined list of expected artifacts with objective evaluation criteria, presentation
2017.1	7	1 team (7 stud.)	Fictitious project Lecturer as client	Individual performance, evidences of achieving expected outcomes, artifacts, presentation.
2017.2	11	2 teams (5-6 stud.)	Real Project Lecturer as interface of the client	Individual performance, evidences of achieving expected outcomes (gamification), artifacts, presentation.

### 4.3.1 Type of project and realism

In the **2016.1 cycle**, there was a single project for the entire class. To achieve realism, we planned the use of a real project with a real customer. The driving problem was the management of room allocation and their keys at UFLA. The customer were employees at DADP (Pedagogical Support and Development Office), the sector responsible for the reservation of physical spaces and for providing resources (e.g., datashow) and keys to access the requested rooms.

The use of a real project, with a real customer, motivated students. However, the customer was not always available. In the middle of the project, the responsibility over keys and resources for classroom was moved from the original customer to another department at the institution. Therefore, we learned that the use of real clients can be risky, because it introduces variables that are out of the lecturer’s control, such as availability of the stakeholders, and abrupt changes in business rule.



In the **2016.2 cycle**, students were free to choose their own projects, conditioned to the lecturer's approval. This decision led to new problems. The selection of the projects consumed substantial effort, as each team had to reach consensus on what they wanted to do, and the professor had to approve that scope, considering a minimum complexity expected and viability. For instance, Team B initially wanted to develop a simple website. The instructors had to intervene and assist them in choosing relevant features to transform the project in a Web application with more opportunity to apply software engineering concepts. Additionally, students change their scope during the project execution, as they try to reduce its complexity in order to make it viable. One of the teams (Team C) decided to change their project in the second iteration, and they had substantial rework with the specification. Moreover, we observed the risk of unbalanced complexity among projects of different teams, that may lead to conflicts.

Furthermore, by having different projects for each team, it was harder to promote discussion among students. Students had difficulty in sharing lessons with other teams and discussing about the techniques they had to apply in their projects abstracting different project scopes. Mentoring teams in classroom activities was difficult to manage, because instructors had to consider different contexts and it was difficult to provide meaningful examples that would cover all students' projects. Scaling this approach would also require more teaching assistants to support a higher number of different projects. Finally, the students seemed less interested in watching the presentation of their peers.

In the **2017.1 cycle**, we tried a different approach: the students were given a single problem (proposed by the lecturer), and the lecturer acted as requirement provider for the project. The change in the type of project solved a number of issues of the previous courses: (i) Less time consumed negotiating the scope and complexity of projects; (ii) increased availability of the customer, therefore the students had more contact with a stakeholder; (iii) higher immersion of students in a simulated work environment; (iv) more control over the project scope, for instance, we could simulate changes in the scope after each sprint review, and ensure the alignment to learning goals; (v) removal of the risk of high impact changes in business rules or in stakeholders composition; and (vi) it was easier to align lecture examples and classroom activities with the project, as there was only one project for all students.

Despite the fact that the project of 2017.1 cycle was completely fictitious (development of a Web application for a hostel), we did not observe decreased interest of the students. Considering the previous cycles, we believe that the realism of the project is not only dependent of the nature of the project (real versus fictitious projects), but also from the realistic simulation of professional environments.



Finally, in the **2017.2 cycle**, we merged the ideas of the 2016.1 and 2017.1 cycles, and proposed a problem from the real world, with the lecturer acting as an interface between students and the real client. The problem was related to the development of a software application to support production chain and logistics in a small factory of cleaning products. The lecturer investigated the needs of the problem domain, and envisioned the specification of a software product, acting as the requirement provider for the students. As a consequence, the lecturer had a greater effort to understand the problem in anticipation. However, it led to a more convincing narrative for the project. It motivated students into thinking in practical solutions for the problem. The students were more inclined to try to understand the context of the problem than in creating their own interpretations of the client's needs.

All teams worked on their own projects, sharing the same theme/problem. It contributed to the activities conducted in class, where the students could share experiences in a common context. Although there was the risk of students copying the work of others, it did not happen. The teams developed their projects in different technologies, and each team produced different types of documentation. We believe the focus on the activities, rather than the deliverables, contributed to this outcome.

From our observation, we believe that realism can be achieved by designing meaningful narratives and activities that simulates professional software engineering environments. It is more important to setup an appropriate narrative that is believable for students than necessarily relying on real projects, with real customers. It is important to define meaningful activities during the projects and to make sure that students understand and reflect on the importance of each activity, rather than introducing a variety of topics, methods and tools for students to use.

### 4.3.2 Guidance, freedom of choice, and evaluation

In the **2016.1 cycle**, at the beginning of each iteration, the lecturer and external client negotiated with students the goals of the iteration in terms of artifacts and increments to be delivered by the end of the iteration. Every week, the lecturer devoted classroom time to discuss the progress of the project, and to mentor students in hands-on activities related to the project. However, following the open-ended nature of the canonical PBL approach, the project was loosely structured. There were no specific tasks that students had to complete, and students were completely free (and encouraged) to reflect on how to achieve the goals of the iteration. Therefore, the evaluation of students was subjective, focusing on the adequacy and correctness of the artifacts they developed, and the individual contribution observed in classroom.



The lack of clear evaluation criteria was confusing to students. The progress of the project was heavily dependent on the instructor's guidance. The students were not proactive in discovering the next relevant steps for continuing the project, and they always delivered their work products in the last day of the iterations, with little room for improvements. Despite the fact that students were asked to use version control tools, for instance, the course failed to motivate them in the use of such tools to improve collaboration.

In the **2016.2 cycle**, we tried to address these problems by introducing a list of work products the students were expected to produce, and a checklist of attributes we would use to evaluate those work products (as mentioned in Table 4.2). This decision was made to provide clear evaluation criteria both for instructors and for students.

By providing students with evaluation criteria based on the artifacts they have to deliver, students focused more on the documentation rather than on the execution of activities and their purpose. We observed that some teams created documentations that did not reflect the activities they performed in their projects. In some cases, instead of documenting their decisions and what they did in the project, students opted to use templates found in the internet that were much more complex than what they were expected to deliver. Therefore, the evaluation criteria may have led students to believe they would be evaluated for the quality of the documents they produced rather than the activities they performed to produce those artifacts. Again, students only delivered their work products in the exact deadlines. Therefore, it was difficult to provide guidance and feedback to avoid this problem in advance.

In the **2017.1 cycle**, instead of providing a full list of expected work products, in the beginning of each iteration, the lecturer negotiated the goals of the iteration, and the outcomes students were expected to achieve. These expected outcomes were related to the solution of smaller problems that would require students to apply software engineering theory. These outcomes were described as a list of activities that had specific results, but needed students to investigate how to achieve them (e.g., “describe the customer needs”, “document system requirements”, “create a baseline of the project”). Students were free to choose how to document the outcomes of each activity. The evaluation of students was based on the achievement of the expected outcomes and iteration goals, the individual contribution, and the correctness of the work products. In this particular cycle, the observation of individual contribution was facilitated by the smaller number of students.

Students had a more active role in negotiating their learning goals. Consequently, students were more interested in the activities. For instance, the use of Git increased considerably, the students explored the use of branches, 5 (out of 7) students acted as



active collaborators in the GitHub project, and they reached the mark of 60 commits. After the second iteration, students proactively felt the necessity of refactoring the code to increase maintainability. They changed from a perspective of documenting first, to a perspective of discussing possibilities, experimenting and then documenting what was relevant.

In the **2017.2 cycle**, we improved the format introducing gamification. The expected outcomes were described as “quests”. Each iteration had a set of optional and mandatory quests. The mandatory quests were tasks directly related to the goals of the iteration. Optional quests were related to advance topics or further investigation and use of software engineering tools, methods and techniques. These quests were organized in weekly deadlines. Teams were awarded with badges and points (for classification in a leaderboard) for each task they provided adequate evidences of execution in time. Additionally, the lecturer provided feedback for improvement for the teams that submitted evidences in time. The grading was still based on the final artifacts of each iteration and individual performance. However, the quests provided students with a clear roadmap of tasks they had to follow for the progression of the project and criteria for the evaluation of their artifacts. The teams were competing among themselves for the “best team” and they had a shared goal: they were competing for a collective reward (pizza) if they reached a specified amount of points.

The main contribution was providing students with a structured set of activities, and giving freedom to students, so they could choose how and when to address each one. Similar to what was discussed in the lessons of Chapter 3, badges and achievements were understood as smaller goals for students to follow, while striving for the main goal. The idea of having to provide evidences to earn achievements, reinforced the reflection on what they were doing. The feedback mechanics also contributed to a higher involvement of the students of one team, who contacted instructors more often to question about the correctness of their evidences. This also allowed the instructors to observe a continuous effort from this team. The teams were more immersed in the shared goal to obtain the reward (the pizza day), than in the competition among teams.

Therefore, it is possible to balance control and students’ freedom by providing students with meaningful activities that give them room to critically evaluate which ones to perform, in which order to execute them, and with room to accomplish them in varied means. Moreover, allowing students to negotiate aspects of the project is important to increase their participation in the learning process.



### 4.3.3 Teamwork and scalability

In the **2016.1 cycle**, the 13 students enrolled in the course worked together as a single team for the execution of the project. The size of the team led students to organize themselves in sub teams. Consequently, each sub team focused on specific tasks, which limited their practical experience during the assignment. For instance, the version control tool (Git and GitHub) was underused. Only one student used the tool, as he was responsible for integrating the source code provided by his teammates locally. It created communication problems between the sub teams, which resulted in problems in work products consistency. Additionally, the distribution of effort was unbalanced, as some students worked more than others. Lastly, it was difficult to assess the individual participation of students.

In the **2016.2 cycle**, 18 students were enrolled in the course. We limited the size of team to five members, which resulted in the students organizing themselves in five teams. The size of the teams decreased the problem of specialization, and allowed for better tracking of individual performance. Specially with the support of a teaching assistant (the author). However, the different projects of each team impacted in extra effort to track the progress and support students more closely. The execution of hands-on activities in classroom was also compromised by the variety of projects, as it was difficult for the lecturer to provide practical examples that could resemble all projects.

In the **2017.1 cycle**, only seven students were enrolled in the course. We allowed the students to work as a single team. However, the low number of students allowed easier guidance and tracking of individual progress of students. The lecturer and teaching assistant were allowed to offer more mentoring activities.

In the **2017.2 cycle**, there were 11 students enrolled. The students organized themselves into two teams of 5 and 6 members, respectively. As discussed previously, both teams had to work on their own projects, sharing the same theme/problem. It decreased the difficulty to manage classroom activities, allowed the use of the project as an example for contextualizing theory in lectures, and decreased the effort to support the execution of the teams' projects, as they did not need to shift contexts from team to team.

Therefore, we acknowledge that PBL is not easy to scale, in terms of number of students. Consequently, the greater the quantity of teams to support, the greater the effort for the instructors. However, allowing the students to form large teams leads to the risk of students specializing in fewer activities of the assignment, which in turn hampers their learning. Furthermore, larger teams make it difficult to track individual performance, and to identify team members that are not contributing.



Based in our experience in these 4 cycles regarding composition of teams, we believe that the best approach would be limit team size to 3 to 6 members. Project complexity should be tuned in accordance to the team size, in order to avoid overload. Having a single driving problem for the projects of all teams is also a good practice to decrease the effort to manage PBL activities. Finally, the use of gamification and roadmaps of activities, also decrease dependency on instructor's guidance for the project progression, as they encourage proactivity.

Finally, we observed that teamwork is a skill that is difficult to teach, and it was a recurring complaint from students. Difficulty in scheduling meetings out of classroom time and lack of commitment of team members were observed in all cycles. The individual evaluation and allocation of classroom time for the execution of project activities may alleviate these problems, but are not definite solutions.

## 4.4 Questionnaire analysis

In order to understand the students' perception on the use of PBL to support practical education in the software engineering course, we performed an Opinion Survey study. According to Easterbrook et al. [2008], survey studies are used to identify characteristics of a wide population and are usually associated with the application of questionnaires. Surveys are meant to collect data to describe, compare or explain knowledges, attitudes and behaviors [Easterbrook et al., 2008].

A questionnaire was created to collect the participants responses. Table 4.3 presents the structure of the questionnaire. It was structured in four sections, and ten questions, labeled with unique identifiers (Q1 to Q10). The first section (Participant Information) is composed of questions about the background of the participants. The second (Learning Method) is composed of questions designed to evaluate the perception of students about the use of practical development projects in the course and the use of traditional classes. The third part (Learning Topics) inquires students about their perception on the contribution of the project in learning specific software engineering topics. The fourth ("Positive and Negative aspects") asks the participants to describe the positive and negative aspects of the project as a practical assignment.

For the purpose of comparison, we also applied the questionnaire to students of a software engineering course with similar syllabus and a practical assignment with similar scope (in 2017). However, this course adopted traditional teaching process (Non-PBL).



**Table 4.3.** Questionnaire Structure

Section	ID	Question	Type of answer
Participant Information	Q1	Undergraduate Program	Open answer
	Q2	Academic period	[1 to 12]
	Q3	Team during the project	Open answer
	Q4	Was this your first contact with a Software Engineering course? Which other SE related course you have taken?	[yes/no] Open answer
	Q5	Do you have any experience with SE as a professional or trainee?	[yes/no]
Learning method	Q6	Evaluate the statement: “Practical assignments, focused on the development of a software project, are fundamental for developing skills/learning SE”	Likert Scale: (1) Totally disagree; (2) Partially disagree; (3) Indifferent; (4) Partially agree; (5) Totally agree
	Q7	Evaluate the statement: “Traditional expository lectures, with punctual evaluation methods (exams and specific assignments), are sufficient for learning SE”	Likert Scale: (1) Totally disagree; (2) Partially disagree; (3) Indifferent; (4) Partially agree; (5) Totally agree
Learning Topics	Q8	Rate how much the Software Project Assignment contributed for developing skills/learning the following topics a. Software requirements b. Software design and analysis c. Software construction d. Software configuration management e. Agile methods	Nominal Scale: (1) Nothing; (2) Very little; (3) Reasonably; (4) Substantially; (5) Totally
Positive and Negative aspects of the Software Project as a practical assignment	Q9	What are the positive aspects of the Software Project as a practical assignment?	Open answer
	Q10	What are the negative aspects of the Software Project as a practical assignment?	Open answer

#### 4.4.1 Population sample

Table 4.4 describes the sampling of the target population of this study. We considered two samples for analysis and discussion. One for the SE course using PBL as learning method (labelled “PBL”) and the other for a course using traditional teacher-centered approach (labelled Non-PBL). The PBL population is composed of 36 students formally enrolled in three cycles of the action research study (2016.2, 2017.1 and 2017.2 cycles). We did not collect data from the 2016.1 course. The Non-PBL population is composed of 35 students formally enrolled in a single installment of the course. The courses are offered in two distinct institutions - UFLA (for the PBL course) and UFMG (for the Non-PBL course). The author of this thesis acted as teaching assistant in both institutions during the PBL and Non-PBL courses.

The students were invited to participate in the study in the last day of each course. They were informed that participation was not mandatory, that it would not reflect in grades, and that their anonymity would be preserved.



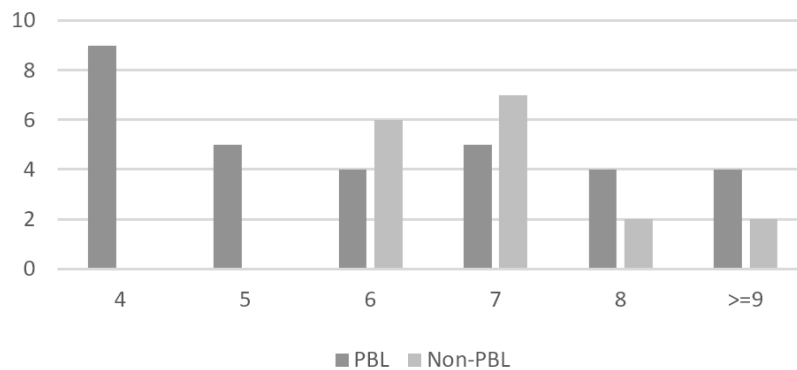
Forty-nine students participated in the study. The PBL sample is formed of 32 students. Not all students enrolled in the courses took part in the survey. As a result, the 32 participants include 15 (out of 18) from the 2016.2 cycle, 6 (out of 7) from the 2017.1 cycle and 11 (out of 11) from the 2017.2 cycle. The Non-PBL sample is composed of 17 (out of 35) students from a single installment of the course. In the Non-PBL course, the students formed 10 teams, and the sample includes participants of 8 different teams.

**Table 4.4.** Population sampling

Class	Population	Sample	Participation Rate
PBL 2016-2	18	15	83.3%
PBL 2017-1	7	6	85.7%
PBL 2017-2	11	11	100%
PBL (Total)	36	32	88.9%
Non-PBL	35	17	49%
Total	71	49	69%

#### 4.4.2 Participants background

Most of the participants were undergraduate students in Information Systems (44 participants – 89.8%). Figure 4.2 shows the academic terms (semester) in which the students were formally enrolled at the time of their participation in the study (Q2). Participants of the Non-PBL group were students at UFMG, in which the SE course is later in their degree program.

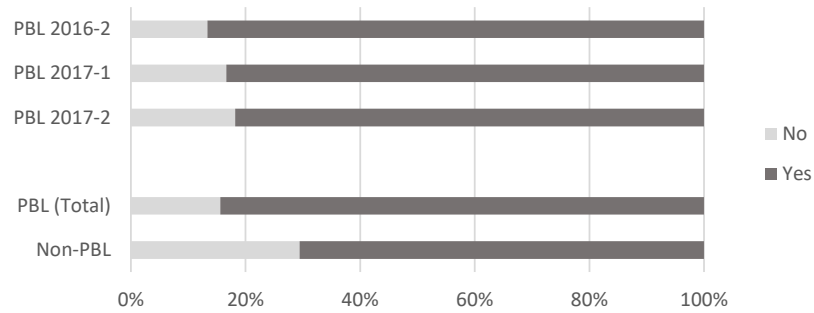


**Figure 4.2.** Academic period of the participants (Q2).

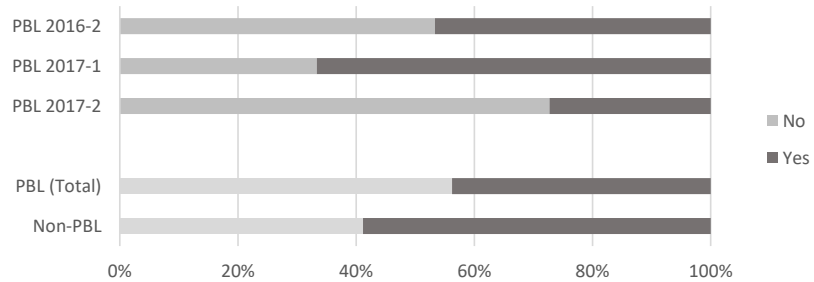
Figure 4.3 shows the distribution of students for whom the course was their first contact with software engineering (Q4). For most of the students, this course was their first contact with software engineering in the academia (39 participants – 79.5%). However, considering that the participants of the Non-PBL sample take the



SE course later (see Figure 4.2), it is reasonable that a larger number of participants of this sample had already taken some software engineering related course. Figure 3 shows the distribution of participants with and without professional experience with software development and software engineering activities (Q5). Similarly, a bigger percentage of participants in the Non-PBL course had previous professional experience with software development and software engineering activities (58.8%, in comparison to 43.7% of the PBL sample). However, for both Q4 and Q5 questions, there was no significant statistical difference between PBL and Non-PBL samples, using the two-proportions hypothesis testing, which may be an indication that both samples have similar backgrounds.



**Figure 4.3.** First Contact with software engineering in academia (Q4).



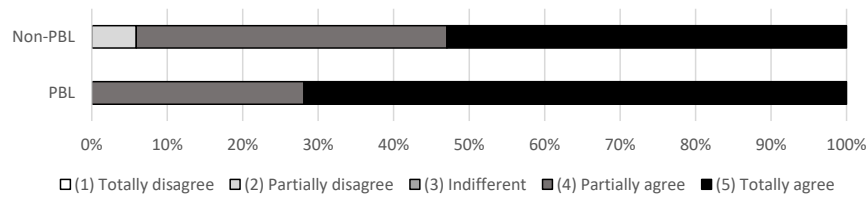
**Figure 4.4.** Professional experience with software development or software engineering (Q5).

### 4.4.3 Evaluation of the learning method

The goal of question Q6 is to collect the participants perception on the relevance of a practical assignment for the development of a software project in the context of developing skills or learning software engineering. Figure 4.5 shows that most of the participants indicated that it is important to some degree. In the PBL sample,



all responses were positive (“4” or “5” in the Likert scale): twenty-three participants (71.9%) responded that “Totally agree” with the statement, and 9 participants (28.1%) responded that “Partially agree” with the statement. In the Non-PBL sample, 9 participants (52.9%) answered that “Totally agree” with the statement, 7 participants (41.2%) answered that “Partially agree” with the statement, and only 1 participant (5.9%) answered that “Partially disagree” with the statement. No significant statistical difference was found for the two samples using unpaired t-test ( $t(47)=1.7218$ ,  $p=0.09$ ).

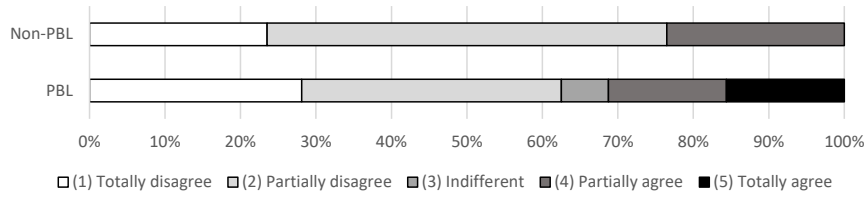


**Figure 4.5.** Evaluation of the use of software projects as practical assignment in software engineering education (Q6).

The purpose of question Q7 was to evaluate if the participants agree that the use of traditional lectures, with punctual evaluation methods (exams and specific assignments), are sufficient for learning software engineering. Figure 4.6 shows that, for the PBL sample, the opinions were divided. However, there was a majority of negative responses (20 – 62.5%). Nine participants (28.1%) responded that “Totally disagree” with the statement, 11 participants (34.4%) responded that “Partially disagree” with the statement, 2 participants (6.2%) responded that are “Indifferent” toward the statement, 5 participants (15.6%) responded that “Partially agree” with the statement, and 5 participants (15.6%) responded that “Totally agree” with the statement. The Non-PBL sample was more emphatic toward a negative perspective of traditional lectures for software engineering education: 4 participants (23.5%) answered that “Totally disagree” with the statement, 9 participants (52.9%) answered that “Partially disagree” with the statement, and 4 participants (23.5%) answered that “Partially agree”. However, no significant statistical difference was found for the two samples using unpaired t-test ( $t(47)=0.8110$ ,  $p=0.42$ ).

Therefore, both samples agree that it is important to introduce development projects in the context of software engineering education, and both samples believe, to some degree, that traditional lectures and exams are not sufficient for learning software engineering. It is interesting to notice that the PBL sample had a greater proportion of positive responses towards the use of traditional approaches in comparison to the Non-PBL sample (Q7). However, all responses of the PBL sample were positive regarding





**Figure 4.6.** Evaluation of the use of traditional lectures and punctual assignments in software engineering education (Q7).

the use of software development projects as assignments.

#### 4.4.4 Evaluation of the project contribution to learning software engineering topics

In question Q8, the participants were asked to what extent the software development project contributed to learn of development skills in five topics covered in the introductory software engineering course. Namely, the topics assessed were: “Software Requirements”, “Software Design”, “Software Construction”, “Configuration Management”, and “Agile Methods”.

Figure 4.7 shows the distribution of responses of the PBL sample. Results show that “Software Requirements” and “Software Design” were the topics most benefited from the use of the project as a learning instrument, followed by Agile Methods, Software Construction and Configuration Management, in this specific order.

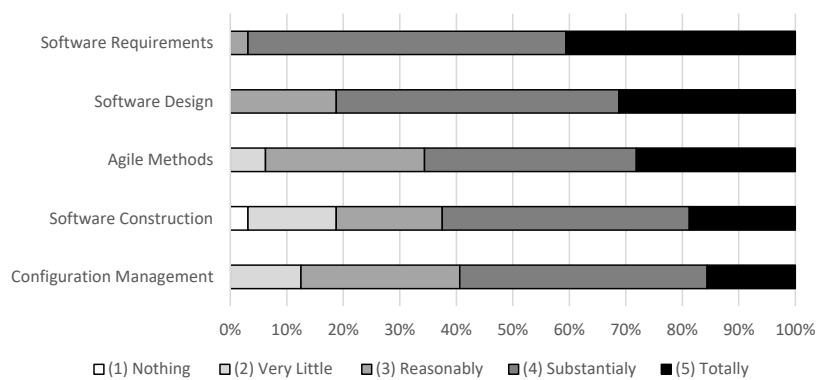
In the case of “Software Requirements”, we believe that the students were highly motivated with the opportunity to simulate the interaction with stakeholders. In these courses, one instructor played the role of a customer, and the students had to arrange meetings with him, interview him for requirement elicitation, and validate requirements and increments in each project iteration.

In the case of “Software Design”, we believe that a software project they had to really implement provided students with a more tangible experience on the impact of design than asking students to design hypothetical software systems. On the other hand, the implementation of the software project was considered a negative aspect for some students (as will be discussed in Section 5.4). Therefore, “Software Construction” was the only topic directly mentioned in the negative aspects. Consequently, it makes sense that this topic had the greater proportion of low score responses.

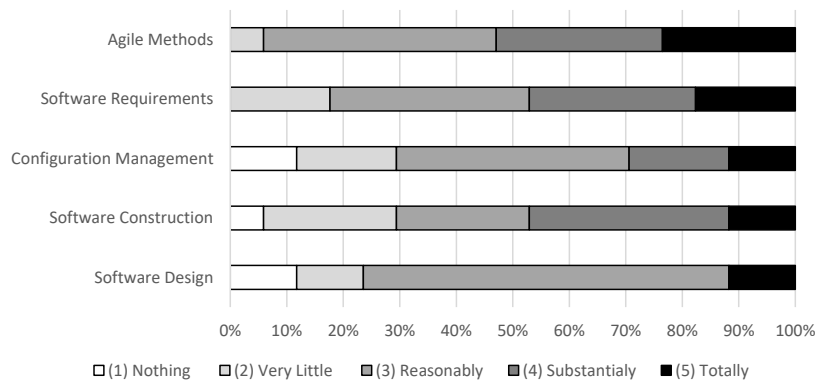
Figure 4.8 shows the distribution of responses of the Non-PBL sample. Only one topic (Agile Methods) had the number of responses “4. Substantially” and “5. Totally” superior to 50%. We believe that a key difference in the results was the role of



the instructors in the practical assignments. In the PBL course, the instructors were constantly guiding students in the appliance of theory in the project and ensuring that the project contributed to their learning goals. In the Non-PBL course, understanding the link between theory and practice was a responsibility of the students. For instance, two of the most recurring negative aspects pointed by the Non-PBL sample was the lack of orientation in activities and the lack of classroom activities to support the project development. One of the students stated that “[the project] led students to perform some activities without real interest or without acknowledging the real application of what they were doing”.



**Figure 4.7.** Contribution of the project in learning specific software engineering topics (Q8) for the PBL sample.

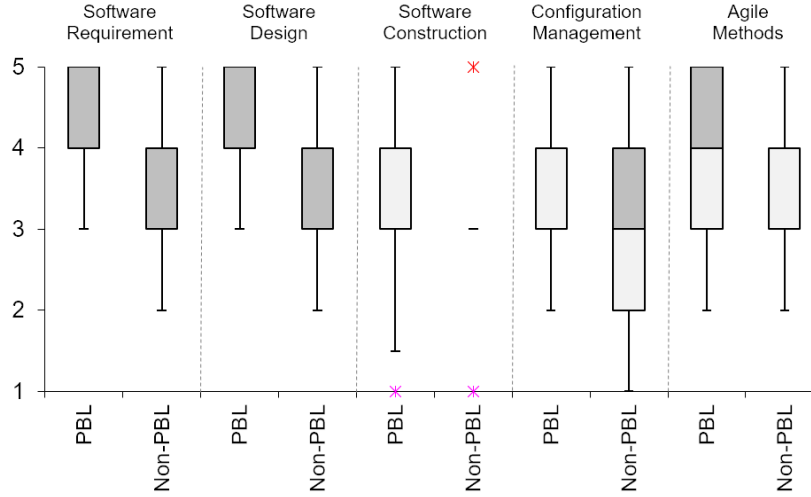


**Figure 4.8.** Contribution of the project in learning specific software engineering topics (Q8) for the Non-PBL sample.

Figure 4.9 shows the analysis of the median values for the responses of both samples. It is interesting to notice that, except for “Agile Methods”, all topics had the median value one point higher in the PBL sample responses, in comparison to the Non-PBL sample responses. Significant statistical difference was observed for the results



of the topics “Software Requirements” ( $t(47) = 4.0724$ ,  $p=0.0002$ ), “Software Design” ( $t(47) = 4.9216$ ,  $p < 0.0001$ ) and Configuration Management ( $t(47) = 2.0714$ ,  $p = 0.04$ ). However, considering the small sample size and the single installment of the Non-PBL course, additional data are required to allow generalization.



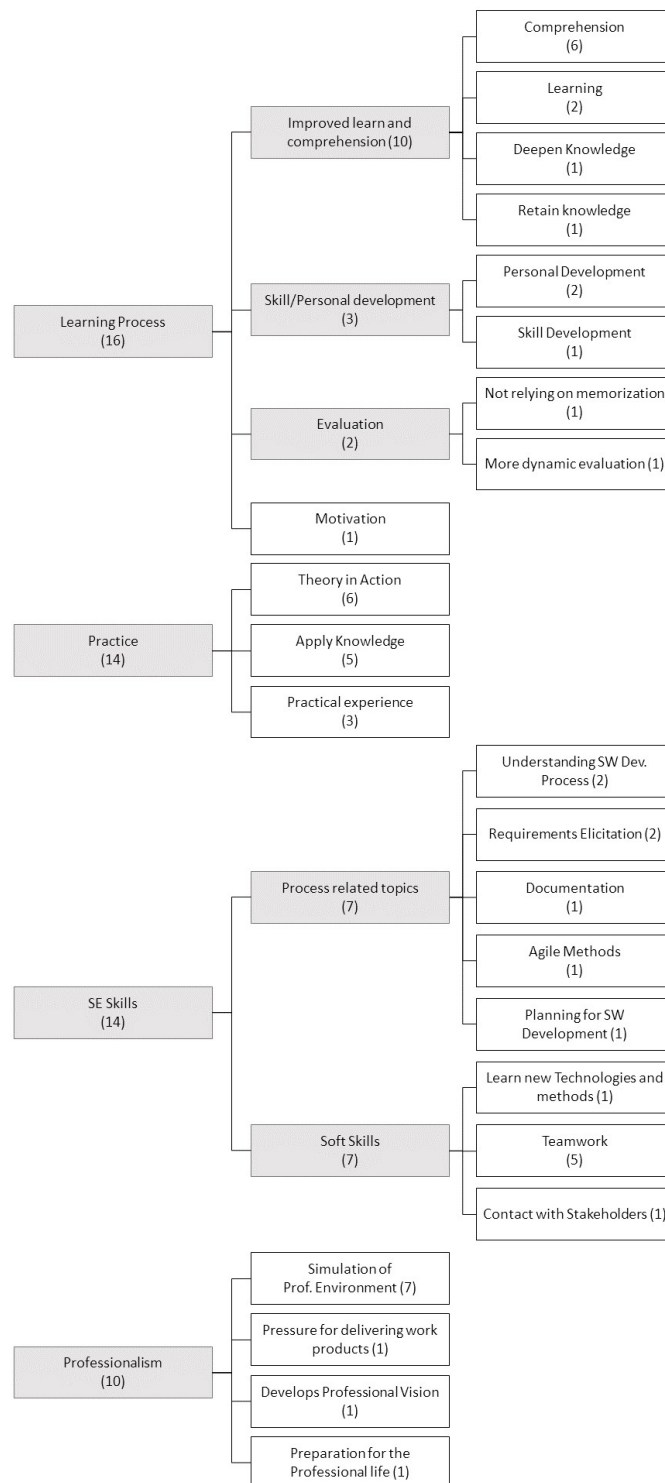
**Figure 4.9.** Comparison of the results for Q8.

#### 4.4.5 Positive and negative aspects of the PBL course

In questions Q9 and Q10, the participants were asked to describe the positive and negative aspects of the software development project assignment they participated. To analyze the answers, we used an approach inspired by the coding phase of Ground Theory [Stol et al., 2016]. Therefore, two researchers analyzed the responses individually and marked relevant segments with “codes” (tagging with keywords). Later, the researchers compared their codes to reach consensus, and tried to group these codes into relevant categories. Consequently, it is possible to count the number of occurrences of codes and the number of items in each category to understand what recurring positive and negative aspects are pointed by the participants, and propose possible lessons learned.

Regarding the **positive aspects**, the data analysis resulted in 24 different codes, which occurred 54 times. The codes were grouped in five main categories: Learning Process (16 occurrences), Professionalism (10 occurrences), Practice (14 occurrences), and software engineering Skills (14 occurrence). Figure 4.10 presents the categories, subcategories and codes. The numbers in parenthesis represent the number of times the code was assigned to the responses.





**Figure 4.10.** Positive aspects stated in the responses of Q9.

The category “Learning Process” groups codes related to the participants statements regarding how the project helped them in acquiring, retaining and deepening



knowledge, and how it facilitated the understanding of the software engineering topics learned in class. For instance, 10 students mentioned positive aspects related to improved learning and comprehension of software engineering topics, 3 students mentioned the project contributed to personal development or skill development, and 2 students mentioned they liked the evaluation process because it was more dynamic and it did not rely on memorization.

The category “Professionalism” groups codes related to the participants perception on how the project simulates a work environment similar to the professional context of software engineering. The simulation of a professional environment was directly mentioned by 7 students. Three students also mentioned that the project allowed them to develop a professional perception, that it prepares for the future professional life, that the pressure for delivering work products was relevant for understanding industry dynamics.

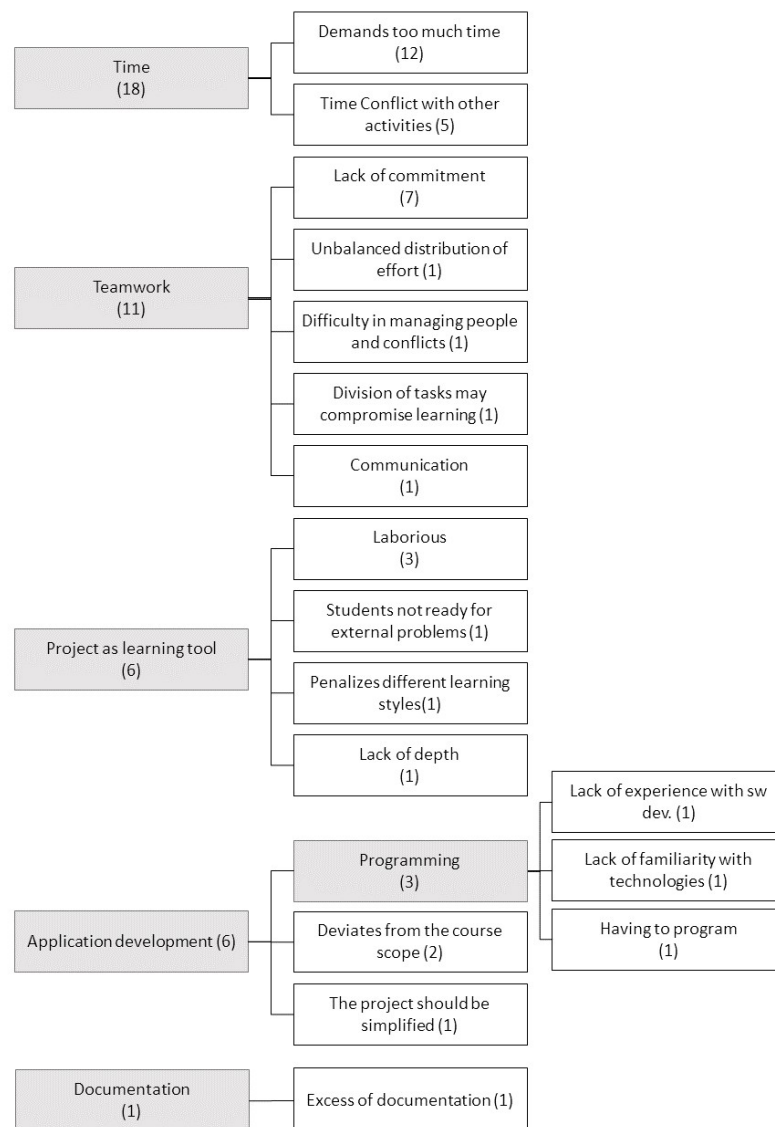
The category “Practice” groups codes related to the positive aspects of the practical nature of the project. The most recurrent codes were related to how the project allowed the participants to see in action the theory they learned in classroom (6 participants), and how the project allowed the participants to practice (apply) the concepts they learned (5 participants). Three participants claimed that the practical experience was positive in general.

The category “SE Skills” groups codes related to software engineering activities the participants explicitly stated as positive outcomes of the project. Seven participants mentioned soft skills such as teamwork (in 5 responses), learning new technologies and methods (in 1 response) and dealing with stakeholders (in 1 response). Seven participants mentioned process related topics, such as understanding the software development process (2 responses), requirements elicitation (2 responses), agile methods (1 response), documentation (1 response), and planning for software development (1 response).

For the **negative aspects**, the data analysis resulted in 17 different codes, which occurred 42 times. The codes were grouped in five main categories: Time (18 occurrences), Teamwork (11 occurrences), Application Development (7 occurrences), Project as a Learning Tool (5 occurrences), and Development Process (1 occurrence). Figure 4.11 presents the categories, subcategories and codes. The numbers in parenthesis represent the number of times the code was assigned to the responses.

The most recurrent issues pointed by the participants were related to Time (18 occurrences). Students complained that this type of activities demands too much time (12 occurrences), and that it was troublesome to divide their time with other activities from the university (6 occurrences).





**Figure 4.11.** Negative aspects stated in the responses of Q10.

The problems in the category of Teamwork (11 occurrences) were mainly related to the lack of commitment of some team members (4 occurrences), and unbalanced effort distribution (4 occurrences), i.e., some students did more activities than others. In fact, these problems were observed in every iteration of the course (as described in Section 4.2.1). Other issues related to this category were the difficulty in managing people and conflicts (1 occurrence), and communication problems (1 occurrence). One participant pointed a issue that is similar to a concern described in Section 4.3.3: the division of the activities among team members may compromise learning, since students tend to develop the tasks they are already familiar with. Therefore, activities



in classroom should promote at least a minimum contact of the students with each topic, and instructors should encourage students to participate in all activities.

Regarding the Application Development (6 occurrences), two participants believed that the development of the application was not related to the scope of the course. One participant stated that the development of the application should be simplified. Other issues were specifically related to programming skills, as three students complained about having to code, the lack of familiarity with the technologies (programming language and framework), and the lack of experience with software development compromised the student ability to contribute.

The category “Project as a Learning Tool” (6 occurrences) grouped three negative aspects: three students claimed the project was too laborious; one student claimed that the students were not prepared to deal with problems that are external to the academia; and one participant believed that the assignment penalizes students that prefer a theoretical approach and are not used to work under pressure. One student mentioned that the approach lacks deepness regarding the execution of each phase of software development. Finally, only one student complained about the excess of documentation.

#### 4.4.6 Comparison of positive and negative aspects between PBL and Non-PBL courses

Table 4.5 lists all positive aspects found captured from the results of Q9, their categories, the number of occurrences (#) and the percentage of participants who mentioned them (%) for each sample (PBL, Non-PBL, and Total). Comparing the positive aspects identified for the PBL and Non-PBL samples, we observe a total of 28 unique codes, with 4 exclusive codes from the Non-PBL sample, 16 exclusive for the PBL sample, and 8 codes in common for both samples. There were 87 occurrences of these codes, with 7 occurrences for the exclusive codes from the Non-PBL sample, 19 for the exclusive codes of the PBL sample, and 61 for the codes in common for both samples.

Table 4.6 presents the distribution of the positive aspects in relation to their categories for the PBL and Non-PBL samples. The data show that the majority positive aspects stated by the Non-PBL sample are grouped in the categories “Specific SE Skills” and “Practice”. The positive aspects pointed by the PBL sample are more evenly distributed, with a higher count of positive aspects related to the learning experience. The category “Professionalism” was also more present in the responses of the PBL samples.



**Table 4.5.** Positive aspects identified in the responses of the PBL and Non-PBL samples for Q9.

Positive Aspects	Category	# PBL	% PBL	# Non- PBL	% Non- PBL	# Total	% Total
Apply knowledge	Practice	5	15,6%	8	47,1%	13	26,5%
See theory in action	Practice	6	18,8%	4	23,5%	10	20,4%
Better comprehension of the learning topics	Learning	6	18,8%	3	17,6%	9	18,4%
Simulation of Professional Environment	Professionalism	7	21,9%	1	5,9%	8	16,3%
Teamwork	SE Skills	5	15,6%	2	11,8%	7	14,3%
Understanding the SW dev. Process	SE Skills	2	6,3%	4	23,5%	6	12,2%
Agile Methods	SE Skills	1	3,1%	3	17,6%	4	6,2%
Practical experience	Practice	3	9,4%	1	5,9%	4	8,2%
Use of collaboration tools	SE Skills			3	17,6%	3	6,1%
Personal development	Learning	2	6,3%			2	4,1%
Requirement Elicitation	SE Skills	2	6,3%			2	4,1%
Learning	Learning	2	6,3%			2	4,1%
Understanding the relevance of process artifacts	SE Skills			2	11,8%	2	4,1%
Not having to rely on memorization	Learning	1	3,1%			1	2,0%
Learn new technologies and methods	SE Skills	1	3,1%			1	2,0%
More dynamic evaluation	Learning	1	3,1%			1	2,0%
Motivation	Learning	1	3,1%			1	2,0%
Retain Knowledge	Learning	1	3,1%			1	2,0%
Documentation	SE Skills	1	3,1%			1	2,0%
Pressure for delivering work products	Professionalism	1	3,1%			1	2,0%
Planning for the SW development	SE Skills	1	3,1%			1	2,0%
Preparation for the future	Professionalism	1	3,1%			1	2,0%
Professional vision	Professionalism	1	3,1%			1	2,0%
Skill development	Learning	1	3,1%			1	2,0%
Deepen knowledge	Learning	1	3,1%			1	2,0%
Contact with stakeholders	SE Skills	1	3,1%			1	2,0%
Product release	SE Skills			1	5,9%	1	2,0%
Play a process role	Professionalism			1	5,9%	1	2,0%

**Table 4.6.** Categorization of the positive aspects identified in the responses of the PBL and Non-PBL samples for Q9

Categories	# PBL	% PBL	# Non-PBL	% Non-PBL	# Total	% Total
SE Skills	14	25,9%	15	45,5%	29	33,3%
Practice	14	25,9%	13	39,4%	27	31,0%
Learning	16	29,6%	3	9,1%	19	21,8%
Professionalism	10	18,5%	2	6,1%	12	13,8%

% PBL =  $[\# \text{ PBL}] / 54$

% Non-PBL =  $[\# \text{ Non-PBL}] / 33$

% Total =  $[\# \text{ Total}] / 87$

Similarly, Table 4.7 lists the negative aspects identified in the answers of Q10. Comparing the negative aspects stated by the PBL and Non-PBL samples, we observe a total of 34 unique codes, with 17 exclusive codes from the Non-PBL sample, 10 exclusive for the PBL sample, and 7 codes in common for both samples. There were 82 instances of these codes, with 30 for the exclusive codes from the Non-PBL sample, 11 for the exclusives codes of the PBL codes, and 41 for the codes in common for both samples.



**Table 4.7.** Negative aspects identified in the responses of the PBL and Non-PBL samples for Q10.

Negative Aspects	Category	# PBL	% PBL	# Non- PBL	% Non- PBL	# Total	% Total
Demands much time	Time	12	37,5%	4	23,5%	16	32,7%
Time conflict with other activities	Time	6	18,8%	1	5,9%	7	14,3%
Lack of Commitment	Teamwork	4	12,5%	1	5,9%	5	10,2%
Unbalanced distribution of effort	Teamwork	4	12,5%	1	5,9%	5	10,2%
Lack of orientation in activities	Project as learning tool			5	29,4%	5	10,2%
Laborious	Project as learning tool	3	9,4%	1	5,9%	4	8,2%
Lack of classroom activities to support the project development	Project as learning tool			4	23,5%	4	8,2%
Extensive documentation artifacts	Documentation			4	23,5%	4	8,2%
Difficulty in managing people and conflicts	Teamwork	1	3,1%	1	5,9%	2	4,1%
Deviates from the course scope	Application Development	2	6,3%			2	4,1%
The project should be simplified	Application Development	1	3,1%	1	5,9%	2	4,1%
The process is promotes lack of inter- est/demotivation	Development Process			2	6,3%	2	4,1%
The process seems useless	Development Process			2	6,3%	2	4,1%
The documentation is tiresome	Documentation			2	6,3%	2	4,1%
Division of tasks may compromise learning	Teamwork	1	3,1%			1	2,0%
Lack of communication	Teamwork	1	3,1%			1	2,0%
Lack of familiarity with technology	Application Development	1	3,1%			1	2,0%
Having to program	Application Development	1	3,1%			1	2,0%
Lack of experience with sw development	Application Development	1	3,1%			1	2,0%
Not ready to face external problems	Project as learning tool	1	3,1%			1	2,0%
Penalize students with different learning styles	Project as learning tool	1	3,1%			1	2,0%
Broad and subjective scope	Project as learning tool			1	5,9%	1	2,0%
The phases should resolve quicker	Project as learning tool			1	5,9%	1	2,0%
Lack of depth	Project as learning tool	1	3,1%			1	2,0%
Lack of feedback	Project as learning tool			1	5,9%	1	2,0%
Excess of Documentation	Documentation	1	3,1%			1	2,0%
Rigid Development Method	Development Process			1	3,1%	1	2,0%
The process does not contribute to learning	Development Process			1	3,1%	1	2,0%
Unrealistic process	Development Process			1	3,1%	1	2,0%
Process requires too much details in some parts	Development Process			1	3,1%	1	2,0%
Outdated materials	Documentation			1	3,1%	1	2,0%
Unpractical documentation	Documentation			1	3,1%	1	2,0%
Excessive formalism in documentation	Documentation			1	3,1%	1	2,0%
Documentation susceptible to changes	Documentation			1	3,1%	1	2,0%

Table 4.8 presents the distribution of the negative aspects in relation to their categories for the PBL and Non-PBL samples. The data show that the majority of the negative aspects stated by the Non-PBL sample are grouped in the categories “Project



as Learning Tool”, “Documentation” and “Development Process”.

**Table 4.8.** Categorization of the negative aspects identified in the responses of the PBL and Non-PBL samples for Q10

Categories	# PBL	% PBL	# Non-PBL	% Non-PBL	# Total	% Total
Time	18	42,9%	5	12,5%	23	28,0%
Project as learning tool	6	14,3%	13	32,5%	19	23,2%
Teamwork	11	26,2%	3	7,5%	14	17,1%
Documentation	1	2,4%	10	25,0%	11	13,4%
Development Process	0	0,0%	8	20,0%	8	9,8%
Application development	6	14,3%	1	2,5%	7	8,5%

% PBL = [# PBL] / 42

% Non-PBL = [# Non-PBL] / 40

% Total = [# Total] / 82

The category “Project as Learning Tool” is directly related to the perceptions of the students regarding the learning process. The main complaints were related to the lack of orientation for the execution of the project and the lack of activities in the classroom to provide direct support to the development of the project. These two points were objectively addressed in the PBL courses, where the development of the project was the core activity in the course. Therefore, not only the lecturers devoted several classrooms activities for mentoring students and providing time for students to work in classroom, but also the course was structured focusing on continuous orientation for the project development.

In the “Documentation” category, the participants stated that the documentation they had to provide was too extensive, not practical, and boring. In the “Development Process” category, the students stated that it was far from what the participant believed was the reality of a professional environment, that the students perceived little value in the development process and its documentation, that the process had little contribution for learning, and that the process promoted demotivation in the students. In the PBL courses, a key point addressed was the focus on expected activities students should perform rather than on documentation, giving the students freedom to choose how to perform them. Therefore, in the PBL approach, the students had to focus on goals, not on following a process or filling document templates.

Considering the positive and negative aspects for each population sample, the PBL sample provided 54 positive codes and 42 negative codes. The Non-PBL sample provided 33 and 40 positive and negative codes, respectively. Therefore, the proportion of positive aspects codes was higher in the PBL sample and the proportion of negative codes was higher in the Non-PBL sample.



## 4.5 Discussion

The experience of using PBL in an introductory software engineering course was positive both in the students and instructors' perceptions. There was an initial concern that the learn-by-doing approach would be more confusing than the learn-then-practice approach in the specific case of an introductory SE course. However, we perceived that students were more engaged in the learning process using the PBL approach, and the approach allowed a better immersion regarding the software development process. In the following subsections, we summarize the main observations made in relation to the research questions defined in Section 4.1.1.

### 4.5.1 RQ1 – The challenges of using PBL in an introductory software engineering course

The main challenges we observed during the Action-Research study were:

**Scaling PBL:** The main challenge of scaling PBL is providing the right amount of guidance for students for a large number of students. In the cases described in Section 4.2.1, the class with the best overall results was Class 2017.1. This class had the smallest number of students enrolled. Therefore, the instructors were able to provide better guidance and feedback for students. Having all teams developing the same project and introducing gamification were useful actions to replicate the success of class 2017-1 in class 2017-2. The introduction of gamification in the later instance of the course allowed the instructors to provide general directions to students, in form of achievements and badges, without recurring to a strict process, and streamlined the evaluation of teams progress. The problem of scaling PBL is discussed in the literature [Gary, 2015; Harms and Hastings, 2016].

**Selection of appropriate projects:** Projects play the central role in PBL approaches. The literature suggests the use of real open-ended projects. However, (i) the lack of control over the project and (ii) the volatility of the commitment of external stakeholders are threats that need to be carefully analyzed. For the first (i), especially in the case of introductory courses, instructors must be aware of the risk that the project may not support expected learning outcomes or provide students with meaningful opportunities to apply specific knowledge related to the course syllabus (this problem was previously discussed by Nguyen et al. [2013]). For the second (ii), external stakeholders may lack the motivation or availability for participating in the project. It is also difficult to demand students to meet with stakeholders in environments external to the university, or outside classroom time. Finally, external stakeholders are susceptible



to changes in their business rules, that may invalidate the whole project, and external clients may abandon the project in the middle of the course. All these situations may lead to student frustration, or risk of not addressing expected learning outcomes. Harms and Hastings [2016] state that “projects need to not be too shallow and yet not be too idealistic either”. The issues of using real customers or having instructors playing the role of customer is discussed by Daun et al. [2016]. Kizaki et al. [2014] warn that the participation of external person as real customers may be problematic, as their availability and expectations are not controlled.

**Tracking progress and learning outcomes of students:** Teamwork is an important software engineering skill, not only in curricular guidelines [IEEE/ACM, 2015] but also in the students’ perception (see Table 4.10). However, it is difficult to track the individual progress of students. The data obtained from students’ response show that there is difference in commitment levels in the teams, and that some students work more than others. The more open-ended the project is, the more difficulty some students have in understanding what they are supposed to do. Classroom activities helped alleviating this problem in the cases described in Section 4.2.1, as the instructors could watch the participation of each student in teams. Also, gamification provided the students with clear goals they could use to measure and plan their progress. Finally, defining effective evaluation processes for assessing the students performance is difficult, because of the nature of PBL projects. As discussed by Fagerholm and Vihavainen [2013], the assessment design plays a key role in what students will focus on. We noticed this problem in class 2016-2. All these issues are discussed in previous works [Gary, 2015; Harms and Hastings, 2016; Fukuyasu et al., 2013].

**Balancing guidance and freedom of choice:** Balancing the intervention of professors for guidance and the students freedom to decide what to do is challenging. For instance, in Class 2016.2, the students were given documentation templates, and their evaluation was based on the delivery of such artifacts. As mentioned, this led students to focus on filling artifacts with little reflection on their importance. On the other hand, relying only on the instructors’ guidance, as the case of Class 2016.1, may confuse students if there is no roadmap of activities. Therefore, it is specially important to design meaningful activities or driving questions that guide the development of the project, without imposing specific approaches that limit students choice. Martin et al. [2014] consider that the ill-structured nature of problems is one of the cornerstones of PBL and one of the main sources of difficulties in using PBL. Therefore, we opted to give students a clear roadmap of “what to do” but leaving them free to choose “how to do it”. In this aspect, gamification allowed the instructors to provide this roadmap, by using achievements and badges, and to link it to the evaluation rubrics of the course.



Additionally, the use of a reference model, such as SWECOM [Ardis et al., 2014], supported instructors in defining meaningful goals for students to pursuit during the execution of the project.

#### **4.5.2 RQ2 – Students’ perception on the use of PBL in an introductory software engineering course**

The main findings from the questionnaire responses were that the students perceived a positive contribution of the practical project in their learning process in relation to specific software engineering topics. The topics that received better ratings were “Software Requirements” and “Software Design”. However, in a scale of 1 (no contribution) to 5 (totally contributed), the median value was above 4. In comparison to the perceptions of students in a course using a software development project, but not the PBL approach, these median values of the PBL samples were 1 point higher for all topics, except for “Agile Methods”.

The most recurrent positive aspect observed by the students were related to the simulation of a professional environment (7 responses), the better comprehension of the topics learned in classroom, and the possibility of seeing the theory in practice. In general, most of the responses were related to the learning process. Therefore, we believe that PBL was received positively by the students.

The most predominant negative aspects were related to the time-consuming nature of PBL, and problems related to working in teams. The first problem is inherent to the PBL approach, while the second is a challenge that is often related to software engineering and an expected skill to be developed [IEEE/ACM, 2015]. In contrast, the participants of the Non-PBL sample made more complaints about the project as a learning tool and the documentation that was required for evaluation. Our interpretation of this data is that the projects are more meaningful for students in PBL courses.

Regarding the use of a practical software development project for learning software engineering, both samples shared similar distribution of responses, stating that they are strongly favorable to its relevance. Similarly, both samples shared similar distribution of response disagreeing that traditional expository lectures and evaluation methods are sufficient for learning software engineering. For both questions, the responses were even more emphatic when we segment the population in students with some professional experience with software engineering or software development, and students without experience. The students with professional experience seem to favor learning methods that promotes practice. One student also mentioned that the PBL



format penalizes students who prefer a theoretical approach and are not used to work under pressure, which was already discussed in previous work [Zhi, 2016].

Therefore, there was a general positive acceptance of PBL from the students. A software development project not only helped in balancing theory and practice but also provided students with the opportunity to understand some aspects that only theory would not address. However, we noticed that using the educational project in the context of a PBL process was more beneficial than using it in a traditional course format. Our interpretation is that the project in Non-PBL course was not sufficiently contextualized and synchronized with lectures, and students have difficulty in linking theory to practice in this scenario. In contrast, PBL provides meaningful context for practice in the learning process, enforcing the students perception of the importance of each decision they make in the project.

## 4.6 Threats to validity

In this section, we document potential threats to the study validity and discuss some bias that may have affected the study results. We also explain our actions to mitigate them.

**Results:** The results presented in the study are first and foremost observations, suggestions and lessons learned for further research. We have obviously presented our own interpretation of the analysis of the questionnaires and classroom experiences. However, there may be several other important issues in the data collected, not yet discovered or reported by us. Nevertheless, our reports may provide significant insights for other researchers and educators when planning or evaluating PBL approaches in similar settings.

**Questionnaires:** In order to avoid the risk of misinterpretations of the questions, the questionnaire was developed in stages. The first version of the questionnaire was reviewed by two researchers who are also software engineering professors. It was then piloted with three students in order to assess if the questions were clear, not ambiguous, and if the available options for answers were coherent. Additionally, the participation in the questionnaire was not compulsory, it preserved the participants anonymity, the participation did not contribute for grades, and the questionnaires were always applied at the end of the course. These decisions were made to avoid the bias of students providing positive answers for the sake of fearing bad consequences or hoping that it would somehow benefit them.

**Number of Participants:** A larger number of participants should be surveyed



to capture the general view of a broader audience. However, the study was limited to the population of students that (i) were enrolled in the course, and (ii) were willing to participate in the questionnaire. For instance, the Non-PBL sample had a lower participation rate. However, by forcing students to participate in the questionnaire, or rewarding the participation with grades, we would introduce more bias. Additionally, this type of study is limited by the availability of professors willing to allow the author to participate in their teaching activities, and who were willing to use the approaches considered for this study.

**Population sampling:** The comparison of the PBL and Non-PBL samples suffers from the bias of being from different institutions. Therefore, there is the bias of the participants having different backgrounds (universities, cities, lecturers, class size) and the comparison not being adequate. However, other options were considered such as having half of the class using PBL and the other using traditional lectures, or alternating the learning methods in different semesters in the same institution. However, the cost-benefit of both approaches was not relevant. In the case of this study, both courses are part of the degree plan for undergraduate programs in Information Systems, both share similar syllabus, both are 60 hours courses, and both are provided by public institutions. The author of this work acted as a teaching assistant in both setups. We acknowledge that further investigation is required, and that our results may not be appropriate for generalization.

## 4.7 Final remarks

This chapter presented an experience report on the use of PBL in an introductory software engineering course. The approach was applied in four academic periods, for a total of 49 students participated in this course. An Action Research study was performed to support the incremental evolution of the course, identifying key problems and gradually proposing changes to the course. The main challenges faced during the Action Research study were related to scaling PBL, selecting appropriate projects, tracking progress and learning outcomes of students, and balancing guidance and freedom of choice. Gamification, the use of training techniques from the industry (e.g., mentoring), and the adoption of reference models for the definition of meaningful goals for students were relevant resources for addressing those issues.

In addition to the observation of the cases, 32 students responded a questionnaire to collect their perceptions about the course. The responses show an overall positive reception of the method. We compared these responses to the responses of 17 students



who participated in an introductory software engineering course with similar syllabus that also used a software development project with similar learning goals. However, this second course adopted a traditional teaching instead of PBL. The overall responses of the PBL sample were more positive than the responses of the Non-PBL sample, both in relation to the contribution of the project to learning specific software engineering topics, and in relation to the proportion of positive and negative aspects stated by the students. However, both samples agree in similar proportion that practical development projects are necessary for learning software engineering, while they disagree in similar proportion that traditional lectures are sufficient for learning software engineering.

In the context of this thesis, the results of this chapter support the specific goal SG2 (*Investigate how PBL can be used to support software engineering education*). The observed challenges of adopting PBL are in accordance with the challenges identified in the literature (Chapter 2). In the 2017.2 Research cycle, we introduced the use of gamification for specific software engineering practices. Therefore, in this thesis we explored both the gamification of the classroom experience, and the gamification of specific SE practices, as discussed in Chapter 2. Hence, this chapter also provided insights for the specific goals SG1 (*Investigate how Gamification can be used to support software engineering education*) and SG3 (*Investigate the benefits and drawbacks of the joint use of gamification and PBL to support software engineering education*).

The next chapter describes GaPSEE: a framework to support the adoption of PBL and gamification in software engineering education. The lessons learned from the experience described in this chapter are inputs for the proposal of this framework.







## Chapter 5

# A Framework for the Gamification of Project Based Software Engineering Education

This chapter describes GaPSEE (Gamification of Projects for Software Engineering Education), a framework for gamification of project-based software engineering education. According to Mora et al. [2015], “a framework is a real or conceptual structure intended to serve as a support or guide for the building of something that expands the structure into something useful”. As a framework, GaPSEE is intended to provide a set of guidelines and a process to support lecturers in planning the execution of practical assignments to introduce practice in software engineering related courses. The framework is based on the lessons obtained from the studies described in the previous chapters.

The remainder of this chapter is organized as follows. Section 5.1 describes the goals and scope of GaPSEE. Section 5.2 describes the target audience of GaPSEE. Section 5.3 describes the organization of GaPSEE in components. Section 5.4 provides an overview of the organization of a practical assignment in accordance to GaPSEE. Section 5.5 presents the guidelines defined by GaPSEE. Section 5.6 describes the process suggested by GaPSEE and its activities. Section 5.7 presents a set of observations and suggestions to be considered when implementing GaPSEE in a software engineering related course.



## 5.1 Goal and scope

The goal of GaPSEE is to support educators in the planning and execution of practical assignments for software engineering related courses, using principles of PBL and Gamification. Therefore, GaPSEE has three core concepts in its foundation: software engineering education, Project-based Learning (PBL), and gamification.

From the perspective of software engineering education, GaPSEE is intended to provide an alternative approach to introduce practice in the educational process of software engineering. Therefore, it provides a process and recommendations to help educators in the setup, execution, evaluation, and refinement of practical assignments, using principles from Project-Based Learning and Gamification.

From the PBL perspective, GaPSEE is intended to provide good practices on how to adapt PBL principles to the context of software engineering education, in order to fill the gap between theory and practice. The principles of PBL are intended to introduce a real-world basis to the assignment, with convincing projects where students can practice theory, and apply appropriate tools and methods to develop solutions in context. However, it is not a PBL framework. Therefore, GaPSEE identifies a set of practices and elements of PBL that are useful and adapts them to the context of software engineering education. The focus of GaPSEE is the practical assignment, which can be used from the start to end of a course, or restricted to a portion of a course. Educational projects play central role in GaPSEE, as instruments to achieve desired learning outcomes, not only regarding technical knowledge, but also exposing students to other practical aspects that are often difficult to illustrate in lectures.

From the gamification perspective, GaPSEE provides instructors with a selection of game elements adapted to the context of project-based software engineering education. Therefore, gamification is used to present the educational project in the format of a game, providing directions and systematically rewarding and acknowledging students actions that support the achievement of learning goals or the execution of the project.

It is not included in the scope of this framework:

- Recommendations regarding evaluation methods (i.e., grading);
- Recommendations towards the learner perspective (i.e., lecturers are the target audience, not learners);
- Being a PBL framework (i.e. this framework adopts concepts and practices from PBL for the specific context of software engineering education);



- Being a Gamification framework (i.e., this framework proposes the use and contextualization of an initial set of game elements for the specific context of software engineering education);
- Suggestion of learning topics.

## 5.2 Target audience

GaPSEE is designed for software engineering lecturers who seek alternative methods for introducing practice in the teaching process. Considering the constructivist and active nature of PBL and Gamification in learning, GaPSEE target audience are lecturers who are comfortable with the idea of acting as facilitators rather than acting as the central role in the teaching process. The results of Rodrigues et al. [2018] motivate the development of the framework, showing that there is a demand for more accessible resources to support the adoption of games and gamification in software engineering education.

## 5.3 Components

GaPSEE is structured in three elements: guidelines, process, and suggestions for implementation. GaPSEE guidelines are a set of recommendations that summarize the lessons learned from the literature and empirical studies described in previous sections. GaPSEE process is a recommended workflow for the planning, execution, evaluation, and refinement of practical assignments in accordance to GaPSEE guidelines. GaPSEE suggestions for implementations are a set of observations on specific activities of the GaPSEE process that should be considered when instantiating GaPSEE in a practical assignment. The components are not prescriptive, but rather provide directions on relevant aspects of the planning and execution of practical assignments in software engineering education.

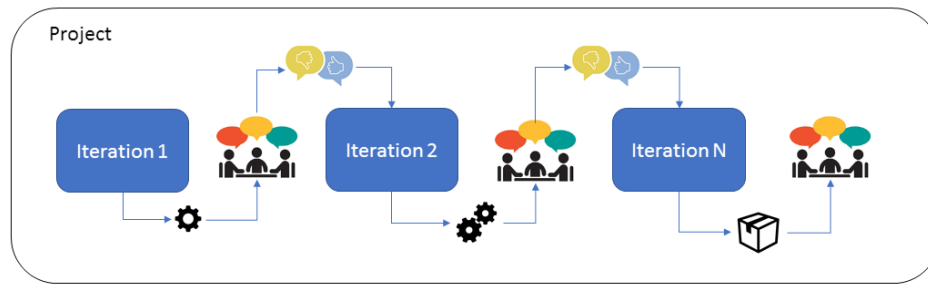
## 5.4 Structure and roles in GaPSEE practical assignments

A practical assignment defined in accordance to GaPSEE assumes the form of a software engineering project, with a goal that should be achieved by the development of a tangible product. A tangible product is any work product that can be objectively



verified and evaluated. Examples of tangible products include software applications, reports, specifications, documentation. Results such as acquisition of new knowledge, development of skills, and discussions are not tangible products.

The project is organized in iterations with clear goals, that sequentially culminate in the goal of the project. In each iteration there is a set of activities that should provide students with directions for the achievement of the iteration goal. These activities should allow students to have active voice, with room to decide and negotiate how to complete each task. Fulfilment of activities is confirmed by providing evidences that demonstrate their conclusion. The fulfilment of iterations is marked by the delivery of partial artifacts that address the goals of iterations. Figure 5.1 illustrates the expected structure of a practical assignment in accordance to GaPSEE.



**Figure 5.1.** GaPSEE project layout

This roadmap of activities is presented in the format of a game, in a competitive environment. Project iterations are presented as levels, with tasks or quests (the activities) that are worth points and badges for ranking teams in a leaderboard. Each task has specific deadlines, in order to motivate students to work in a structured schedule (simulating the context of professional software development). The fulfilment of tasks is validated by the lecturer with feedback, allowing and motivating students to make improvements to their work products (chance to fail). The conclusion of levels takes the form of a Boss Battle<sup>1</sup>, in which teams have to present their partial artifacts for the lecturers to validate them in accordance to the iterations goals. By the end of the project, the best ranked teams are rewarded with awards or recognition (e.g., position in a hall of fame).

Lectures should be planned in order to provide the necessary support for the execution of the project. Therefore, the project should be central to the organization of the pedagogical sequence of topics addressed in lectures, synchronized with the

<sup>1</sup>Boss Battle is a game element that introduces a special challenge at the end of a game level, that players have to overcome in order to complete that level [dos Santos et al., 2018a].



iterations of the project. Students work in teams, and should take protagonist role in the execution of the assignment. Lecturers act in a supportive role, providing feedback and helping teams in the pursuit of the project goals.

## 5.5 GaPSEE Guidelines

This section describes GaPSEE guidelines. These guidelines summarize lessons learned from literature and empirical studies, discussed in previous chapters. These guidelines are designed as recommendations for the conception of practical assignments in software engineering education. However, these guidelines do not define directions on “how” to systematically define a practical assignment, but general recommendations on “what” should and should not be considered in the design of assignments using gamification and PBL. Table 5.1 summarizes the 15 guidelines from GaPSEE, for which the justification is given in the following paragraphs. The first version of these guidelines was presented by Souza et al. [2019a].

**GaPSEE-01:** *“The assignment should have significant real-world basis, grounded in realistic problems or methods, in order to provide an interesting, concrete and convincing example of software engineering practice”.*

Justification: One of the guidelines of IEEE/ACM [2015] is that “the curriculum should have a significant real-world basis”. Therefore, the goal is to use the “Realism” characteristic of PBL [Blumenfeld et al., 1991], to introduce practical situations contextualized in real problems and/or real methods. Previous experiences (Chapter 4) show that creating an immersive environment that simulates professional software engineering activities is also observed as “realistic” by students, rather than only relying in real problems that are often difficult to identify and introduce in classroom.

**GaPSEE-02:** *“The project should balance realism (authenticity) with level of control and viability, in order to allow the achievement of learning goals while remaining significant”.*

Justification: The literature [Harms and Hastings, 2016; Delgado et al., 2017; Nguyen et al., 2013] and our previous experiences (Chapter 4) show that it is important to carefully balance realism with control in the planning of practical assignment. The use of real projects, with external stakeholders is interesting to provide more authenticity to a project. However, the use of real projects results in additional effort and risks related to ensuring the adequacy of the assignment to learning goals, to the availability of these stakeholders, and to the stability of business rules and goals. Therefore, realism is important, but cannot compromise the adequacy of the assign-



**Table 5.1.** GaPSEE Guidelines

ID	Guidelines
GAPSEE-01	The assignment should have significant real-world basis, grounded in realistic problems or methods, in order to provide an interesting, concrete and convincing example of software engineering practice.
GAPSEE-02	The project should balance realism (authenticity) with level of control and viability, in order to allow the achievement of learning goals while remaining significant.
GAPSEE-03	The assignment should be driven by a question or problem without a predetermined solution.
GAPSEE-04	The assignment should result in a series of artifacts that culminate in a tangible final product that addresses the driving question or problem.
GAPSEE-05	The assignment should use software process to contextualize software engineering practice and promote the use of appropriate and up-to-date tools, methods, and standards.
GAPSEE-06	The assignment should promote the development not only of technical knowledge, but also expose students to personal and professional skills related to software engineering.
GAPSEE-07	The assignment should allow students to work collaboratively with peers and lecturer to construct knowledge.
GAPSEE-08	Lecturers should act as knowledge facilitators, supporting students for the conclusion of the project and achievement of learning goals.
GAPSEE-09	The assignment should allow students to have significant autonomy of choice, motivating continued and self-directed learning for the development of solutions in context.
GAPSEE-10	Gamification should have software engineering as a central theme, tailored for specific contexts.
GAPSEE-11	Gamification should systematically support the development of specific behaviors that converge to expected learning outcomes or that contributes for the project execution.
GAPSEE-12	The gamification approach should provide students with directions for the execution of the project and mechanisms to track their progress.
GAPSEE-13	The gamification approach should promote social recognition of the students efforts.
GAPSEE-14	Gamification strategy and selection of game elements should be grounded in clear objectives, aligned with learning goals of the assignment.
GAPSEE-15	The assignment rubrics should provide clear indication of what students should focus on during the project execution, and the evaluation criteria should consider some aspects of the gamification approach.

ment to its learning goals. As stated by Harms and Hastings [2016], “projects need not to be too shallow and yet not be idealistic either”.

**GaPSEE-03:** *“The assignment should be driven by a question or problem without a predetermined solution”.*

Justification: The anchoring question or problem of the project should allow a variety of solutions, in order for the students to develop their own approaches to answering the question or solving the problem [Blumenfeld et al., 1991].

**GaPSEE-04:** *“The assignment should result in a series of artifacts that culminate in a tangible final product that addresses the driving question or problem”.*

Justification: The development of a tangible product by means of the creation of a series of partial artifacts is a key characteristic of PBL [Blumenfeld et al., 1991]. This characteristic is also interesting for the students to get used to a process-based approach of software engineering. This process also allows addressing and linking



different learning topics in a coherent sequence.

**GaPSEE-05:** *“The assignment should use software process to contextualize software engineering practice and promote the use of appropriate and up-to-date tools, methods, and standards”.*

Justification: One of the guidelines of IEEE/ACM [2015] is that software process should be central to the curriculum organization and to students’ understanding of software engineering practice. Additionally, the concept of a process makes it easier for lecturers to create a logical sequence of steps for the execution of the project addressing the diverse topics in the course syllabus. However, the implementation of this guideline should not conflict with the guideline GaPSEE-09.

**GaPSEE-06:** *“The assignment should promote the development not only of technical knowledge, but also expose students to personal and professional skills related to software engineering”.*

Justification: According to IEEE/ACM [2015], graduates in software engineering programs should be able to demonstrate not only technical knowledge, but also skills related to professional knowledge, teamwork, design of solutions in context, trade-offs, end-user awareness, and continuing professional development. Therefore, not only the assignment should promote development of technical skills, but also expose students to issues related to nontechnical aspects of software engineering, including management, communication, and teamwork.

**GaPSEE-07:** *“The assignment should allow students to work collaboratively with peers and lecturer to construct knowledge”.*

Justification: Teamwork is not only a key element in PBL [Blumenfeld et al., 1991; Thevathayan, 2018] but also an expected skill for software engineers [IEEE/ACM, 2015]. Therefore, the assignment should encourage collaboration.

**GaPSEE-08:** *“Lecturers should act as knowledge facilitators, supporting students for the conclusion of the project and achievement of learning goals”.*

Justification: In PBL, lecturers assume a supportive role, as knowledge facilitators, while students assume protagonist role on their learning process. This way, the assignment should encourage students to develop self-learning skills, with lecturers supporting the students with insights on how theory could be applied to achieve goals. For instance, Delgado et al. [2017] suggest that technical topics included in the syllabus should support the development of the project and should be covered early in the course.

**GaPSEE-09:** *“The assignment should allow students to have significant autonomy of choice, motivating continued and self-directed learning for the development of solutions in context”.*



Justification: Blumenfeld et al. [1991] state that in PBL, project should allow students to have active voice on how to achieve their goals and negotiate some aspects of the project. This is important for developing self-directed learning skills and for motivating students with freedom to use the technologies they desire or feel comfortable.

**GaPSEE-10:** *“Gamification should have software engineering as a central theme, tailored for specific contexts”.*

Justification: Literature on gamification of software engineering education revealed two approaches for using gamification in software engineering education [Souza et al., 2018]: the gamification of the classroom experience, and the gamification of specific software engineering practices. While the first is concerned in motivating and engaging students in classroom activities, the later is more concerned in promoting the use of specific software engineering abilities and practices, or in the development of specific skills. We have observed that the later approach is better suited for supporting practical approaches and PBL. However, it requires customization of the gamification approach to the specific courses or projects.

**GaPSEE-11:** *“Gamification should systematically support the development of specific behaviors that converge to expected learning outcomes or that contributes for the project execution”.*

Justification: In the study described in chapter 4 and in some studies described in chapter 2 [Akpolat and Slany, 2014; Singer and Schneider, 2012; Long et al., 2011], gamification was used as a device to motivate students in conforming to desired behaviors, such as the more frequent use of specific tools, acquiring the habit of applying specific techniques, or being more participative in the classroom. Gamification was also used as a strategy to induce learners to use specific software engineering abilities or practices, by promoting competition or systematically rewarding learners as they performed expected actions or showed expected behaviors. Therefore, gamification is a relevant strategy to support students in developing an appreciation of the importance of continued learning and in acquiring habits for professional software development [Souza et al., 2018].

**GaPSEE-12:** *“The gamification approach should provide students with directions for the execution of the project and mechanisms to track their progress”.*

Justification: Martin et al. [2014] consider that the ill-structured nature of problems is one of the cornerstones of PBL and one of the main sources of difficulties in using PBL. In our previous experiences, gamification provided the students with clearer goals they could use to plan and track their progress in the form of achievements, quests and badges (chapter 4). In the study described in chapter 3, badges were perceived as secondary goals for student to strive for in addition to approval in the course, and



elements such as leaderboards provided additional instruments for students to compare their performance with classmates. Therefore, gamification should be used to define a roadmap of activities in a game format.

**GaPSEE-13:** *“The gamification approach should promote social recognition of the students efforts”.*

Justification: Gamification should be used to support systematic recognition of students behaviors in the form of feedback, and to increase the social status of students, as they work their way to the top positions of leaderboards and hall of fame in the course.

**GaPSEE-14:** *“Gamification strategy and selection of game elements should be grounded in clear objectives, aligned with learning goals of the assignment”.*

Justification: As described in guideline GaPSEE-10, GaPSEE adopts gamification as an instrument to provide incentives for the development of specific behaviours and skills, and it requires appropriate tailoring for each context. However, it is important to define goals for gamification. For instance, gamification can be used to increase students confidence in performing tasks, for motivation, for competition. Therefore, it is important to define the purpose of gamification and define only mechanics that support these goals. The selection of game elements that are not focused or aligned with the expected outcomes of the assignment, may lead to students’ overhead, or to deviation of the focus to the game and not to learning aspects.

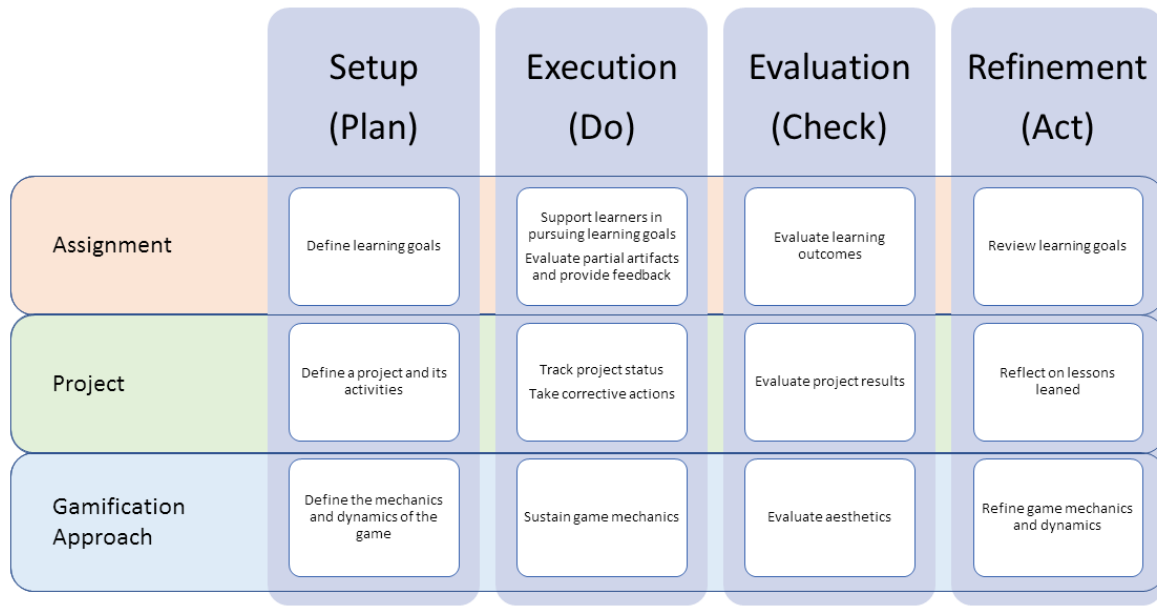
**GaPSEE-15:** *“The assignment rubrics should provide clear indication of what students should focus on during the project execution, and the evaluation criteria should consider some aspects of the gamification approach”.*

Justification: Defining and providing students with objective evaluation criteria is a valuable resource for keeping students from feeling lost or confused about what should they focus [Fagerholm and Vihavainen, 2013]. The assignment rubrics should clearly define how the performance on the gamification aspect of the assignment is translated to grades, and what are the boundaries between grades and gamification scores.

## 5.6 GaPSEE Process

GaPSEE process is a suggested workflow of activities to support the planning, execution, evaluation and refinement of practical assignments, in accordance to GaPSEE guidelines. This process adopts a PDCA (Plan-Do-Check-Act) cycle, for three dimensions: the assignment, the project, and the gamification. The assignment dimension





**Figure 5.2.** Expected actions for each component of GaPSEE

is related to the definition of the learning goals of the practical activity. The project dimension is related to the planing of the projects goals and activities. And the gamification dimension is related to the definition of the game elements and mechanics to support the assignment. Figure 5.2 illustrates PDCA phases for each dimension.

The GaPSEE process streamlines these activities in a structured process, presented in Figure 5.3. The process is composed of four phases: Setup, Execution, Evaluation, and Refinement. However, none of these phases and their activities are mandatory. This process is a suggestion of sequential steps in order to systematically apply the recommendations proposed in GaPSEE. The following subsections describe the goals, expected actions and expected outcomes of the activities for each phase.

### 5.6.1 Setup phase

The goals of this phase are to: define the expected learning outcome of the educational project; plan a software engineering project from both a PBL and a Gamification perspective; and prepare useful resources for the assignment execution.

The Setup Phase includes the following activities: “Define Learning Goals”, “Plan the Educational Project”, “Establish a Gamification Strategy” , and “Prepare Resources”.

The learning goals or outcomes define what knowledge is expected for the students to learn or skills to develop. The educational project is the mean for the students



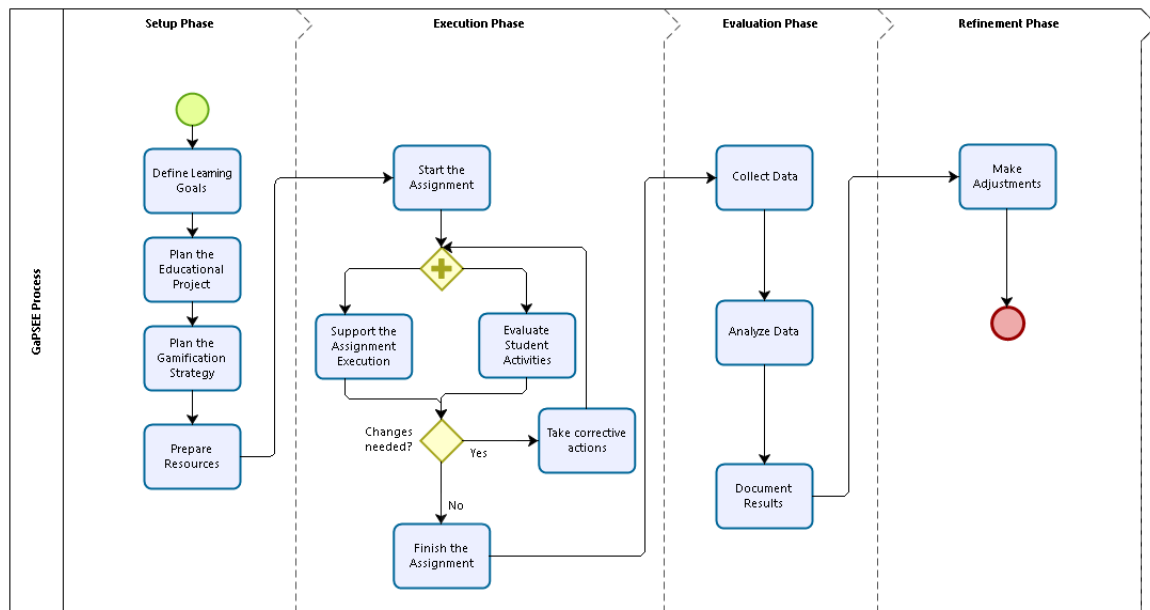


Figure 5.3. GaPSEE process

to achieve the learning goals. The gamification strategy is the form of delivering the project, focusing on enriching the learning experience. The following paragraphs describe details of each activity.

**Define Learning Goals:** The goal of this activity is to plan the expected learning outcomes for the educational project. Therefore, it is important to define (i) which topics of the course syllabus should be addressed in the assignment; (ii) expected learning outcomes, both in terms of specific skills related to the learning topics of the course, and general skills relevant for software engineering professionals; and (iii) a driving question or problem, to foster inquiry along the assignment.

**Plan the Educational Project:** The goal of this activity is to define a software engineering related project to address the driving question or problem, and the learning goals of the assignment. This includes: defining the goal and scope of the project; planning the people involved (in case of external stakeholders); and defining iterations and activities for the project (a project roadmap).

**Plan the Gamification Strategy:** The goal of this activity is to plan how gamification is going to be incorporated in the assignment. Therefore, this activity includes: definition of goals for the gamification; selection of game elements; and definition of game mechanics and dynamics.

**Prepare Resources:** The goal of this activity is to prepare all necessary materials for the execution of the assignment (including tools, environments, handout



materials and rubrics). We recommend that at least the following information should be made available for students in anticipation to the start of the assignment:

- Assignment instructions: Clear information about the goal of the assignment, learning topics involved.
- Project schedule and plan: The details on the scope of the project, activities, deadlines, milestones, stakeholders, expected results, etc.
- Game rules: Details on the mechanics of the game, regarding how participants are expected to interact with instructors, what are the goals (tasks, challenges, achievements), how to deliver evidences of completed tasks, how to score points, what are constraints, and other relevant information for students to know how to behave in the game.
- Evaluation Rubrics: Objective details on the rubrics to be used for the evaluation and grading of students. Special attention should be given to how gamification and PBL relate to students' grades. The open nature of PBL activities may be confusing for students. As students direct their activities based on the given assessment criteria, the assessment design plays a key role in what students will focus on Fagerholm and Vihavainen [2013](*GAPSEE-15*). It is also important to link the gamification approach to the evaluation strategy, in order to provide students with converging incentives for pursuing desired outcomes (*GAPSEE-16*).

### 5.6.2 Execution phase

The goals of this phase are to: kick-off the project; guide and support students to achieve learning goals; guide and support students in the project execution; reinforce the execution of the game mechanics; review the status of the assignment, project and gamification approach; and take corrective actions when needed. The activities of this phase are:

**Start the Assignment:** The goal of this activity is to perform the kick-off of the project, describing the rules of the assignment and getting the commitment of students and relevant stakeholders. For the assignment, it is important to communicate the learning goals, the driving question, and the evaluation rubrics. For the project, it is important to communicate the goals, the schedule, the people involved (and their roles), and details about the process (e.g., mandatory tools, communication channels, and standards). For the gamification approach, it is important to detail the mechanics of the game and, if applicable, how the game mechanics relate to evaluation rubrics.



**Support the Execution of the Assignment:** The goal of this activity is to monitor the status of the project, to track the students' progress, to provide guidance for students towards achieving expected learning outcomes, and to sustain gamification along the assignment.

**Evaluate Students Activities:** The goal of this activity is to review the artifacts produced by students and to provide relevant feedback.

**Take Corrective Actions:** The goal of this activity is to perform changes in the assignment, project or gamification mechanics during the execution of the assignment to address problems and deviations. In case of problems or deviations in the assignment execution, corrective actions may be necessary in order to ensure the achievement of the expected learning outcomes. It is important to negotiate and communicate any changes in the assignment, project, and gamification approach to all parties involved (students, instructors, and external members).

**Finish the Assignment:** The goal of this activity is to conclude the assignment, with proper reflection on the outcomes of the project.

### 5.6.3 Evaluation phase

The goal of this phase is to evaluate the outcomes of the assignment in the perspective of students and instructors, regarding learning outcomes, project results, and gamification goals. This activity is not related to course evaluation or grading process, nor the instructors' assessment of students specific activities. This activity is intended to evaluate the level of success of the assignment, and possible causes for positive and negative outcomes, for improvements for future replications of the approach. Therefore, it is important to collect and analyze data on the perception of all stakeholders of the assignment (students, lecturers, and external members). Therefore, the activities of this phase should be executed after the conclusion of the course or assignment, because: (i) only at the end of the course, the lecturer could have insight on the students' accomplishment of learning outcomes; and (ii) students could provide authentic opinion on the course without fear of compromising their grades or performance in the course. The activities of this phase are:

**Collect Data:** The goal of this activity is to prepare and apply appropriate instruments to collect data about the performance and perception of all stakeholders involved in the project about the assignment execution and results. This includes students, instructors, and external members. The data to be collected should be in accordance to definitions of the planning phase.

**Analyze Data:** The goal of this activity is to analyze the data collected in the



previous activity and identify possible lessons. The instructors should use appropriate quantitative or qualitative methods to evaluate the data. For qualitative data obtained from questionnaires, interviews and post-mortem analysis, techniques from ground theory (such as open coding) may be useful to identify key information from unstructured responses. For objective answers, descriptive statistics may be suitable to describe and graphically present interesting aspects of the data [Wohlin et al., 2012].

**Document Results:** This activity aims to document the lessons learned from the analyzed data to provide inputs for future instalments of the assignment (in case of recurring courses), for planning new assignments for future courses, or for maintaining a data set for benchmarking. Additionally, this phase may result in reports for academic or educational communities. Therefore, it is important to document the settings used for the assignment (as defined in the “Setup Phase”) for long term analysis of changes that lead to improvements in the assignment evaluation.

#### 5.6.4 Refinement phase

The goal of the “Refinement Phase” is to plan adjustments for future assignments based on the lessons learned. This phase is composed of a single activity “Make adjustments to the assignment”, in which the instructors should review the learning goals, evaluation rubrics, project details, and gamification elements.

The lessons documented in the previous phase may indicate the need for adjustments of the assignment for future installments. Possible signs of this necessity are: observed learning outcomes below the expected goals, too much effort being directed to secondary activities in the project, recurrent failure in meeting deadlines or addressing the project scope, low participation of external stakeholders, etc.

Therefore, it is suggested that, among other factors, the assignment is reviewed regarding: learning goals; evaluation rubrics; project stakeholders and their roles; project process and deadlines; support tools and environments; gamification mechanics, dynamics and aesthetics.

### 5.7 Suggestions for implementation

This section describes general considerations about each phase of the GaPSEE process, which may be useful for the implementation of GaPSEE in software engineering related courses. These considerations are exemplified by referencing a pilot study executed in an introductory software engineering course (“PLT case”, henceforth).



### 5.7.1 Planning the assignment

In the activity “Define Learning Goals” (Section 5.6.1), the assignment should be planned considering two types of skills to be developed by the students, which we refer as “specific skills” and “general skills” (*GAPSEE-06*). The specific skills are directly related to topics of the course syllabus that are planned for the assignment. These skills are related to technical knowledge. For instance, for the PLT case, these specific skills were related to: software requirements elicitation; software requirements specification; software requirements verification and validation; software architectural design; software construction planning; managing software construction; detailed design and coding; debugging and testing; software test planning; and software testing techniques.

The general skills are related to personal skills expected from software engineers. For instance, in the PLT case, we considered the qualities desired for software engineering graduates (ACM/IEEE, 2015):

**Professional Knowledge:** Show mastery of software engineering knowledge and skills and of the professional standards necessary to begin practice as a software engineer.

**Technical Knowledge:** Demonstrate an understanding of and apply appropriate theories, models, and techniques that provide a basis for problem identification and analysis, software design, development, implementation, verification, and documentation.

**Teamwork:** Work both individually and as part of a team to develop and deliver quality software artifacts.

**Design Solutions in Context:** Design appropriate solutions in one or more application domains using software engineering approaches that integrate ethical, social, legal, and economic concerns.

**Perform Trade-Offs:** Reconcile conflicting project goals, finding acceptable compromises within the limitations of cost, time, knowledge, existing systems, and organizations.

**End-User Awareness:** Demonstrate an understanding and appreciation of the importance of negotiation, effective work habits, leadership, and good communication with stakeholders in a typical software development environment.

**Continuing Professional Development:** Learn new models, techniques, and technologies as they emerge and appreciate the necessity of such continuing professional development.

The driving question guides the inquiry of the assignment, leading students to



reflect on practical aspects of the adoption of the theory from the topics of interest along the development of the expected outcomes. This driving question or problem is the theme or anchor of the assignment, and the reference to support the selection of relevant projects for the course. For the PLT case, the driving question was: How can we systematically design and develop a software product to meet customer needs?. The defined question leads to outcomes that are within the scope of the course, however the approaches to be taken and the results should not be predefined (*GAPSEE-03*). A counter example would be to ask students to apply a specific bad smell detection tool against a controlled data set and objectively evaluate the precision of students responses. Although the later assignment would fit the course, it does not take advantage of the constructivist and explorative ideas of PBL, giving little room for reflection.

Table 5.2 summarizes the planning of the assignment for the PLT case. This assignment was planned to be executed in the entire duration of the course.

**Table 5.2.** Learning goals of the assignment in PLT case.

<b>Driving Question</b>	"How can we systematically design and develop a software product to meet customer needs?"
<b>Topics of Interest</b>	Software Requirements; Software Design; Software Construction; Software Testing.
<b>Specific SE Skills</b>	<p><b>Software Requirements:</b> Software Requirements Elicitation; Software Requirements Specification; Software Requirements Verification and Validation.</p> <p><b>Software Design:</b> Software Architectural Design.</p> <p><b>Software Construction:</b> Software Construction Planning; Managing Software Construction; Detailed Design and Coding; Debugging and Testing.</p> <p><b>Software Testing:</b> Software Test Planning; Software Testing Techniques.</p>
<b>General SE Skills</b>	<p><b>Professional Knowledge:</b> Understand and apply practices and tools from professional software development.</p> <p><b>Teamwork:</b> Plan and execute activities in teams of 3 to 5 students. Use tools to support collaborative work (version control systems and project management tools).</p> <p><b>Design Solutions in Context:</b> Develop a software product considering aspects related to application domain, business domain, and specificities from customer needs.</p> <p><b>Perform Trade-Offs:</b> Prioritize requirements and critically analyze technologies and design strategies, balancing cost (effort) and quality.</p> <p><b>End-User Awareness:</b> Interact with requirement provider to understand, validate, and negotiate functional and non-functional requirements.</p> <p><b>Continuing Professional Development:</b> Investigate state-of-the-art tools and methods for software development.</p>

When defining the learning goals of the assignment, relevant source materials may



be considered, such as: international or local curricular guidelines, bodies of knowledge, and reference models. For instance, SE 2014 [IEEE/ACM, 2015] provides recommendation on topics and skills that should constitute an undergraduate software engineering education. Similarly, SWECOM [Ardis et al., 2014] provides a comprehensive body of knowledge for skills and competences that are expected from software engineers (for instance, the specific knowledges in the PLT case are derived from SWECOM). Finally, best practices bodies of knowledge and industry reference models, such as SWEBOK [Bourque et al., 2014] and CMMI [Team, 2010], may also provide directions on important topics and skills for software engineers.

### 5.7.2 Planning the project

According to [Bender, 2012], PBL may be defined as using authentic, real-world projects, based on a highly motivating and engaging question, task, or problem, to teach students academic content in the context of working cooperatively to solve the problem. Therefore, in GaPSEE, the main PBL principle to be considered is the authenticity of PBL projects, in order to motivate learning through scenarios that resembles real contexts for software engineering practice (*GAPSEE-01*). Projects should be grounded in the real world. It means that it is interesting that the project fits in one or more of the following criteria:

- **Authentic Problems:** The project goal is anchored in real-world problems. It may be designing a software to address the problem of a real organization, assessing the quality of a real product or process, or creating guidelines for a problem from the literature.
- **Authentic stakeholders:** The project has external stakeholders actively participating in the process. It may be a real customer providing requirements, a real customer to validate a prototype, an open-software community to evaluate pull requests, or dealing with real end-users of an application.
- **Authentic methods:** The project uses real world process, tools or standards, simulating the professional environment. It may be a project that requires a process in adherence to specific reference models (e.g., CMMI for development), the adoption of community rules for contributing to an open-source project, or simulation of processes used in real professional environment.

Lecturers should consider the balance between authenticity and control of the assignment (*GAPSEE-02*). Addressing real problems may limit the coverage of topics



of the course syllabus that are relevant for the project. Using external stakeholders for providing requirements or validation of work products may lead to risks related to availability and stability of business rules. However, relying in fictitious projects, may lead to decreased immersion. Therefore, we suggest that lecturers contextualize the projects in authentic problems that are relevant for students, and focus on the use of real methods, processes and tools for the execution of the project, simulating the professional practice of software engineering. In case of external stakeholders, we suggest that lecturers act as interface between them and the students.

The project goal must be aligned to the driving question of the assignment. In accordance to the expected characteristics of a PBL project, it has to result in a tangible product (*GAPSEE-04*). Therefore, the project goal must drive students to the development of a product, and, during its execution, expose learners to situations that encourage them to investigate the relevant learning topics.

A “tangible product” is a concrete result composed of one or more artifacts created by the learners. In the context of software engineering, this final product may be a software product, a software specification, a set of guidelines, reports, and other relevant artifacts related to the software engineering practice. It means that the goal of the project cannot be “learning”, “understanding”, or similar abstract and immeasurable achievements, as those goals are part of the expected learning outcomes. The project has to be the instrument to allow achieving these learning goals, but it has its own goal in the form of a final product that addresses the driving question of problem of the assignment.

In order to provide guidance for the development of the project, GaPSEE suggests instructors to provide a set of desired activities for students. The idea is that these activities provide a roadmap for the project in the form of a light process. Therefore, these activities should not be strict tasks with rigid predefined outputs. Instead, these activities should preferably give students room to reflect, plan and execute them with varied approaches (*GAPSEE-09*). For instance, an activity such as “identify customer needs” is open enough to allow students to investigate how to collect the customer requirements and how to document it. In contrast, the activity “interview the customer to elicit needs” with a requirement document template provided for students, may limit the students opportunity to negotiate, investigate and plan the activity. Additionally, the first scenario provides more room for students and instructors to interact and reflect on different approaches. We are aware that some activities cannot have this level of openness, however, it is interesting to pursue a balance.

In accordance to *GAPSEE-04*, in order to create a tangible product, the project may result in a series of partial artifacts. However, these artifacts are not necessarily



mapped one to one with activities. We suggest that the project should be planned in phases or iterations with a set of activities. Each phase or iteration results in a partial artifact that is relevant for the end product. The idea is that these phases or iterations (i) provide students with a notion of a process, (ii) provide a roadmap for the project, and (iii) provide control points or milestones to evaluate and reflect on the status of the project (*GAPSEE-05*). However, depending on the size and duration of the project, it may not be viable to segment the project in more than one iteration. In our experiences, we have always opted for using three iterations, which has been successful.

For the PLT case, the goal of the project of the teams was “the development of a web application to support management of practical assignments”. Table 5.3 describes the goal and expected artifacts of each iteration of the project. Table 5.4 exemplifies the activities for one iteration of the PLT case.

**Table 5.3.** Iterations of the PLT case

Goal	Expected outcome
1. Specify project needs and plan the project.	1. Requirement specification. 2. Selection of support tools. 3. Project baseline.
2. Design, implement and test the first increment of the project.	1. Software design description. 2. Software increment. 3. Verification reports. 4. Project baseline.
3. Design, implement and test the second increment of the project.	1. Revised software design description. 2. Software increment. 3. Verification reports. 4. Project baseline.

**Table 5.4.** Tasks of the second iteration of the PLT case

ID	Task
#18	Negotiate sprint backlog.
#19	Select development technologies.
#20	Design the high-level structure of the system.
#21	Describe an architectural design pattern for the system.
#22	Document the detailed design of the system.
#23	Suggest 2 design patterns for the system.
#24	Develop software in accordance to requirements and detailed design.
#25	Establish and apply project standards for coding.
#26	Document code through comments.
#27	Define commit rules for Git in the project Readme file.
#28	Support traceability between requirements and backlog items.
#29	Support and maintain traceability between commits and backlog items.
#30	Deliver at least 60% of the sprint backlog.
#31	Create test cases.
#32	Test the software and report defects.
#33	Report project status.
#34	Document project baselines.
#35	Validate the product increment.



### 5.7.3 Planning gamification

As mentioned before, gamification can focus on classroom experience or on specific software engineering practice behavior. In this activity of GaPSEE, the focus is on the second approach. Therefore, the design of the gamification approach should incorporate software engineering as the main theme, and it should be tailored to the specific context of the educational project planned in the previous activity (*GAPSEE-10*). Despite the fact that gamification introduces a ludic aspect to the serious context of education, grounding it on the specifics of software engineering practice, may not only create a more attractive assignment, but also help to make it more meaningful.

Therefore, the gamification approach should promote concepts and principles of software engineering, and reinforce the use of specific practices and skills that are in accordance to the project activities and to the desired learning outcomes. The gamification approach should support instructors in guiding students on how they should behave and execute activities, by systematically rewarding and acknowledging desired behaviors (*GAPSEE-11*). For instance, gamification could promote professionalism by rewarding students that meet deadlines accordingly, or who conform to specific standards or quality criteria in the development of artifacts.

We suggest that lecturers should define goals related to feelings that are expected to be instilled in students during the assignment. These goals are similar to the concepts of “Aesthetics” in the MDA (Mechanics, Dynamics and Aesthetics) framework [Hunicke et al., 2004]. In MDA, game design is organized in three elements: “Mechanics” represent structured use of game components that create the game rules; “Dynamics” are the interactions that emerge from mechanics during gameplay; and Aesthetics are the emotions that result from the gameplay. Therefore the first step is the definition of aesthetics or gamification goals (*GAPSEE-14*). In our previous experiences (including the PLT case), we considered the following goals for gamification: challenge, confidence, collaboration, engagement, fun, learning, motivation, relevance, recognition, and satisfaction.

Based on previous experiences and findings from the literature, we propose an initial set of game elements we found relevant for the achievement of these goals (Table 5.5). However, as a framework, this is not an exhaustive list of game elements nor are these elements mandatory. Instructors are free to adapt these elements to their needs or preferences, and to include different elements that are not present in this initial set. In case the goal is to create a more generalist framework to support gamification, the following resources may be useful: “Six Steps to Gamification” or 6D Werbach and Hunter [2012]), 5H2W [Klock et al., 2016], or Octalysis [Chou, 2015].



**Table 5.5.** GaPSEE suggestion of game elements to achieve gamification goals.

Element	Description	Goal
Quests,missions, or tasks	Set specifics objectives that users must complete in order to progress in the game or to earn rewards.	Challenge, Confidence, Learning, Engagement, Relevance.
Levels	A section or part of a game with specific goals. A progress unit in the game.	Challenge, Confidence, Learning, Engagement, Relevance.
Points	Reward for the accomplishment of specific actions or achievements.	Engagement, Fun, Motivation, Recognition, Satisfaction.
Leaderboard	Visual representation of the team position in a ranking system, based on any metrics related to the progress in the game (e.g., points, badges, completed levels).	Engagement, Fun, Motivation, Recognition, Satisfaction.
Time pressure	Time constraints related to the execution of actions in the game.	Challenge, Collaboration, Engagement, Fun, Motivation, Relevance, Satisfaction.
Feedback	Immediate response of the game in relation to player actions. Used to notify players of accomplishing any relevant action in the game.	Confidence, Collaboration, Learning, Motivation, Relevance.
Rewards	Recompense, award or prizes related to accomplishment of specific actions or achievements in the game.	Collaboration, Engagement, Fun, Motivation, Recognition, Satisfaction.
Badges	Reward in the format of visual representation of the accomplishment of specific actions or achievements.	Fun, Motivation, Recognition, Satisfaction.
Boss Battles	Specially difficult challenges, usually related to the conclusion of levels in games.	Challenge, Engagement, Fun, Learning, Motivation.
Hall of Fame	Visual recognition of the top players in the game.	Fun, Motivation, Recognition, Satisfaction.
Narrative	A story to contextualize gamification.	Collaboration, Engagement, Fun, Learning, Motivation, Relevance.













These game elements were organized in mechanics and dynamics tailored to the specific context of software engineering education. These mechanics and dynamics are organized in five aspects of the assignment, that should be considered when designing the gamification approach: the context or scope of the project; the process; rewards; feedback; and social aspects.

For the “context or scope of the project” dimension, instructors might find relevant to create a fictitious narrative to contextualize the project, especially when there is not a real customer interacting with the students. Therefore, the game element “Narrative” can be used to create a more immersive environment for the project, consequently creating a more convincing context for competition. For instance, in the PLT case, the project was contextualized as a stage of a fictitious bidding process to select a provider for an educational organization, where teams act as small software companies competing for the contract. Therefore, we introduced a narrative to foster the competition between teams, and to motivate the collaboration between students.

To organize the process as a game (process dimension), the project iterations may be presented as game levels, grouping a set of activities and having a clear goal. In each level, the activities planned in the “Plan the educational project” activity can be defined as tasks, missions or quests, each one with specific deadlines (time pressure).



These tasks can have different levels of challenge, and can be described as mandatory (main quests) or optional (side quests). By the end of each level, the teams have to obtain the approval or validation of the lecturer, in a presentation. This “confrontation” can be enacted as a Boss Battle. Therefore, in each level, the students have to plan which tasks they are going to execute and how to complete them, giving them choices and reinforcing reflection and trade-offs by having students planning how to progress in the project. These mechanics also enable a sense of goal-orientation and the ability to measure the progress in the project/game (*GAPSEE-12*). These dynamics can be augmented if supported by visual indications of progress, such as progress bars or navigation maps with indication of the status of the project. Figure 5.4 shows the tasks for the first level of the PLT case.

Level 1: Requirements and Planning					
Week 1		Week 2		Week 3	
	Create a Git-Hub project		Define the project scope		Create user interface prototypes
	Define policies for the use of Git-Hub (regarding access rules and directory structure)		Specify system requirements		Define the project development life-cycle
	Elicit customer needs		Establish verification criteria for the requirements		Define the product backlog in a management tool
	Define functional and non-functional requirements		Define policies for the use of Git-Hub (regarding access rules and directory structure)		Create a Project Baseline

**Figure 5.4.** First level of the assignment in PLT case.

Rewards create an instant sense of achievement. They represent benefits a user gets for completing some action or reaching some achievement. Rewards are the most basic mechanics to provide positive feedback for players when they are doing the right expected actions in games. Points, badges, physical goods, and in-game currency are all examples of rewards. In GaPSEE, we strongly recommend the use of points and badges to create a more entertaining and game-like experience for the assignment. Not only for competing against others, these elements may also provide students with the opportunity to create personal goals and compete with themselves.

Badges and points are specially recurring game elements in gamification approaches. In GaPSEE they can be used to reward the successful conclusion of tasks. For instance, in the PLT case, each task was associated to a trophy (example of badge), with different tiers associated to their complexity (bronze for easy tasks, silver for medium tasks, and gold for difficult tasks). Therefore, students could track the progress of all teams by identifying which trophy each team had obtained. Each trophy was also



associated to points, that was used as a score system to rank teams in a leaderboard. These mechanics supported competition, completion, and progress dynamics.

For the “Feedback” dimension, we suggest using feedback as incentive for students to execute tasks in time. In PLT case, the conclusion of tasks were not graded, only the artifacts and presentation delivered at the end of each level. However, these tasks guided students towards the development of these artifacts and presentations in accordance to expected criteria for the evaluation. Therefore, the lecturer provided feedback for teams each time they completed tasks in time. The feedback included confirmation of the appropriate execution of tasks (which would lead to appropriate artifacts by the end of the level) or suggestions for improvements. Consequently, this mechanic provided students with the chance to fail (i.e., students could retry, considering the improvements suggested by lecturer’s feedback), and systematic incentive to follow the project schedule. The adoption of badges, progress bars, notifications, scores and other components also contribute for providing feedback on the students’ actions. The goal is to create an environment where students have some freedom to experiment, creating a feeling of confidence.

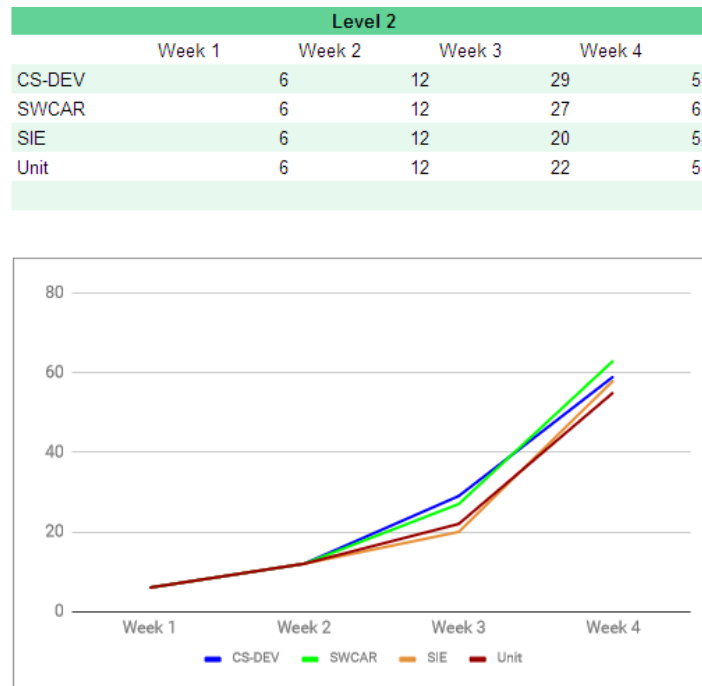
In the “social recognition” dimension, leaderboards, scoreboards, and hall of fame are elements that can promote social status among students and reinforce competition dynamics. In our previous experiences, these components also helped students in understanding their performance in contrast to their peers, and motivated them to try to improve their performance. Badges are also another mechanism for students to showcase their achievements and progress in the project, leading to the feeling of social recognition. Social recognition may provide an important motivator factor for students (*GAPSEE-13*). However, it is important to remember that, in an educational environment, competition may not please everyone. For instance, students with lower performance may feel demotivated or even embarrassed. The use of avatars and nicknames or alias, may help introducing some level of anonymity. Figure 5.5 shows the weekly leaderboard used in the PLT case.

#### 5.7.4 Executing the assignment

The instructors must periodically assess the achievement of the learning goals. Gary [2015] points that, given the open-ended nature of PBL activities, each student has different learning experiences, and no two projects or teams are the same, however the learning goals are the same for everyone. Therefore, students need the right amount of guidance towards learning outcomes, and tracking their progress is difficult.

As suggested in the PBL method, the project should take a central role in the





**Figure 5.5.** Leaderboard in the PLT case.

learning process, therefore, we suggest instructors to use classroom time to explore the assignment project in collaboration with the students (*GAPSEE-07*). Classroom time can be used to provide hands-on guidance in the use of specific techniques and tools, or at least provide specific examples using the students' projects when lecturing specific topics that students are expected to apply on their assignments.

During the assignment execution, the instructors must reinforce the students' commitment to the activities of the project and to the evaluation rubrics of the assignment. The first provides students with clear goals for what they have to do to advance in the project, and the later provides students with objective criteria about how they are supposed to advance in the project (*GAPSEE-15*). As a facilitator, it is the instructor's responsibility to motivate students about the importance of ("why") the actions they are performing (*GAPSEE-08*).

As the project plays a central role in the assignment (and in the learning process, if adopting PBL), it is important to check its progress continuously. Instructors should periodically check if the project is still viable, i.e., if the students are able to develop the final product in the planned schedule, if the tools and processes are adequate, and if there is still commitment from the parties involved. Changes may be necessary in order to ensure that the project is adequate for the assignment purpose, for the people involved, and for the time available.



In GaPSEE, we suggest that the instructors guide students towards the identification of evidences of performing the activities, instead of relying in providing predefined documents templates for students to fill or strict format of artifacts to be produced. This suggestion comes from our previous experiences (Chapter 4) where we observed that students associate documentation to beadedness. Additionally, by giving freedom to the students to choose how to demonstrate the completion of an activity, we are also giving them more room to reflect on what strategy they will use to actively keep track of their activities. The goal is to induce students to think more about the activity they are doing, than in the manual process of filling documents. Evaluating these evidences periodically allows instructors to understand the progress of students project, and to anticipate misinterpretations or difficulties students may face during iterations. The number of activities successfully completed allows instructors to quantitatively track the progress of the project. The control points at the end of each iteration allow instructors to qualitatively assess the status of the project and to evaluate the students in relation to the learning outcomes.

Giving feedback to students is very important in GaPSEE, as it allows students to continually improve and to reflect about their mistakes. However, the instructors may have time restrictions, specially in large classes, to continually provide feedback for all activities performed by students, in timely manner. Gamification mechanics may be used to limit the number of feedback a team is allowed to have for each activity, or to limit the time-frame in which the opportunity to receive feedback is possible. Therefore, students are rewarded for the good behavior of meeting deadlines.

However, we advise instructors not to convert gamification points directly into grades. The progress of students in the gamification aspect of the assignment should give them indication of how they are progressing in the right direction. We advise lecturers to consider varied criteria to evaluate students, as teams and individually, such as participation, collaboration, demonstration of technical or professional knowledge, and others (*GAPSEE-15*).

By the end of each phase or iteration, the lecturer should promote reflection on the results. These milestones are important for assessing the status of the project. From the instructor perspective, it is a major opportunity to evaluate the progress of students and take corrective actions in the assignment structure, if needed. For students, it is a relevant opportunity to share experiences with classmates and with instructors, and to receive objective feedback on the progress of their projects. Each phase ends with a reflection activity planned as a presentation and a discussion.

From the gamification perspective, it is important to ensure the execution of the game mechanics and to monitor if the expected dynamics are emerging from these



mechanics. In case of the use of software tools to support specific game elements and mechanics (such as management of the point system, handling and keeping track of badges, or maintaining a leaderboard), this activity requires little effort. However, in case of manual approaches, this activity may require some additional attention.

Some mechanics may not work properly as intended, therefore instructors need to observe mechanics that are often violated, forgotten, that are difficult to implement, or that are causing unexpected dynamics that are leading students to negative outcomes. Control points of the assignment should be used to discuss problems in the gamification mechanics and reflect on the possibility of changing some rules.

In case of problems or deviations in the assignment execution, corrective actions may be necessary in order to ensure the achievement of the expected learning outcomes. It is important to negotiate and communicate any changes in the assignment, project, and gamification approach to all parties involved (students, instructors, and external members).

## 5.8 Final remarks

This chapter presented GaPSEE, a framework to support lecturer in the planning and execution of practical assignments for software engineering education using principles of PBL and gamification. The framework is composed of guidelines, a process, and suggestions for implementation. The design of the framework is intended to use the best qualities of PBL and gamification to address the challenges of contextualizing practice in software engineering education. PBL is a comprehensive method to introduce practice in classroom, engaging students in team activities, and encouraging the development of both technical and personal skills. However, the adoption of PBL suffers from some challenges. Similarly, gamification is a recent trend that requires considerable effort to adapt to each context appropriately.

Therefore, this framework proposes that PBL and gamification can be used to compliment each other: PBL provides meaningful contextualization for software engineering practice in educational environment, while gamification adds systematic acknowledgement and feedback for students actions, providing motivation, guidance, and confidence in the learning process. Consequently, the novelty of this framework lies on: (i) proposing a set of recommendations for the adoption of PBL and gamification considering the specific characteristics of software engineering education; (ii) using PBL and gamification complementing each other; and (iii) proposing a reusable structure for practical assignments in a game format, preserving relevant concepts and philosophy



of PBL.

In this chapter, the description of GaPSEE is focused on its core components and recommendations. However, a practical guide for the use of this framework is in continuous development at <http://www.gapsee.com.br> .

In the following chapter, this framework is evaluated in a series of case studies, in the perspective of lecturers and students with respect to its adequacy in defining practical assignments for software engineering education.







## Chapter 6

# Evaluation of the Proposed Framework

This chapter describes the planning, execution and analysis of a set of case studies to evaluate practical assignments defined in accordance with the recommendations of GaPSEE. The evaluation was based on case studies with four professors from three federal universities in Brazil. These lecturers applied GaPSEE to setup and execute practical assignments using PBL principles and gamification in five distinct software engineering related courses: two software engineering courses, a Web development course, a software quality and measurement course, and a software processes course. After the execution of the assignments, the students of these courses were invited to answer a questionnaire, and the lecturers were interviewed by the researcher.

This chapter is organized as follows. Section 6.1 describes the study design, research questions, methods, and procedures for data analysis. Section 6.2 describes the planning and execution of the case studies used for the evaluation of the GaPSEE approach. Section 6.3 presents the execution of a survey study to collect the students' perception on the GaPSEE approach, as it was implemented in the practical assignments of the case studies. Section 6.4 describes the results of interviews with the lecturers of each case study. Section 6.5 promotes a discussion of the results in relation to the research questions of the study. Section 6.6 discusses possible threats to the validity of the study and actions performed to mitigate them. Finally, Section 6.7 closes the chapter with final remarks.



## 6.1 Study settings

This section describes the planning and execution of this study, in relation to its goal, research questions, and the research strategy.

### 6.1.1 Study goal and research questions

The goal of this study is to evaluate GaPSEE, regarding its adequacy to software engineering education, in the perspective of lecturers and students, in the context of practical assignments planned and executed in accordance with the recommendations of the framework.

In order to achieve this goal, we defined three research questions to guide the execution of this study:

**RQ1.** What is the lecturers' perception towards a practical assignment planned and executed in accordance with the recommendations of GaPSEE?

**RQ2.** What is the students' perception towards a practical assignment planned and executed in accordance with the recommendations of GaPSEE?

**RQ3.** What are the main benefits and drawbacks of the adoption of Gamification and PBL principles for software engineering education, in accordance with the recommendations of GaPSEE?

For questions RQ1 and RQ2, the goal was to investigate the lecturers' and students' perception regarding the adequacy of GaPSEE regarding the definition of practical assignments for software engineering education. This evaluation study was not intended to evaluate the documentation of the framework and its usability.

### 6.1.2 Study design and research methods

In order to address the research questions, this study was structured in a process as illustrated in Figure 6.1. Therefore, this study adopts case study method to observe the use of GaPSEE for the planning and execution of practical assignments in software engineering related courses. For data collection, this study relies on a survey study and interviews. For the data analysis, we use descriptive statistics and the Open Coding technique from Ground Theory [Stol et al., 2016].

Runeson et al. [2012] define case study as “an empirical enquiry that draws on multiple sources of evidence to investigate one instance (or a small number of instances) of a contemporary software engineering phenomenon within its real-life context, especially when the boundary between phenomenon and context cannot be clearly specified”.



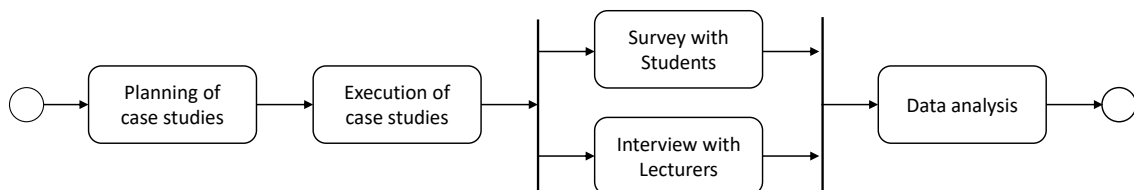
Therefore, our case studies consisted of the use of GaPSEE to setup and execute practical assignments in real educational environments (i.e., software engineering related courses), to investigate the perception of lecturers and students towards this educational intervention. Section 6.2 presents details about the case studies planning and execution.

According to Easterbrook et al. [2008], survey studies are used to identify characteristics of a wide population and are usually associated with the application of questionnaires. Surveys are meant to collect data to describe, compare or explain knowledge, attitudes and behaviors [Easterbrook et al., 2008]. Given the large number of students, we considered the use of questionnaires more appropriate to collect data from the students' perceptions on the case study, and for the analysis using descriptive statistics. The results from this segment of the study are related to RQ2. Section 6.3 presents details about the survey study execution and results.

An interview is a research method defined by a conversation where questions are asked and answers are given [Wohlin et al., 2012]. In this study, we used interviews to collect data about the lecturers perception on the application of GaPSEE. Considering the smaller population of lecturers, interviews are plausible as instruments for data collection. The results from this segment of the study are related to RQ1. Section 6.4 presents details about the interviews execution and results.

## 6.2 Case Studies

This section presents the details of the planning and execution of case studies in order to investigate the application of the recommendations proposed in GaPSEE in real educational environments, without direct intervention of the researcher.



**Figure 6.1.** Study design.



### 6.2.1 Selection of case studies

In order to select appropriate courses for running case studies, we defined a set of criteria:

- **The course must address (direct or indirectly) software engineering topics:** The target course should fit to the scope of GaPSEE, i.e., there should be value in incorporating practical aspects of software engineering in the course.
- **The course must have been offered before by the lecturer:** The lecturer of the course should have already offered the same course before, in order to allow insights from comparison between the case study and previous installments of the course.
- **The course must be offered in the available time frame:** Given the time restrictions of this study, the course should be offered in the first semester of 2019.
- **The lecturer of the course must have availability for meetings:** The lecturer should have availability for periodic meetings with the researcher for the planning of the study and for data gathering during its execution.

Considering the limitations imposed by these criteria, six professors from four federal universities were invited to participate in the study. Four professors, from 3 different institutions (UFLA, UFMG, and UFPA), accepted the invitation. The professors were chosen by convenience, given the necessity of a long period of observation and interaction. In all case studies, the researcher supported the planning of the assignments in accordance with GaPSEE. However, the researcher did not interfere in the execution of the assignments directly, only supported the professors in preparing materials and answering questions in moments of doubt.

Table 6.1 presents the background of the participant professors (identified as L1, L2, L3, and L4, describing their experience (in years) in higher level education, experience (in years) in software engineering education (as lecturer or researcher), experience (in years) in software engineering practice, and experience with specific educational methods (PBL, gamification, game-based learning, or any other mentioned by the lecturers).

Therefore, a pilot study was executed in 2018 (case study PLT), in order to validate the format of the case studies and the procedures for data collection and analysis. In 2019, we conducted four additional case studies, referenced as SQM,



**Table 6.1.** Background of the participants

Participant	L1	L2	L3	L4
Exp. in Teaching	4	10	6	3
Exp. In SE Education	4	10	6	3
Exp. as Practitioner in SE	10	0	5	0
Exp. with Serious games	No	Yes	Yes	No
Exp. with Gamification	Yes	Yes	Yes	No
Exp. with PBL	Yes	No	Yes	No
Exp. with other teaching methods	No	No	Yes	No
Case Studies	PLT, PRO	SQM	SWE	WEB

SWE, PRO, and WEB. Table 6.2 describes the history of each course, regarding the number of times it was offered before by its lecturer, and the previous use of practical assignments, PBL, and Gamification in each course. All case studies were executed in courses that have been offered before at least twice by their lecturers. This was intended to allow the lecturers of each course to notice the differences between the use of the approach proposed in GaPSEE, and their previous teaching experiences in those courses. Four of them had already used some kind of practical assignment. One had already used PBL as educational method. And two had already used gamification in the context of that specific course.

**Table 6.2.** Use of practical assignments, PBL, and Gamification in previous iterations of each course

Case	Course	# Times this course has been offered before	Practical Assignments	PBL	Gamification
PLT	Sw Engineering	4	Yes	Yes	Yes
SQM	Sw Quality and Measurement	5	No	No	No
SWE	Sw Engineering	2	Yes	No	Yes
PRO	Sw Processes	2	Yes	No	No
WEB	Web Development	2	Yes	No	No

The procedures for collecting data consisted of four instruments:

- **Feedback from lecturers:** The lecturers provide the researcher with periodic feedback on the execution of the assignments in each case study. The researcher took notes and interpreted the feedback into information to understand difficulties and patterns in the adoption of GaPSEE.
- **Interviews:** The researcher interviewed lecturers to collect data about their perception on the execution of the assignments in their courses.



- **Surveys:** The researcher applied questionnaires to collect the students' perceptions on the execution of the assignments.
- **Data extraction from tools:** Data from the execution of the gamification was gathered from electronic sheets and communication channels.

## 6.2.2 Preparation and execution of the case studies

An initial meeting with each participant lecturer was executed before the start of their respective courses. In this meeting, the researcher presented GaPSEE to the participant professors, exposing its recommendations and process, with examples. During this meeting, the researcher and lecturers planned the scope of the projects for the assignment of their respective courses, in accordance with expected learning outcomes for the course. The following driving questions and project goals were defined for each course:

**PLT:** *“How to systematically develop a software to meet customer needs?”* - Development of a Web application to support management of practical assignments.

**SQM:** *“How to use software metrics to evaluate the internal quality of a software?”* - Execution of a quality assessment, and the development of an action plan for quality improvement.

**SWE:** *“How to document the specification of a software project for third party development?”* - Development of software project plan, including specification, design and test cases.

**PRO:** *“How to implement configuration management and measurement processes in a software organization?”* - Development of process improvement plan for the processes of configuration management and measurement in accordance with process improvement reference models.

**WEB:** *“How to systematically develop a Web application?”* - Development of a Web application to support management of personal contents.

The subsequent meetings had the purpose to support the lecturers in defining the project structure and gamification strategy. Before the start of the semester, all lecturers had defined a roadmap of activities for their assignments, organized in three iterations, and the gamification approach. For all case studies, the gamification was planned with the following target aesthetics: Challenge, Confidence, Engagement, Motivation, Collaboration, Recognition, Learning, Satisfaction, Relevance, and Fun.

Table 6.3 presents details on the execution of each case study, regarding the number of students enrolled, number of teams, and period of execution. The courses are offered to undergraduate students of Computer Science and Information Systems.



The case study SQM was also attended by four graduate students. A total of 112 students (28 teams) attended these courses. The courses were offered in the first semester of 2019 (except for the pilot case study, PLT, executed in 2018).

The case studies PLT, WEB, and SWE extended throughout the entire duration of their respective courses. The case studies SQM and PRO were limited to specific stages of the courses, in accordance with the topics addressed in the assignment. The remainder of their respective courses were executed using traditional methods. Therefore, the adoption of the framework was customized in accordance to the needs and characteristics of each course.

The lecturers with no previous experience with PBL required additional support of the researcher in the planning of their assignment, specially in designing the roadmap of activities for the projects and in planning their lectures to be aligned with the schedule of the assignment. Therefore, it was noticed that the framework usability differs according to the user background experience.

**Table 6.3.** Organization of the case studies

Case	Course	Programs	# Stds	Teams	Period
PLT	Sw Engineering	Und. Inf. Systems Und. Comp. Science	14	4 teams (3 - 4 stds.)	2018 April - July
SQM	Sw Quality and Measurement	Und. Inf. Systems Und. Comp. Science Grad. Comp. Science	22	6 teams (3 - 6 stds.)	2019 March - May
SWE	Sw Engineering	Und. Inf. Systems	27	5 teams (5 - 6 stds.)	2019 April
PRO	Sw Processes	Und. Inf. Systems Und. Comp. Science	10	5 teams (2 - 3 stds.)	2019 April - May
WEB	Web Development	Und. Inf. Systems Und. Comp. Science	39	8 teams (4 - 5 stds.)	2019 April - June

The assignments initiated with a kick-off presentation, where the lecturers presented the assignment, the goals of their respective projects, the rules of gamification, evaluation and grading procedures.

During the execution of the case studies, online electronic sheets were used to document the progress of the teams for each course. These electronic sheets described the roadmap of activities in the form of quests (mandatory and optional), deadlines, badges and points associated with each task, and the status of each team in respect to the execution of these tasks. The sheet was organized in levels (the iterations of the project), and also presented a leaderboard weekly updated. Figure 6.2 presents one of



**Level 1: Define a quality assessment plan for SimulES**

ID	Task	Description	Type	Difficulty	Deadline	G1	G2	G3	G4	G5	G6
#1	Select the target product for evaluation	Select a SimulES fork, and provide a general description of the product (main features, development technologies, etc)	Mandatory	3 - Bronze	3/22/2019	3 ▾	3 ▾	3 ▾	3 ▾	Fc ▾	3 ▾
#2	Establish criteria for the quality assessment of the product	Define the parameters for the quality assessment. The focus of the assessment must be related to internal quality of the product. Other aspects can be considered, but shall not be the main goal of the assignment.	Mandatory	5 - Silver	3/29/2019	5 ▾	5 ▾	5 ▾	5 ▾	5 ▾	5 ▾
#3	Use norms, reference models or standards to support the definition of quality criteria	Investigate a norm, reference model or standard to support the definition of quality criteria for the assessment.	Optional	5 - Silver	3/29/2019	5 ▾	5 ▾	5 ▾	5 ▾	5 ▾	5 ▾
#4	Plan measurement procedures for the quality assessment	Plan the measurement procedures for the assessment of the selected quality criteria. These procedures must include methods for data collection and analysis, with details on tools and heuristics.	Mandatory	5 - Silver	4/8/2019	5 ▾	5 ▾	5 ▾	5 ▾	5 ▾	5 ▾
#5	Select tools for code smell identification	Select a tool, method or technique for identification of code smells to support the quality assessment.	Optional	5 - Silver	4/8/2019	5 ▾	5 ▾	5 ▾	5 ▾	▾	5 ▾
#6	Compare tools for code smell analysis	Compare at least two tools, methods or techniques for code smell identification to justify your selection.	Optional	7 - Gold	4/8/2019	7 ▾	7 ▾	7 ▾	7 ▾	▾	▾
<b>Total</b>				<b>30</b>		<b>30</b>	<b>30</b>	<b>30</b>	<b>30</b>	<b>15</b>	<b>23</b>

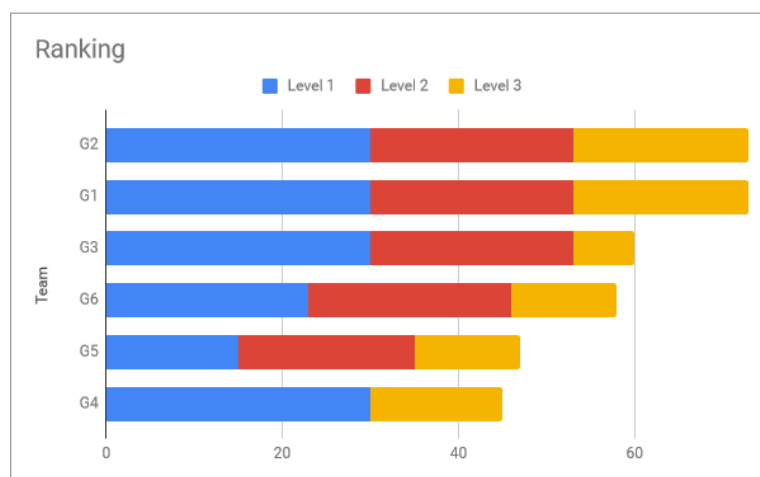
Expected Outcomes	Description
Assessment plan	Document defining the target product, quality criteria, and measurement procedures 4/10/2019
Activity report	Report describing schedule of activities with responsables. 4/10/2019
Presentation	Presentation of the assessment plan 4/10/2019

**Figure 6.2.** A level from the SQM case study

the levels of the assignment from case SQM, represented in the electronic sheet, and Figure 6.3 illustrates the leaderboard of the course.

Rank	Team	Level 1	Level 2	Level 3	Score	%*
1	G2	30	23	20	73	100.00%
1	G1	30	23	20	73	100.00%
3	G3	30	23	7	60	82.19%
4	G6	23	23	12	58	79.45%
5	G5	15	20	12	47	64.38%
6	G4	30	0	15	45	61.64%

\* % = Score / Max Score  
Max Score= 73

**Figure 6.3.** The leaderboard from SQM case study

Besides the initial effort to setup the assignments, the case studies were conducted



without intervention of the researcher. Periodical meetings were held to discuss the progress of the assignments and to identify needs for changes. However, no change was necessary in respect to the execution of the assignments.

By the end of the assignments, each lecturer presented the winning teams, and awarded them with the predefined rewards (for instance, the lecturers of the SWE, SQM, and PRO, rewarded the winning teams with pizza). At this moment, the students were invited to participate in a survey study (Section 6.3) voluntarily, with no impact on grades. After the conclusion of the assignments, the lecturers were interviewed by the researcher (Section 6.4).

## 6.3 Survey with students

This section describes the execution of a survey study to collect and analyze data on the students perceptions about the resulting educational intervention applied to their respective courses using GaPSEE.

A questionnaire was created in Google forms <sup>1</sup> to collect data from participants. The questionnaire was structured in five sections: (i) participants background; (ii) importance and adequacy of the practical assignment to the software engineering courses; (iii) gamification goals; (iv) development of skills; and (v) positive and negative aspects of the assignment. The first version of this questionnaire was created with the support of the lecturers involved in the case studies, and was piloted with three students to understand if its items were clear and if they addressed the research question RQ1 of this study. The final version of the questionnaire is presented in Appendix B.

### 6.3.1 Population Sample

Table 6.4 describes the population and sample size for each case study and the totals for the study. The study was executed in the context of case studies “SQM”, “SWE”, “PRO”, and “WEB”. We did not apply this survey study in the pilot case study (“PLT”). Therefore, the total population for this study was 98 students, from which 76 (77.6%) students voluntarily participated in the study. The sample size of the case studies SQM, SWE, PRO, and WEB corresponds to 21.1%, 31.5%, 13.2%, and 34.2% of the total sample, respectively.

---

<sup>1</sup><http://www.google.com/forms>



**Table 6.4.** Population sample for the survey study.

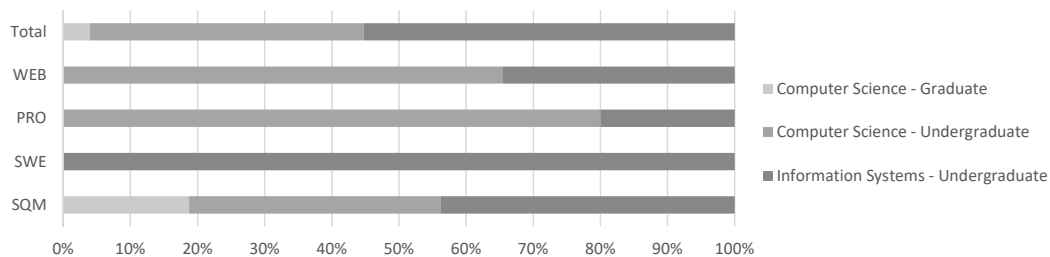
Course	Population	Sample	%
SQM	22	16	72,7%
SWE	27	24	88,9%
PRO	10	10	100,0%
WEB	39	26	66,7%
Total	98	76	77,6%

## 6.3.2 Results

This subsection describes the results of the survey. The following five subsections describe the results for each section of the survey questionnaire.

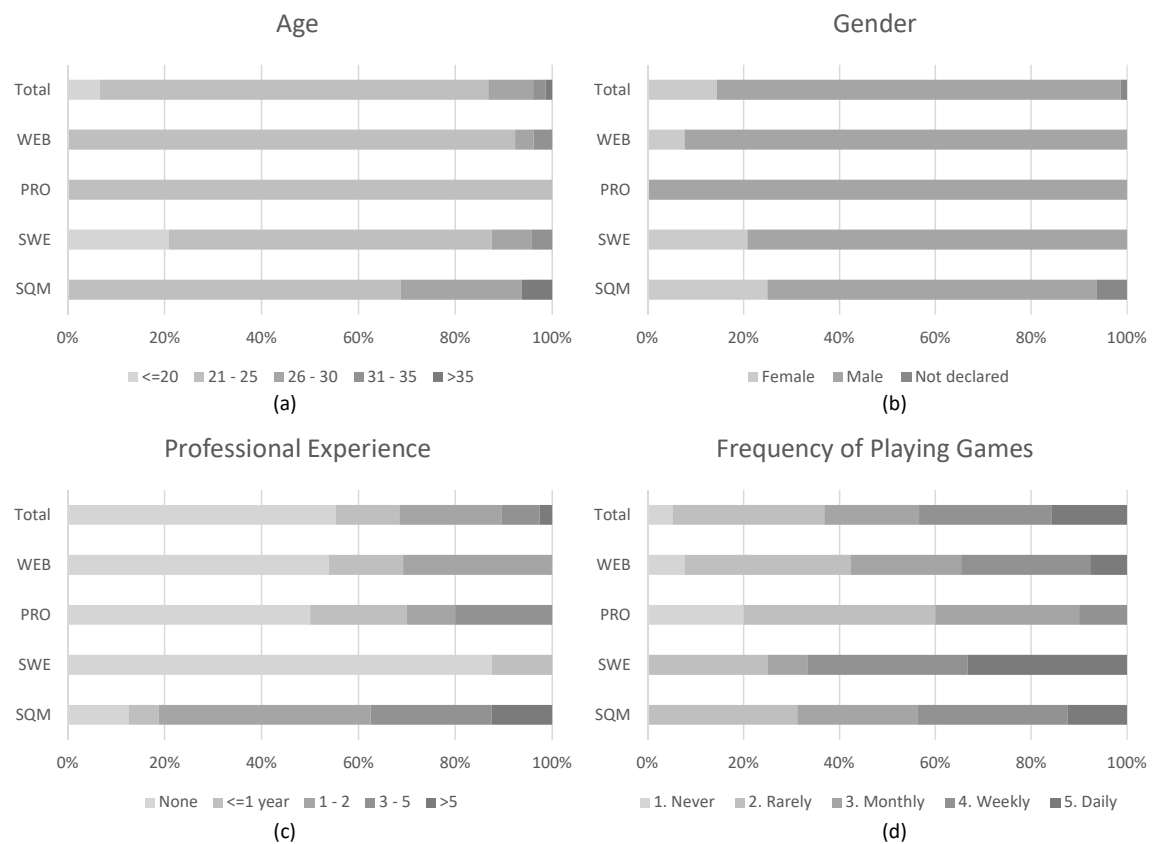
### 6.3.2.1 Participants background

The first section of the questionnaire was composed of questions related to the background of participants: superior education program attended, age, gender, professional experience, and frequency of playing games. The sample population consists of graduate and undergraduate students from Computer Science and Information Systems programs (as described in Figure 6.4). Figure 6.5 presents the participants' (a) age, (b) gender, (c) professional experience (in years), and (d) frequency of playing games. Most (80%) of the participants are between 21 to 25 years old, and most participants are male (84%). Most of the participants (55%) had no professional experience in software engineering or software development, while 26 participants (34%) have up to 2 years of professional experience, and only 8 participants have more than 3 years of experience. Regarding the frequency in which the participants play games, only 5 participants declared never playing games, 39 play in a monthly basis, and 33 declared to play at least in a weekly basis.

**Figure 6.4.** Higher education program attended by the participants (N=76).

Therefore, the typical profile of the participants is an undergraduate student, male, 21 to 25 years old, with no professional experience and with the habit of playing games in a somewhat regular basis (between rarely and weekly). This profile corresponds to 23 participants (30.3%) of our sample.





**Figure 6.5.** Participants' (a) age, (b) gender, (c) professional experience (in years), and (d) frequency of playing games (N=76).

### 6.3.2.2 Importance of practice and adequacy of the assignment to software engineering education.

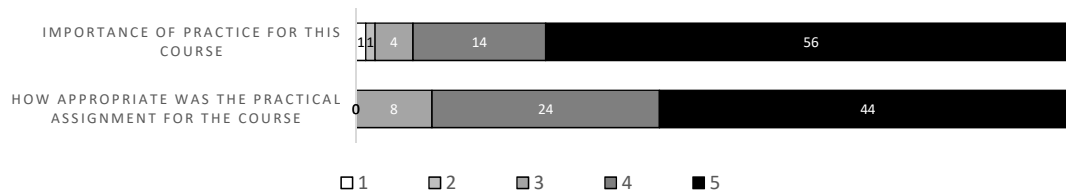
The second section of the questionnaire is composed of two items, asking the participants to rate, in a scale of “1” (totally irrelevant) to “5” (fundamental), (i) the importance of practice to the course they attended, and (ii) the adequacy of the practical assignment to the context of the course. Figure 6.6 presents the results for these items.

For the first (i), 56 participants (73.7%) assigned the highest rate (5) for this item, and only 2 participants (2.6%), from the WEB and SQM case studies, assigned the the lowest values (“1” or “2”) for this item. Therefore, there is a clear indication that students perceive “practice” as a very important aspect of the software engineering related courses.

For the second (ii), 44 participants (57.9%) rated the adequacy of the assignment as “5” (completely appropriate), and no participant assigned the values “1” or “2” for



this item. Therefore, there is evidence that the practical assignment (in accordance with GaPSEE) was perceived as adequate for the courses.



**Figure 6.6.** Importance of practice for software engineering education and adequacy of the assignment for the courses.

### 6.3.2.3 Gamification goals

In the third section of the questionnaire, participants were asked to rate several aspects related to the expected goals of the gamification (the aesthetics, in the MDA framework). Therefore, the participants had to evaluate to which extent the gamification approach instilled feelings related to: challenge, confidence, engagement, motivation, collaboration, recognition, learning, satisfaction, relevance, and fun. All items were statements that the participants had to evaluate using Likert scale values of “-2” (completely disagree) to “2” (completely agree). This section of the framework was inspired in the MEEGA+ questionnaire [Petri et al., 2017].

Figure 6.7 presents the results for each item, organized by goal. For all items, there is a majority of positive answers (“Partially agree” or “Totally agree”). However, some items related to motivation, recognition and fun received lower ratings. Specifically, these items are related to: motivation as a consequence of rewards and being the best teams in the course; considering a personal recognition factor to be in one of the best teams; and all questions related to fun. On the other hand, some items related to challenge (Q1), confidence (Q3, Q4), engagement (Q6), learning (Q15, Q16, Q17, Q19), and relevance (Q24, Q25) received the highest ratings.

Figure 6.8 summarizes these results in box-plot for better comparison. Except for “challenge”, all aspects had more than one related questionnaire item. Therefore, the box-plot graph uses the average value of the items for each aspect. All results were positive. However, “Fun” had the lowest overall ratings of all aspects, while “Challenge”, “Confidence”, “Engagement”, “Learning”, and “Relevance” were better rated.



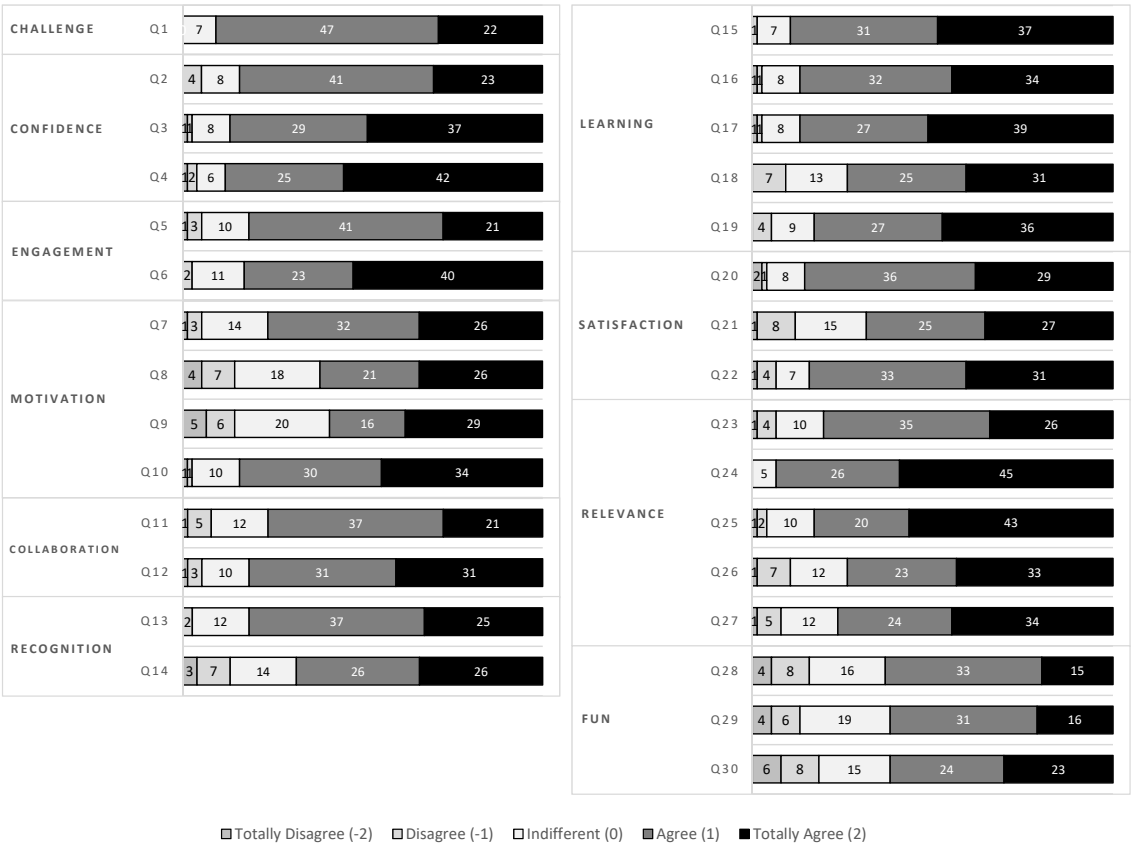


Figure 6.7. Results of the Survey for the Aesthetics of the gamification approach.

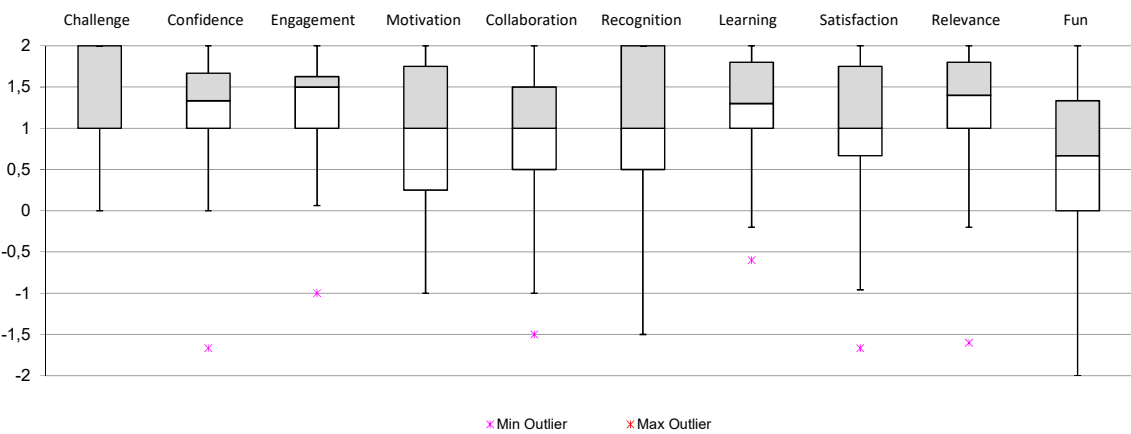
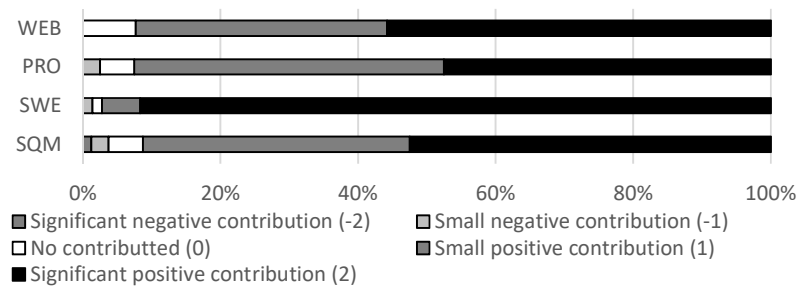


Figure 6.8. Participants' perception on aesthetic aspects of the gamification approach.



### 6.3.2.4 Development of skills

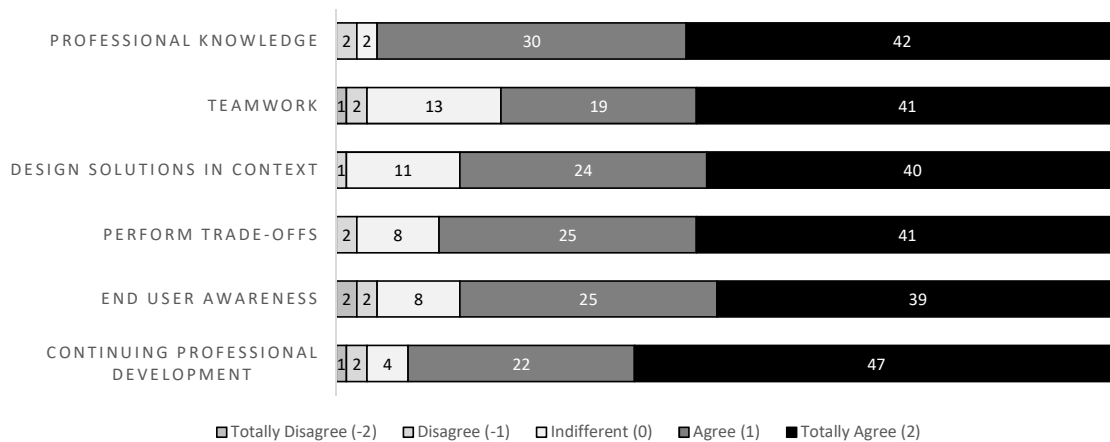
In the fourth section of the questionnaire, the participants were asked to rate how much the assignment supported the development of general and specific skills. Each course had a specific set of expected skills to be developed during the assignment. Participants were asked to evaluate the contribution of the assignment for the development of these skills in a nominal scale ranging from “significant negative contribution” (-2) to “Significant positive contribution” (2). Figure 6.9 presents the distribution of the total number of each rating for the responses of participants of each case study. For all case studies, the participants perceived a positive contribution of the assignment for the development of the specific skills related to each course. The SWE case study presented the highest number of ratings indicating “Significant positive contribution” (2).



**Figure 6.9.** Participants’ perception on development of specific skills.

Similarly, participants were also asked to rate the support of development of general skill using the same rate. All courses shared the same expected general skills (as described in Chapter 5): professional knowledge, teamwork, ability to design of solutions in context, ability to perform trade-offs, end user awareness, and continuing professional development. Figure 6.10 presents the overall distribution of responses for each general skill. All items received majority of positive ratings (over 78.9%). The “professional knowledge” and “continuing professional development” received larger number of positive responses (“1” or “2” ratings), and even the larger number of “significant positive contribution” (2) ratings. “Teamwork” received the larger number of neutral responses (13). The SQM and WEB study cases gathered the larger number of neutral and negative responses, while the SWE case study had a single occurrence of a neutral (0) response, and the “PRO” case study had two neutral responses (0) and a single “small negative contribution” (-1) response.





**Figure 6.10.** Participants' perception on development of general skills.

### 6.3.2.5 Positive and negative aspects of the assignment

The fifth section of the questionnaire provided three open questions for the participants regarding their thoughts on positive and negative aspects of the assignment, and additional comments with critics or suggestions for the assignment. The first and second questions were mandatory, only the latter was optional. To analyze the answers, we used an approach inspired by the coding phase of Ground Theory [Stol et al., 2016]. The researcher analyzed the responses in iterations and marked relevant segments with codes (tagging with keywords). The analysis was finished after a number of coding iterations, when no new codes were found (saturation). Then, these codes were grouped in categories. Consequently, it is possible to count the number of occurrences of codes and the number of items in each category to understand what recurring positive and negative aspects were pointed by the participants, and propose possible lessons learned.

Tables 6.5 and 6.6 presents the results from the open coding phase of the analysis of positive and negative aspects pointed by participants. These tables describe the codes, the number of occurrences of the codes for each case study and the total number of occurrences, and define a category for each code.

We mapped a total of 100 codes, where 55 codes are related to positive aspects (positive codes, henceforth) and 45 codes are related to negative aspects (negative codes, henceforth). These codes occurred 156 and 72 times, respectively, in the responses of the participants. It is important to notice that each response can be associated with more than one code. Figure 6.11 presents the distribution of the occurrences of positive and negative codes. It is possible to notice that 61 (39.1%) occurrences of positive codes are concentrated in the SWE case study.



**Table 6.5.** Positive codes and categories mapped from the responses of the participants.

Code	PRO	SQM	SWE	WEB	Total	Category
Learning and using new tools and technologies	0	2	1	8	11	Learning Improvements
Escape from traditional methods	1	1	2	4	8	Learn./Teach. Approach
Application of theory in practice	3	4	1	0	8	Learn./Teach. Approach
Competition	1	1	5	1	8	Gamification
Increased engagement	1	1	5	0	7	Learning Improvements
Roadmap of activities	1	0	2	3	6	PBL / Project Structure
lecturer support	0	3	3	0	6	Learn./Teach. Approach
Practical approach	2	0	0	4	6	Learn./Teach. Approach
Increased motivation	2	0	4	0	6	Learning Improvements
Feedback	1	3	1	1	6	Gamification
Simulation of real scenario	1	2	0	2	5	PBL / Project Structure
Gamification	0	1	4	0	5	Gamification
Teamwork	0	1	2	1	4	Teamwork
Dynamic	0	0	3	0	3	Qualities
Interesting	0	0	2	1	3	Qualities
Partial deliveries	0	1	0	2	3	PBL / Project Structure
Applied learning/teaching	0	1	2	0	3	Learn./Teach. Approach
More engaged lecturer	0	2	1	0	3	Learn./Teach. Approach
Knowledge acquisition	0	0	3	0	3	Learning Improvements
Rewards	1	0	2	0	3	Gamification
Diverse team	0	0	0	2	2	Teamwork
Quality: Satisfactory	0	1	1	0	2	Qualities
Flexibility and freedom	0	0	0	2	2	PBL / Project Structure
PBL	0	2	0	0	2	PBL / Project Structure
Work in a entire project	0	0	1	1	2	PBL / Project Structure
Educational method	0	0	2	0	2	Learn./Teach. Approach
Promotes immersion	2	0	0	0	2	Learn./Teach. Approach
Quality of content and instructions	0	0	2	0	2	Learn./Teach. Approach
Search for appropriate solutions/knowledge	1	0	1	0	2	Learn./Teach. Approach
Arouse interest in continuous learning	0	0	0	2	2	Learning Improvements
Knowledge sharing between students	0	0	0	2	2	Learning Improvements
Learning	0	0	1	1	2	Learning Improvements
Professional knowledge	0	0	2	0	2	Learning Improvements
Ranking	0	1	1	0	2	Gamification
Did not cause overhead (time) in parallel with other courses	0	1	0	0	1	Time and Effort
Improved performance of teams	0	0	1	0	1	Teamwork
Beneficial	0	0	1	0	1	Qualities
Enriching	0	0	0	1	1	Qualities
Fun	0	0	1	0	1	Qualities
Not boring	1	0	0	0	1	Qualities
Organized structure	0	1	0	0	1	Qualities
Positive	0	0	1	0	1	Qualities
Development of a tangible product	0	0	0	1	1	PBL / Project Structure
Practical project	0	0	0	1	1	PBL / Project Structure
Combine theory and practice	0	1	0	0	1	Learn./Teach. Approach
Explores individual qualities	0	0	0	1	1	Learn./Teach. Approach
Good for beginners	0	0	0	1	1	Learn./Teach. Approach
Interaction between student and lecturer	0	0	1	0	1	Learn./Teach. Approach
Learn by doing	0	0	1	0	1	Learn./Teach. Approach
Participative course	0	0	1	0	1	Learn./Teach. Approach
Theoretical foundation	0	0	0	1	1	Learn./Teach. Approach
Better comprehension	0	1	0	0	1	Learning Improvements
Exposes students to the importance of practice	0	0	0	1	1	Learning Improvements
Knowledge retention	0	1	0	0	1	Learning Improvements
Skill development	0	0	0	1	1	Learning Improvements
Totals	18	32	61	45	156	



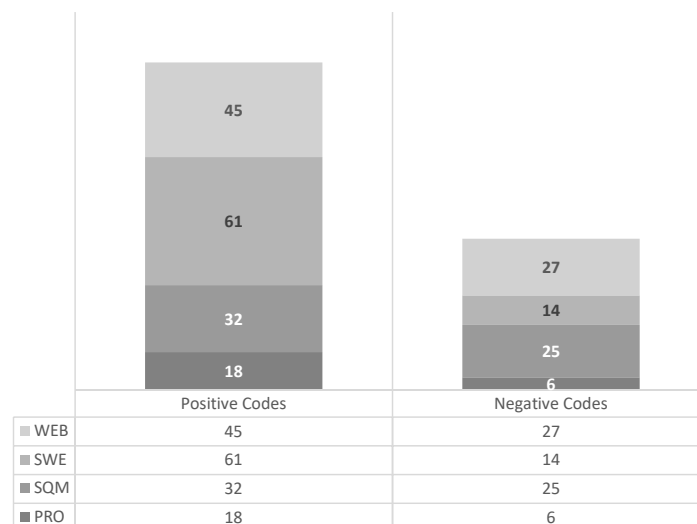
**Table 6.6.** Negative codes and categories mapped from the responses of the participants.

Code	PRO	SQM	SWE	WEB	Total	Category
Unclear information or instructions	0	4	1	3	8	PBL/Project Structure
Time	0	2	2	3	7	Time and Effort
Lack of introductory theory	1	0	0	4	5	Learn./Teach. Approach
Lack of freedom to choose their own projects	0	2	0	2	4	PBL/Project Structure
Teamwork	0	2	2	0	4	Teamwork
Lack of details or difficulty in a particular topic	0	0	2	1	3	Learn./Teach. Approach
Selected tools or technologies	0	2	0	1	3	PBL/Project Structure
Structure of presentation	0	1	0	1	2	Other
Unbalanced effort or engagement from teammates	0	1	1	0	2	Teamwork
Frequency of deliveries	1	1	0	0	2	Time and Effort
Number of tasks/deliveries	0	2	0	0	2	Time and Effort
Lack of bonus grades	0	1	0	0	1	Evaluation and Grading
Unclear details about grading	0	0	0	1	1	Evaluation and Grading
Competition	0	1	0	0	1	Gamification
Boring /tiresome lectures that did not contribute for the activities	0	0	0	1	1	Learn./Teach. Approach
Did not cover the course syllabus	0	0	0	1	1	Learn./Teach. Approach
Restricted	0	0	0	1	1	Learn./Teach. Approach
Focus on aspects that should be secondary	0	0	0	1	1	Learn./Teach. Approach
Hard for participants without practical experience	1	0	0	0	1	Learn./Teach. Approach
Lack of alignment between assignment and the course	0	1	0	0	1	Learn./Teach. Approach
Prefer smaller individual tasks	0	0	0	1	1	Learn./Teach. Approach
Reduced learning	0	0	0	1	1	Learn./Teach. Approach
Students felt lost	0	1	0	0	1	Learn./Teach. Approach
The approach to cover the topics in classroom	0	0	0	1	1	Learn./Teach. Approach
Infrastructure	0	0	0	1	1	Other
Lack of incentives for collaboration	0	1	0	0	1	Other
Lack of personal approach	0	0	1	0	1	Other
Delivery of work products	0	1	0	0	1	PBL/Project Structure
Difficulty in handling tools and technologies	0	0	1	0	1	PBL/Project Structure
Lack of templates	0	1	0	0	1	PBL/Project Structure
Loose structure	0	0	0	1	1	PBL/Project Structure
Unbalanced projects	0	1	0	0	1	PBL/Project Structure
Unclear goals	0	0	0	1	1	PBL/Project Structure
Individualism	0	0	1	0	1	Teamwork
Motivation in teamwork is unfair	0	0	1	0	1	Teamwork
Team composition	0	0	1	0	1	Teamwork
Team size	0	0	1	0	1	Teamwork
Additional extra-class effort	1	0	0	0	1	Time and Effort
Constant effort may overlap with other courses and hampers performance	0	0	0	1	1	Time and Effort
Number of tasks with same deadline	1	0	0	0	1	Time and Effort
Should use less classroom time	1	0	0	0	1	Time and Effort
Totals	6	25	14	27	72	



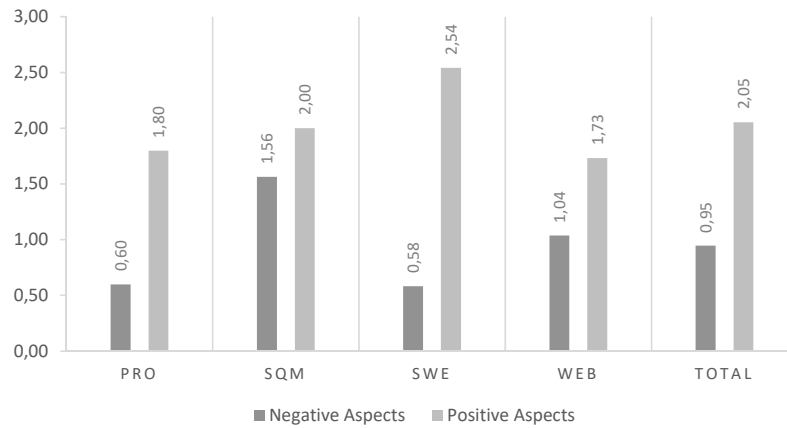
However, considering the different sample size for each case study, these totals are not appropriate for comparison. To understand the density of positive and negative codes in each case study, we calculated the ratio of the number of occurrences of positive and negative codes per participants (positive code ratio and negative code ratio, for reference). Figure 6.12 presents these ratios for the entire sample and for each case study. For the entire sample, the negative code ratio is 0.95 occurrences per participant and the positive code ratio is 2.05 occurrences per participant. For negative codes, the lowest ratios are related to the SWE and PRO case studies (0.58 and 0.6, respectively.), and the highest ratios are related to the SQM and WEB case studies (1.56 and 1.04, respectively). For positive codes, the highest ratios are related to the SWE and SQM case studies (2.54 and 2.0, respectively.), and the lowest ratios are related to the PRO and WEB case studies (1.80 and 1.73, respectively). For all case studies, the ratio of positive codes is higher than the ratio of negative codes. Therefore, the case study SWE has the best overall results, regarding the students perception.

Regarding the positive codes, all participants stated at least one positive aspect about the assignment. There were 156 occurrences of these codes in the responses (each answer could present more than one code). The top five positive codes (in number of occurrences) are: “Learning and using new tools and technologies”, “Escape from traditional methods”, “Application of theory in practice”, “competition”, and “Increased



**Figure 6.11.** Distribution of occurrences of positive and negative codes for the case studies.





**Figure 6.12.** Ratio of Codes per Participants for positive and negative aspects (# occurrence of positive or negative codes) / (size of the sample).

engagement”. In relation to the negative codes, 53 participants (69.7%) indicated negative aspects. Even though this item was mandatory in the questionnaire, 21 participants (27.6%) stated they did not have any negative points to indicate about the practical assignment. There were 72 occurrences of these codes in the responses (each answer could present more than one code). The top five negative codes (in number of occurrences) are: “Unclear information or instructions”, “Time”, “Lack of introductory theory”, “Lack of freedom to choose their own projects”, and “Teamwork”.

The negative and positive codes were grouped in 9 categories: “PBL and project structure”, “Learning and Teaching Approach”, “Time and Effort”, “Teamwork”, “Evaluation and Grading”, “Gamification”, “Learning Improvements”, “Qualities” and “Others”. Table 6.7 summarizes the categories of codes and the respective number of occurrences for positive and negative codes.

**Table 6.7.** Occurrences of positive and negative codes for each category.

Category	Pos. codes	Neg. codes	Total
Learn./Teach. Approach	49	16	65
PBL/Project Structure	22	21	43
Learning Improvements	39	1	40
Gamification	24	1	25
Teamwork	7	10	17
Time and Effort	1	15	16
Qualities	14	1	15
Other	0	5	5
Evaluation and Grading	0	2	2
Total	156	72	228

The categories with higher occurrences of positive codes are: “Learning and Teaching Approach” (49 occurrences), “Learning Improvements” (39 occurrences), “Gamification” (24 occurrences), and “PBL and Project structure” (22 occurrences).



This is an evidence of a positive attitude of the students towards the format of the assignment, and the perception of positive impacts of this assignment to their learning process. In addition, elements of PBL and gamification were also consistently mentioned as positive aspects of the assignment format. Gamification, in special, was almost exclusively seen as a positive aspect, with only one participant stating that competition is not an appropriate strategy to motivate students. For PBL, however, there were mixed opinions (22 versus 21 occurrences of positive and negative codes). Positive codes in this category include the existence of a roadmap of activities to guide the execution of the project, fidelity to the professional context of software development, and working in iterations with partial deliveries. Negative codes includes the clarity of information or instructions, the desire of students to choose their own projects, and disagreement on the tools and technologies used in the project.

The categories with higher occurrence of negative codes are: “PBL and Project structure” (21 occurrences), “Learning and Teaching Approach” (16 occurrences), “Time and Effort” (15 occurrences), and “Teamwork” (10 occurrences). Although there were only 16 occurrences of negative codes out of the 65 occurrences of all codes in “Learning and Teaching Approach”, negative aspects in this category include lack of introductory theory, insufficient coverage of the course syllabus, and difficulties or lack of details in specific topics. The category “Time and Effort” groups complaints about the time required to execute the project, the number of tasks, and the frequency of deliveries. For “Teamwork”, the occurrence of negative codes was slightly superior to positive codes, indicating problems about unbalanced engagement of team members and individualism.

In the “Qualities” categories, we mapped statements using adjectives to describe the assignment as fun, interesting, dynamic, beneficial, enriching, positive, not boring, and organized. In only one code the assignment is described with a negative quality: “restricted” (as in slow, as opposed to dynamic).

The “Evaluation and Grading” category groups two comments about the lack of bonus grades to reward the execution of secondary tasks in the projects, and the lack of clear information about the grading system. The “Other” category groups additional complaints about the structure of presentations, infrastructure in the laboratory for the execution of practical activities, and lack of a personal approach in the evaluation of activities.

These results in general are evidences of a positive perception of the students towards the GaPSEE approach, as it was implemented in each course.



## 6.4 Interviews with lecturers

We conducted semi-structured interviews to capture the perception of the four lecturers in respect to the application of GaPSEE in their courses. The script for the interview was first piloted in the pilot case study (PLT), and its final configuration is presented in Table 6.8. It is divided in four sections: the first includes questions about the experience background of the interviewee; the second includes questions about the interviewee's experience with educational methods such as game-based learning, gamification, and PBL; the third section includes questions about the previous installments of the course under investigation; the fourth section includes questions about the case study. The responses for questions 1.1 to 3.1 were already addressed in Section 6.2. Therefore, this section focuses on the analysis of the answers of the questions 3.2 to 4.10.

**Table 6.8.** Script for interviewing lecturers

---

<b>1. Background</b>
1.1. What is your experience background as a Professor? (years)
1.2. What is your experience background in Software Engineering education? (lecturing or research)
1.3. What is your experience background in Software Engineering as a practitioner?
1.4. Which Software Engineering related courses have you teach?
 <b>2. Use of teach/learning methods</b>
2.1. How often do you use practical assignments in your courses? What are your thoughts about it?
2.2. Have you ever used Project-Based Learning in your courses? What are your thoughts about it?
2.3. Have you ever used games in your courses? What are your thoughts about it?
2.4. Have you ever used gamification in you courses? What are your thoughts about it?
2.5. Have ever used any other educational methods not mentioned?
 <b>3. About the course</b>
3.1. How many times have you offered this course before?
3.2. What is the relevance of practice in this course?
3.3. What have you done to promote practice in previous installments of this course?
3.4. What are your thoughts on students motivation in previous installments of this course?
 <b>4. About the PBL/Gamification approach used in the case study</b>
4.1. Did you perceive any difference in students reaction to the course activities in relation to previous installments of the course?
4.2. Did you perceive any difference in the management of the course/assignment? 4.3. Did you find relevant the use of such approach (PBL/Gamification)? Why?
4.4. Do you believe this approach is useful for introducing practice in classroom?
4.5. What are the positive aspects of this approach?
4.6. What are the negative aspects of this approach?
4.7. What would you have done differently?
4.8. Do you intend to use this approach again (with improvements, if needed) in future installments of this course?
4.9. Do you intend to use this approach again (with improvements, if needed) in other courses?
4.10. Would you suggest using this approach to other lecturers?
4.11. Do you believe it would be relevant to have guidelines or instructions to support the use of this approach?

---



### 6.4.1 Previous installments of the courses

In question 3.2, the interviewees were asked about the relevance of practice for their respective courses. All four lecturers agreed on the importance of practice.

[SWE] *“Essential to allow students to connect the concepts from the diverse specific areas from software engineering”.*

[SQM] *“Important, however I always find challenging to propose a practical assignment in this course”.*

[PRO] *“It is essential. Not necessarily in the format of a project. But it is important because if students are provided only with theory, they will only memorize the processes, without knowing how they really work in real contexts”.*

Regarding the previous attempts to introduce practice in the courses (item 3.3), in previous instances of the courses related to cases SWE, PRO, and WEB the lecturers used sporadic exercises and isolated, smaller practical assignments to address specific topics. For the SQM case, the lecturer indicated that the proposal of a relevant practical assignment has always been a challenge to him. Therefore previous installments only used exercises, serious games and, sometimes, experiences with tools, to introduce practice to the SQM course.

Regarding the attitude of students in previous installments of the courses (item 3.4), the interviewees revealed different perspectives. For the lecturer of the SQM case, the students of his institution usually show high engagement and motivation. However the participation of the students in this course is usually low. For the lecturer of the SWE course, the motivation of the students was intimately related to their affinity with the topics of the course. As a consequence, students with no affinity for the topics addressed in the course had lower motivation and engagement, and the lecturer had no other resource to raise their engagement and motivation rather than grades. The lecturer of the WEB case mentioned that students were generally less engaged, and that they were usually highly motivated in the first two weeks of course, until they were faced with difficulties and challenges. Additionally, this lecturer also mentioned that the participation of the students was limited to some students with prior professional experience, which shared experiences, while other students often interrupted the lectures to raise questions about topics unrelated to the course (usually associated with difficulties in programming and object-oriented paradigm).



### 6.4.2 Changes in students' attitudes

Regarding the difference in the attitude of students (item 4.1), the interviewees mentioned improvements in: participation, engagement, confidence, collaboration,

The participants were unanimous regarding improvements in the participation of the students in comparison to previous instances of the courses. Both in terms of frequency and quality. Not only students interacted more frequently with lecturers, raising questions, suggesting ideas, and seeking validation, but also these interactions were more focused to the project and related topics.

[PLT] *“There was a higher interest of the students because it was a real project. Furthermore, there was increased participation of students in suggestions of new ideas to improve the product”.*

[PRO] *“Students also asked more questions related to the project”.*

[SQM] *“There was little improvement in engagement. However, there was a clear difference in the higher interaction between students and lecturers (professor and teaching assistant), because of the practical assignment and gamification (competition, roadmap of activities and the students' desire to obtain feedback)”.*

[SWE] *“The students started to ask more questions, because they wanted to deliver work products adequate to the tasks, by the end of each level. Therefore, they made lots of validations and reviews of these work products with me, before delivering. (...) in each theoretical lecture, the students asked questions related to the practical project”.*

[WEB] *“The students were more participative in the course, using the ‘campus virtual’ [system used to support courses at the university] to raise questions. (...) The questions from students were better, more focused on the context of the project, and not arbitrary in respect to topics unrelated to the course”.*

Interviewees described an increased engagement of the students in the realization of the activities and really trying to understand the topics from the course, in order to propose appropriate solutions and achieve better results in their projects.

[PRO] *“They tried to understand the problem and propose an appropriate solution. The students really sought to understand how the processes worked”.*



[SWE] *“As a lecturer, I was always doing this abstraction of how to associate the theory to the activity that was being executed [in the assignment], and getting them [the students] interested by the topic in order to understand how it would help them in the execution of the task. (...) As a consequence, I perceived increased interest in the execution of these tasks. It was pretty much caused by the competition, the leaderboards, the desire for the rewards of each level, and the desire to achieve the first place in the course”.*

[PLT] *“The students seemed more competitive, trying to do their best. I don’t know if it happened because there was a reward, or because they were really striving to develop the best project. (...) They were more engaged in the development of the project. (...) There was much larger utilization of Git and the real interest in learning about its use, in comparison to previous installments of the course”.*

[SQM] *“The approach provided incentives for the student to be engaged, to strive to achieve their best results”.*

[WEB] *“I perceived that when you provide a challenge, they [the students] appear more engaged, there is a competition among them. Previously, it was like they were ‘simply doing yet another exercise’ ”.*

Participants described the impact of feedback on students confidence in the execution of the assignments, specially as a consequence of being given a chance to fail and improve before the proper evaluation.

[SWE] *“Students reported that they learned a lot because they were not charged with the deliveries only in the final deadlines. Instead, they were motivated to execute activities with antecedence, which allowed them to refine their work products, delivering results closer to what was expected. This leads to increased students’ satisfaction, because no one likes to be evaluated without the possibility to improve. This immediate feedback cycle makes students more satisfied with the possibility to improve throughout the process”.*

[PRO] *“After the first iteration they get used to the routine of delivering before the deadlines in order to get feedback and improve their work products before the final submission. (...) Feedback makes the students more confident in executing the activities, as they have a chance to fail and improve”.*



[PLT] *“Some students, who were insecure about their capacity to execute the project, used the roadmap of smaller activities and feedback to make progress in the project. There was a particular case in which a team was insecure during the assignment, and yet achieved the best score”.*

The lecturers of cases PRO and WEB mentioned improvements in the interaction or collaboration between students.

[PRO] *“I believe the students discussed more with their teammates, allowing the sharing of knowledge and experiences”.*

[WEB] *“In the current installment, these students [with prior professional experience] acted as leaders of their teams, sharing their knowledge not only with me, but also with their teammates, and students without professional experience could assimilate how a company really works. (...) and they discussed more within their teams”.*

The lecturer of the case PLT also mentioned better results, better understanding of the importance of each activity, and increased volume of partial results. Finally, the lecturers of the WEB and PRO cases mentioned that the students were more proactive and independent.

### 6.4.3 Changes in the management and preparation of the assignment

Participants stated that there is an initial effort to plan the entire project prior of the start of the assignment, however they were unanimous that the approach did not cause additional effort that would make the approach unfeasible.

The lecturer of the SWE case discussed that his adoption of PBL for the whole course made him think of his course as a project, having to propose meaningful activities to address each topic of the course syllabus. This led him to a deep reflection on the course, specially on the development of skills and effective learning. The lecturer of the WEB case mentioned that this approach made it easier to manage the course. As the format made the students more independent and proactive.

The lecturer of the PLT case discussed the positive impacts of having a single project for all teams. It allowed homogeneous communication with the students, making it easier to use the project as example to ground the lectures, easier to provide feedback, and allowing the use of the project in supervised practical activities in classroom time.



#### 6.4.4 Relevance and positive aspects of GaPSEE approach

All interviewees were unanimous in respect to the relevancy of the approach defined by GaPSEE. Their answers for item 4.3 are transcribed as follows.

[WEB] *“Yes, because the students were more motivated and we [educators] have to try new approaches to make students more active and more interested. Also, because students don’t want this traditional format anymore. However, I believe it [GaPSEE] is more applicable to some disciplines”.*

[PLT] *“Yes. I believe that PBL motivates the students in doing something real, meaningful, and gamification supports the motivation. There is an increased motivation, I don’t know if it caused by the competition, or by the reward, or by the fact that students earn badges. It makes them feel more capable”.*

[SWE] *“Yes. PBL helps in the definition of crosscutting activities for the course. The gamification supports motivation and engagement, which was a big challenge I had: motivating students without affinity with the topics addressed in the course. I believe these methods are complimentary. PBL made me analyze the topics and transform them into tasks that allow students to develop competences. The gamification brings motivation, engagement, competition and development of good rapport between students, which is well fitted for the context of a project”.*

[SQM] *“Yes. Specially because of the gamification”.*

[PRO] *“Yes. The project allowed better contextualization of the topics in a meaningful way, grounded in a more realistic problem. The gamification motivated the students towards a better position in the leaderboards, trying to earn more points. The approach was also interesting for promoting teamwork. I believe the students discussed more with their teammates, allowing the sharing of knowledge and experiences”.*

Table 6.9 summarizes the positive aspects of the approach pointed by the participants. We used the open coding technique to code the positive aspects from the responses of interviewees, in order to provide categorization and quantification of these positive aspects. Table 6.9 provides the codes, the number of their occurrence in each interview, the total number of their occurrences in all interviews, and their categories.



**Table 6.9.** Codes and categories related to positive aspects mapped from the interviews.

Codes	WEB	SWE	SQM	PRO	PLT	Total	NDI*	Category
(+) Engagement and interest	1	7	0	1	4	13	4	Impr. students' attitude
(+) Motivation	3	3	0	2	4	12	4	Impr. students' attitude
(+) Participation and interaction with lecturers	6	2	2	1	1	12	5	Impr. students' attitude
Element: Roadmap of activities	2	1	1	2	4	10	5	PBL
(+) Meaningful contextualization of practice	0	2	1	2	3	8	4	Benefits for lecturers
Element: Competition	1	2	2	0	2	7	4	Gamification
Element: Gamification	0	2	1	2	2	7	4	Gamification
(+) Independence and proactivity	4	0	0	1	0	5	2	Impr. students' attitude
Element: Project	2	0	0	0	3	5	2	PBL
(+) Teamwork, collaboration and knowledge sharing	2	1	0	1	0	4	3	Impr. students' attitude
Element: Feedback	0	0	1	1	2	4	3	Gamification
Element: PBL	0	2	0	1	1	4	3	PBL
(+) Better results	1	1	0	0	1	3	3	Impr. students' attitude
(+) Confidence	0	0	0	1	2	3	2	Impr. students' attitude
(+) Lecturer: Reflection of skill development	0	3	0	0	0	3	1	Benefits for lecturers
(+) Exposes students to tools and practical aspects not addressed in the course	0	0	1	0	1	2	2	Impr. students' attitude
(+) Systematic development of activities	0	0	0	0	2	2	1	Impr. students' attitude
Element: Leaderboards	1	1	0	0	0	2	2	Gamification
Element: Rewards	0	1	0	0	1	2	2	Gamification
Element: Teamwork	1	0	0	1	0	2	2	PBL
(+) Lecturer: Better planning of activities	0	1	0	0	0	1	1	Benefits for lecturers
(+) Lecturer: Critical analysis on effective learning	0	1	0	0	0	1	1	Benefits for lecturers
(+) Understanding of professional software development	1	0	0	0	0	1	1	Impr. students' attitude
Element: Badges	0	0	0	0	1	1	1	Gamification
Element: Challenge	1	0	0	0	0	1	1	Gamification
Element: Evidences of application of skills	0	1	0	0	0	1	1	PBL
Element: Freedom to choose or negotiate aspects of the project	1	0	0	0	0	1	1	PBL
Element: Partial deliveries	0	0	0	0	1	1	1	PBL
Element: Simulation of professional practices	1	0	0	0	0	1	1	PBL
Total	28	31	9	16	35	119		

\*NDI: number of distinct interviews mentioning each code.



As a result, the most recurring positive aspects were related to improvements in students attitudes towards the assignment: “Engagement and interest” (13 occurrences), “Motivation” (12 occurrences), and “Participation and interaction with lecturers” (12 occurrences). The first is related to increased interest and engagement of the students in lectures and in the execution of tasks. The second is related to increased motivation for students to strive for better results, to participate in activities, or simply by appearing more enthusiastic about the execution of the assignment. The third is related to increased participation of the students, by actively asking questions, and sharing difficulties and experiences with the lecturer during the assignment.

However, only two codes was unanimous between the lecturers: “Participation and interaction with lecturers” and “Element: Roadmap of activities”. The later is related to the benefits of providing students with a organized plan of activities for the entire project, which was often seen as a cause of systematic delivery of activities by students. The codes “Engagement and interest”, “Element: Competition”, “Element: Gamification”, “Meaningful contextualization of practice”, and “Motivation” were mentioned in at least four distinct interviews.

The positive aspects were grouped in four categories: “improvements in students’ attitude”, “benefits for lecturers”, “gamification”, and “PBL”. Table 6.10 presents the total number of occurrences of the codes for each category. The category with higher occurrence of its codes is “Improvements in students’ attitudes”, which groups the codes related to positive changes in students outcomes and behaviour during the assignment. The categories “PBL” and “Gamification” group codes related to specific mentions of these methods and their respective elements. The codes related to these categories were mentioned in equal proportion (25 and 24 occurrences, respectively).

**Table 6.10.** Categories of codes mapped from the interviews.

Category	# Occurrences	Average NDI
Impr. in students attitude	57	2,7
PBL	25	2,0
Gamification	24	2,4
Benefits for lecturers	13	1,8

#### 6.4.5 Negative aspects of GaPSEE approach and improvements for replications of the case studies

Reflecting on the execution of the case studies, the lecturers mentioned some aspects they would have done differently. Lecturers from cases SWE, WEB, and SQM mentioned changing the strategy for team composition. For the first and second, in future



applications of the approach they would enforce the formation of teams with complimentary skills or experiences, rather than relying in students affinity. However, the later suggest that would not restrict teams in size or composition.

[SQM] *“I would allow more freedom [for students] to select their teams. Students should not be forced to work with people they do not know or do not have affinity. And would create a document clearly defining the gamification rules, and how the grading works. Distinguishing clearly that gamification concepts do not necessarily impact in higher grades”.*

[WEB] *“Forming the teams based on complimentary skill levels, rather than on students affinities. Teams formed by people with previous experience had better performance than teams formed by people without experience”.*

[SWE] *“I would define teams with complimentary skills (would not let this decision to students)”.*

The lecturer from case SWE also mentioned that he would put more effort in the explanation of the project goal, and reinforce its goals in the beginning of each level. The lecturer from the case PLT also mentioned he would insist more remembering students about the dates of deadlines, and track students closer. Finally, the lecturer from the case SQM would document and describe in more details the boundaries between gamification and grading, reinforcing the importance of the guideline *GAPSEE-15*.

[SWE] *“I would worry more about explaining the goal of the project. I thought students had understood the goal of the tasks, however they misunderstood them, and when I realized it, it was too close to the delivery date. (...) Maybe I could assign the first day of each level for the students to explain the goal of that level (instead of me), to validate their understanding of the tasks”.*

[PRO] *“Would dedicate more time to explain the format of the assignment and the gamification before the start of the assignment”.*

[PLT] *“Would insist more about the delivery dates with reminders. And would track students progress closer”.*

Regarding the lecturer position on negative aspects of the GaPSEE approach, the lecturers of cases PLT, PRO, and SWE mentioned the students difficulty in working in



teams. Other shortcomings mentioned were: the amount of time consumed for tracking students' progress in comparison to traditional assignments (PLT); the confusion (for students) regarding the boundaries between gamification scores and grading (SQM); the lack of treatment for different learning profiles (SWE); the lack of systematic support for gamification (SWE); and the difficulty in tracking individual contribution of students (PRO).

[SQM] *"Evaluation was confusing (for students), because it was hard to distinguish what were grades and gamification points."*

[WEB] *"People with less experience or previous knowledge, made less questions. Don't know if this caused by insecurity or because the questions are solved inside the teams. In previous installments, these students asked more, even though their questions were not necessarily related to the course. There are people who do not like to work with periodic deadlines, and would prefer to have a single delivery"*

[SWE] *"Students who do not know how to work in teams. Lack of treatment for different students profile (eg., introverts, students who take competition too seriously and forget about learning goals, and students who do not know how to work in teams). I missed a systematic support for gamification. A tool of platform. Gamification requires immediate feedback, and tool to support this process would increase the benefits of the approach"*

[PRO] *"It consumes more time to track students progress in comparison to traditional practical assignments"*

[PLT] *"Teamwork: some students do not contribute and others work more. However, this is intrinsic from group activities"*

Most of these negative aspects, however, are related to the implementation of the framework and not about the approach itself. Therefore, we agree that the framework does not consider different profiles of learners, and that the effort required to track students progress is a challenge for the adoption of the approach.

#### 6.4.6 Perspective on the use and recommendation of GaPSEE

All lecturers described the approach as relevant to introduce practice in classroom, all of them said they would use GaPSEE again and recommend it to other lecturers. They



were also positive in the perspective of using GaPSEE in other courses they lecture. The lecturer of SQM mentioned that the use of this approach in other courses should be gradual. The lecturer of WEB even considered using the approach in other courses not related to software engineering, such as “compiler design” and “theory of computing”, after some study on how to adapt it.

The lecturers mentioned that the documentation of GaPSEE recommendations and process are useful for an initial use of the approach. The lecturer of SWE mentioned that GaPSEE provides a complete educational approach, allowing lecturer to focus on reflections about development of competences. We observed that, during the meetings for planning the assignments of each case study, having an example case (the PLT study) helped the lecturers in understanding the format of the assignments proposed by GaPSEE. Therefore, a practical guide would be useful for helping GaPSEE users in their first contact with framework.

A final comment provided by the SWE lecturer about GaPSEE approach worth mentioning is:

[SWE] *“Student recognition is a small thing that we eventually fail to do over time, but it makes a lot of difference for students”.*

## 6.5 Discussion

The experience of using of principles PBL and Gamification to plan and execute practical assignments in software engineering related courses, in accordance with the recommendations of GaPSEE, was successful, both for students and lecturers. In the following paragraphs we summarize the main observations made in relation to the research questions defined in Section 6.1.1.

Regarding RQ1 (What is the lecturers’ perception towards a practical assignment planned and executed in accordance with the recommendations of GaPSEE?), the results of the interviews (Section 6.4) show a positive perception of the lecturers towards the joint use of gamification and PBL for introducing practice in software engineering education. The results show that these methods are complimentary: PBL introduces practice in a meaningful way, and gamification contributes with incentives for students motivation, participation and engagement in the execution of projects. GaPSEE seems useful for a gradual introduction of both PBL and gamification elements, in a structured approach. For instance, the lecturer of the case SQM, seemed more interested in gamification and cautious about PBL. In his case, the introduction of elements of PBL were subtle. Additionally, there was unanimity about the relevance of the use of a



gamified roadmap of activities to support the execution of projects. PBL suggests the use of ill-structured problems/projects. However, students often feel lost with this lack of structure and objective directions. Therefore, this element of GaPSEE introduced the idea of badges, tasks, and achievements to organize the projects in a structure that is both more approachable for students and related to process based approaches of software engineering.

In respect to RQ2 (What is the students' perception towards a practical assignment planned and executed in accordance with the recommendations of GaPSEE?), the results from the survey with students (section 6.3) show a positive attitude of the students towards the practical assignments, in terms of relevance, the goals of gamification, development of skills, and general positive aspects. It seems that, for students, the approach was specially welcomed because of the opportunity to escape from the traditional methods, and the opportunity to try a practical approach where they could apply the theory in meaningful contexts. Additionally, there were 10 occurrences of positive codes related to the role of the lecturers in the assignments, which is coherent to the lecturers perception to an increased participation of the students. For students, gamification was specially relevant for creating a competitive environment, and because of the feedback dynamic. PBL was relevant because of its practical nature. However, PBL and the project structure generated mixed feelings in the students. While some enjoyed because they could work in an entire project, with incremental deliveries, that simulated professional environments, others still felt that the instructions were not clear enough, and were not pleased with the restriction in the selection of projects or tools.

Finally, for RQ3 (What are the main benefits and drawbacks of the adoption of Gamification and PBL principles for software engineering education, in accordance with the recommendations of GaPSEE?), this evaluation study confirmed our previous position about how these methods compliment each other, which was supported both by students and lecturers. PBL is an effective learn-by-doing method, and students become more active in this approach, and satisfied by the application of theory. Gamification provides incentives for students to engage in the execution of the project, and provides a strategy to introduce a roadmap of activities to guide students towards the application of specific skills in the assignment. However, there are inherent challenges from PBL which remains troublesome. For instance, teamwork is a skill that is desired to be developed along the courses. However, the difficulties to work in teams that students already have cause discomfort. It is hard for lecturers to objectively measure the individual contribution of students in the assignments, and asking them to evaluate teammates is not the most appropriate approach. Additionally, GaPSEE does not con-



sider the diverse profiles of learners, and not everyone is comfortable with competition, with pressure, with forced interaction with peers. Although these are typical situations of professional environments, not all students like to be exposed to these situations.

From the observation of lecturers using GaPSEE to plan practical assignments for their respective courses, we derived the following lessons in relation to the lecturer role:

- GaPSEE enforced a reflection on skills that students should develop during the course. Therefore, lecturers had to prioritize and organize their lectures in order to really provide students with significant support for the development of the projects.
- Practical assignments were already used by the lecturers. However, the lecturers used to apply short assignments with no continuity. The adoption of a long term project, with sequential iterations that linked several learning topics helped creating more meaningful examples of practice. This practice resulted in additional effort, specially in planning the assignment. However this additional effort is manageable, and the result was positively acknowledged by students.
- Gamification is useful to create continuous challenges for students, and to create an objective measurement of students progress. However, gamification is not necessarily fun. Lecturers must be careful when planning the gamification mechanics, in order to keep the students focused in what really matters: the expected learning outcomes.
- Students appreciated the escape from traditional teaching methods. In addition to being able to apply theory, students were particularly interested in learning new tools and technologies that are used in actual professional software engineering activities.
- All case studies proposed projects to address controlled problems. Yet, only four students complained about the project themes, and only three complained about the technologies used in the projects. Therefore, we believe that the lack of authentic problem is not deterrent. The use of authentic methods, procedures and tools is sufficient to engage students in meaningful practical learning experience.
- Both students and lecturers perceived improvements in engagement and interest. Competition is a key cause pointed by students for increased engagement and determination to achieve better results.



- The approach was manageable in the case studies with different number of students. The “WEB” case study had 39 enrolled students, and even with no teaching assistant and no tools for systematic support for gamification, the effort to execute the assignment was appropriated. The roadmap of activities decreased the dependency of students towards the lecturer guidance, which is important to address the scalability challenge of PBL. However, the most recurring negative aspect pointed by students was related to “unclear information or instructions”.
- Having a roadmap of activities was praised as a positive aspect of the approach, both by students and lecturers. This has proven particularly useful to address the PBL problem of students feeling lost as consequence of the ill-structured nature of PBL projects.

## 6.6 Threats to validity

In this section, we document potential threats to the study validity and discuss some bias that may have affected the study results. We also explain our actions to mitigate them.

**Selection of case studies:** The educators invited for participation in the case studies were selected by convenience. This decision was made because of the expected long duration of the case studies, the necessity of availability of the lecturers for periodic meetings, and the specific scope of the courses for the experiment. Nevertheless, the results also considered the perspective of students, which was in accordance with the perspective of educators.

**Results:** The results presented in the study are first and foremost observations, suggestions and lessons learned for further research. These results are tied to the GaPSEE strategy for the adoption of PBL and gamification. Therefore, it is not representative of all possible strategies to adopt these methods individually or in conjunction. These results also reflect individual perceptions of students and lecturers, and our interpretations of their responses. However, there may be several other important issues in the data collected, not yet discovered or reported by us. Nevertheless, our reports may provide significant insights for other researchers and educators when planning or evaluating educational approaches in similar settings. Additionally, it is hard to distinguish the evaluation of the framework and the evaluation of the specific implementations in each case study.

**Questionnaires and interview scripts:** In order to avoid the risk of misinterpretations of the questions, the questionnaires were developed in stages. The first



version of the questionnaire was reviewed by two researchers who are also software engineering professors. It was then piloted with three students in order to assess if the questions were clear, not ambiguous, and if the available options for answers were coherent. Additionally, the participation in the questionnaire was not compulsory, it preserved the participants anonymity, the participation did not contribute for grades, and the questionnaires were always applied at the conclusion of the course. These decisions were made to avoid the bias of students providing positive answers for the sake of fearing bad consequences or hoping that it would somehow benefit them. For the interviews, the script was first piloted in the PLT case study. The use of semi-structured interviews was also relevant for allowing the adaptation of questions on-the-fly, in case the interviewer noticed any possible misunderstanding of questions. After the conclusion of the analysis, the final manuscript related to the results of the interviews were sent to the interviewees for validation. This validation did not result in change requests from interviewees.

**Number of Participants:** A larger number of participants should be interviewed to capture the general view of a broader audience. However, the time and effort required to execute the case studies were restrictive. In addition, this type of study is limited by the availability of professors willing to allow the researcher to participate in their teaching activities, and that were willing to use the approaches considered for this study. However, we opted to execute the case studies in distinct institutions in order to cover a broader audience of students and courses. The sample of students for each case study was limited to the population of students that (i) were enrolled in the courses and (ii) were willing to participate in the questionnaire. By forcing students to participate in the questionnaire, or rewarding the participation with grades, we would introduce more bias and it would be ethically inappropriate.

**Population sampling:** The participants of the study are from different institutions. Therefore, there is the bias of the participants having different backgrounds and the comparison not being adequate. However, the target population of GaPSEE are lecturers of courses related to software engineering education. Therefore, the diversification of institutions is positive to capture perspectives of lectures inserted in different educational environments, with heterogeneous students.

## 6.7 Final Remarks

This chapter described the planning and execution of a set of case studies to evaluate practical assignments derived from the recommendations of GaPSEE. Data collected



from lecturers and students show a positive attitude of the participants towards the assignments. These results are evidences of: (i) the adequacy of GaPSEE for the planning and execution of practical assignments in software engineering education, and the consequent (ii) adequacy of the joint adoption of gamification and PBL to support software engineering education. Contributing to the specific goal SG3 of this thesis (Investigate the benefits and drawbacks of the joint use of gamification and PBL to support software engineering education), we observed that the joint use of PBL and gamification allows a meaningful contextualization of practice, with incentives for participation, engagement, and motivation of students. The proposal of a gamified roadmap of activities provides a structured guide for students to develop specific skills, without limiting their freedom to choose how to achieve results. As negative aspects, there is the increased effort (however, manageable) to plan the project, and to systematically support gamification. In addition, it is still difficult to track the individual contribution of students, and problems related to teamwork (that are inherent to group activities) is uncomfortable for students. Finally, a limitation of GaPSEE is the lack of consideration for different profiles of learners.



# Chapter 7

## Conclusion

This chapter summarizes the results of this thesis, regarding its goals, contributions and future work. Section 7.1 summarizes the key findings of this thesis. Section 7.2 reviews our main contributions. Section 7.3 outlines possible ideas for future work.

### 7.1 Summary

Introducing practice in software engineering education is challenging, however it is also necessary. Active learning methods are specially relevant for introducing practice in educational environments, and project-based learning is specially useful in the context of engineering and computing education. Similarly, the use of game based approach is not a novelty in software engineering education. However, gamification is a recent trend that has been popularizing in educational context. In this thesis, we propose and evaluate the joint adoption of PBL and gamification to support practice in software engineering education. To achieve this goal, we conducted a set of empirical studies in order to understand how these methods contribute to software engineering education, both individually and in conjunction. Then we proposed a conceptual framework to support lecturers in planning and execution of practical assignments for software engineering related courses, using principles from PBL and gamification. Therefore, this thesis summarizes lessons and perceptions from 10 software engineering related courses that used these methods.

In relation to the specific goal SG1 of this thesis (*“Investigate how PBL can be used to support software engineering education”*), the literature study in Chapter 2 revealed that PBL is a long established educational approach that is useful to contextualize practice, by exposing students to a learn-by-doing paradigm that encourages teamwork and collaboration. However, there are several challenges related to the use



of PBL, related to the setup of PBL courses, the selection of projects, tracking learning outcomes, and managing teamwork and different profiles of learning. In Chapter 4, we confirmed some of these challenges in empirical studies: the difficulty to scale PBL; the importance of balancing authenticity and control in PBL projects; the difficulty of students in understanding what to focus on, and to track their progress; and the difficulty in balancing students freedom of choice and voice with sufficient guidance towards expected learning outcomes.

In relation to the specific goal SG2 of this thesis (*“Investigate how gamification can be used to support software engineering education”*), our systematic literature study (Chapter 2) revealed that gamification is not a “stand-alone” educational method, i.e., it is used as a secondary device, frequently related to raising students engagement and motivation. Literature shows two perspectives for gamification in education: the gamification of the classroom experience, a generalist approach; and the gamification of specific practices of software engineering, a specialist approach. In Chapter 3, we experimented the use of the generalist approach, with limited results. In Chapter 4, we introduced the specialist approach in order to address specific issues of PBL courses, by systematically rewarding students for the application of software engineering practices, providing directions for students with a roadmap of activities, and promoting increased engagement through competition.

In relation to the specific goal SG3 of this thesis (*“Investigate the benefits and drawbacks of the joint use of gamification and PBL to support software”*), we defined a framework (GaPSEE) summarizing the lessons from Chapters 2, 3 and 4, in order to provide a systematic approach to adopt principles of gamification and PBL in practical assignments in software engineering courses (Chapter 5). Then, we conducted five case studies with four lecturers using GaPSEE, and collected data about the perception of students and lecturers (Chapter 6). For lecturers, the main benefits of using gamification and PBL in conjunction are: increased students participation and interaction with lecturers; increased engagement and motivation of students; meaningful contextualization of practice; and having a roadmap of activities to guide and track the progress of students. For students, the main benefits are related to: the format of the assignment (e.g., practical application of theory, and escaping from traditional educational methods); improvements on learning (e.g., getting to know new technologies, engagement, motivation); the competition and having a roadmap of activities to guide them. For the drawbacks, the approach proposed by GaPSEE still suffers from some problems of PBL related to teamwork and the different profiles of learners. The additional effort required to setup and execute GaPSEE was mentioned as manageable by lecturers. For the students, the drawbacks are related to additional time consumed in this type



of assignment, and some aspects related to the project structure (e.g., need for more objective instructions, disagreement with the selection of technologies and projects). Therefore, our results show that the joint use of gamification and PBL was received with more positive than negative perception, from both students and lecturers.

## 7.2 Contribution

As discussed in Chapter 2, the use of gamification and PBL suffers from the lack of recommendations and models for the systematic adoption of these models, considering the specificities from software engineering education. Therefore, the novelty of this thesis lies on the proposal of conceptual framework that supports lecturers in adapting key principles from PBL and gamification to specific characteristics of software engineering education. Additionally, we investigated specific problems of PBL that could be addressed by the introduction of gamification, such as scalability, difficulty in tracking students progress, and the students' difficulty in dealing with the ill-structured nature of PBL projects.

This thesis resulted in the following contributions.

- We documented results of a **systematic study of the literature about game-related methods in software engineering education**, defining, comparing and mapping these methods in three categories: gamification, Game-Based Learning, and Game Development Based Learning.
- We documented results of an ad-hoc **study of the literature on the use of PBL for software engineering education**, exposing key challenges on the use of such educational method.
- We collected **lessons from 5 software engineering courses using gamification and or PBL** to support the definition of GaPSEE. Providing empirical data about the use of these methods to support software engineering education.
- We defined a **conceptual framework (GaPSEE) to support lecturers in the use of gamification and PBL** for planning and executing practical assignments in software engineering education. The framework proposes a set of guidelines, a process, and suggestions of implementation.
- We collected **data from five case studies to evaluate GaPSEE** in relation to the perception of 4 lecturers and 76 students in 5 software engineering related courses.



- We summarized **data about the perception of 76 students and four lecturers on the use of GaPSEE in five case studies.**

## 7.3 Future Work

Considering that our first attempt to evaluate GaPSEE was positive, these results are only preliminary indicatives of the usefulness and adequacy of the joint use of gamification and PBL to support software engineering education. Therefore, it is necessary to expand the scope of our evaluation before proper generalization of our results, with additional case studies and experiments comparing this approach to other educational methods.

During the execution of case studies, we perceived that the different background experience of the lecturers impacted in different levels of readiness to adapt GaPSEE. Therefore, we believe that the documentation of GaPSEE should evolved in order to include instructions for the gradual incorporation of its recommendations. For instance, a model prioritizing sets of recommendations for gradual incorporation over course installments would be useful for lecturers who had never used gamification nor PBL.

One limitation of GaPSEE is not dealing with different profiles of learners. This problem is also discussed in the PBL literature [Zhi, 2016], and has been investigated in the context of educational gamification [Knutas et al., 2014]. In the case studies, we observed students who were motivated by competition, students who used the elements provided by the approach to compete with themselves, and students who were not fond of gamification. Therefore, future work includes the characterization of the different profiles of software engineering learners in practical assignments, and the inclusion of recommendations in GaPSEE in order to achieve positive impact to a broader audience, or to support further tailoring of the approach to address specific profiles.

Other limitation that is relevant to be further explored in future work is the necessity of a customizable tool to support the implementation of GaPSEE. As mentioned by one of the lecturers in the case studies (Chapter 6), software support not only could improve the systematic provision of immediate feedback, but also increase students immersion.

Finally, future work could also explore the adoption of GaPSEE in other knowledge areas. For instance, in Chapter 6, one of the lecturers mentioned his intention to investigate the use of GaPSEE in compiler design. Therefore, it is important to evaluate the adequacy of GaPSEE to other knowledge areas, and investigate the changes needed to achieve adequacy.



# Bibliography

- Akpolat, B. S. and Slany, W. (2014). Enhancing software engineering student team engagement in a high-intensity extreme programming course using gamification. In *2014 IEEE 27th Conference on Software Engineering Education and Training (CSEET)*, pages 149–153. ISSN 1093-0175.
- Alhammad, M. M. and Moreno, A. M. (2018). Gamification in software engineering education: A systematic mapping. *Journal of Systems and Software*, 141:131–150. ISSN 0164-1212.
- Andrews, J. H. and Lutfiyya, H. L. (2000). Experiences with a software maintenance project course. *IEEE Transactions on Education*, 43(4):383–388. ISSN 0018-9359.
- Ardis, M., Fairley, D., Hilburn, T., Nidiffer, K., Towhidnejad, M., and Willshire, M. (2014). The software engineering competency model (swecom). Technical report, Los Alamitos, CA, USA.
- Baker, A., Navarro, E. O., and van der Hoek, A. (2005). An experimental card game for teaching software engineering processes. *Journal of Systems and Software*, 75(1):3–16. ISSN 0164-1212.
- Barnes, T., Powell, E., Chaffin, A., and Lipford, H. (2008). Game2learn: Improving the motivation of cs1 students. In *Proceedings of the 3rd International Conference on Game Development in Computer Science Education*, pages 1–5, New York, NY, USA. ACM.
- Bedwell, W. L., Pavlas, D., Heyne, K., Lazzara, E. H., and Salas, E. (2012). Toward a taxonomy linking game attributes to learning: An empirical study. *Simulation & Gaming*, 43(6):729–760.
- Bell, J., Sheth, S., and Kaiser, G. (2011). Secret ninja testing with halo software engineering. In *Proceedings of the 4th international workshop on Social software engineering*, pages 43–47. ACM.



- Bender, W. N. (2012). *Project-Based Learning: Differentiating Instruction for the 21st Century*. Corwin. ISBN 978-1-4129-9790-4.
- Bessa, B., Cunha, M., and Furtado, F. (2012). Engsoft: Ferramenta para simulação de ambientes reais para auxiliar o aprendizado baseado em problemas (pbl) no ensino de engenharia de software. In *XX Workshop sobre Educação em Computação (WEI)*.
- Blake, M. B. (2003). A student-enacted simulation approach to software engineering education. *IEEE Transactions on Education*, 46(1):124–132. ISSN 0018-9359.
- Blumenfeld, P. C., Soloway, E., Marx, R. W., Krajcik, J. S., Guzdial, M., and Palincsar, A. (1991). Motivating project-based learning: Sustaining the doing, supporting the learning. *Educational Psychologist*, 26(3-4):369–398.
- Booch, G., Rumbaugh, J., and Jacobson, I. (2005). *The Unified Modeling Language User Guide*. Addison Wesley, 2nd edition.
- Bourque, P., Fairley, R. E., et al. (2014). *Guide to the software engineering body of knowledge (SWEBOK (R)): Version 3.0*. IEEE Computer Society Press.
- Chao, J. and Randles, M. (2009). Agile software factory for student service learning. In *22nd Conference on Software Engineering Education and Training*, pages 34–40. ISSN 1093-0175.
- Chen, C. Y., Hong, Y. C., and Chen, P. C. (2014). Effects of the meetings-flow approach on quality teamwork in the training of software capstone projects. *IEEE Transactions on Education*, 57(3):201–208. ISSN 0018-9359.
- Chou, Y.-k. (2015). *Actionable gamification: Beyond points, badges, and leaderboards*. Octalysis Group.
- Coughlan, P. and Coghlan, D. (2002). Action research for operations management. *International journal of operations & production management*, 22(2):220–240.
- Dal Sasso, T., Mocci, A., Lanza, M., and Mastrodicasa, E. (2017). How to gamify software engineering. In *2017 IEEE 24th International Conference on Software Analysis, Evolution and Reengineering (SANER)*, pages 261--271. IEEE.
- Dantas, R., Barros, M., and Werner, C. (2004). A simulation-based game for project management experiential learning. In *In Proceedings of the Sixteenth International Conference on Software Engineering and Knowledge Engineering (SEKE'04)*. Cite-seer.



- Daun, M., Salmon, A., Weyer, T., Pohl, K., and Tenbergen, B. (2016). Project-based learning with examples from industry in university courses: An experience report from an undergraduate requirements engineering course. In *2016 IEEE 29th International Conference on Software Engineering Education and Training (CSEE&T)*, pages 184–193.
- Davison, R., Martinsons, M. G., and Kock, N. (2004). Principles of canonical action research. *Information Systems Journal*, 14(1):65–86.
- Delgado, D., Velasco, A., Aponte, J., and Marcus, A. (2017). Evolving a project-based software engineering course: A case study. In *2017 IEEE 30th Conference on Software Engineering Education and Training (CSEE&T)*, pages 77–86.
- Deterding, S., Sicart, M., Nacke, L., O’Hara, K., and Dixon, D. (2011). Gamification. using game-design elements in non-gaming contexts. In *CHI’11 Extended Abstracts on Human Factors in Computing Systems*, pages 2425–2428.
- Dicheva, D., Dichev, C., Agre, G., and Angelova, G. (2015). Gamification in education: A systematic mapping study. *Educational Technology Society*, 18(3):75–88. ISSN EISSN-1436-4522.
- Diniz, G. C., Silva, M. A. G., Gerosa, M. A., and Steinmacher, I. (2017). Using gamification to orient and motivate students to contribute to oss projects. In *2017 IEEE/ACM 10th International Workshop on Cooperative and Human Aspects of Software Engineering (CHASE)*, pages 36–42. IEEE.
- dos Santos, A. L., Maurício, R. d. A., Dayrell, M., and Figueiredo, E. (2018a). Exploring game elements in learning programming: An empirical evaluation. In *2018 IEEE Frontiers in Education Conference (FIE)*, pages 1–9. IEEE.
- dos Santos, A. L., Souza, M. R. A., Figueiredo, E., and Dayrell, M. (2018b). Game elements for learning programming: A mapping study. In *Proceedings of the 10th International Conference on Computer Supported Education (CSEDU)*, pages 89–101. INSTICC, SciTePress.
- dos Santos, P. S. M. and Travassos, G. H. (2009). Action research use in software engineering: An initial survey. In *2009 3rd International Symposium on Empirical Software Engineering and Measurement*, pages 414–417. ISSN 1949-3770.
- Dubois, D. J. and Tamburrelli, G. (2013). Understanding gamification mechanisms for software development. In *Proceedings of the 2013 9th Joint Meeting on Foundations*



- of Software Engineering*, ESEC/FSE 2013, pages 659–662, New York, NY, USA. ACM.
- Easterbrook, S., Singer, J., Storey, M., and Damian, D. (2008). Selecting empirical methods for software engineering research. In *Guide to advanced empirical software engineering*, pages 285–311. Springer.
- Ellis, H. J. C., Morelli, R. A., Lanerolle, T. R., and Hislop, G. W. (2007). Holistic software engineering education based on a humanitarian open source project. In *20th Conference on Software Engineering Education Training (CSEET)*, pages 327–335. ISSN 1093-0175.
- Fagerholm, F. and Vihavainen, A. (2013). Peer assessment in experiential learning assessing tacit and explicit skills in agile software engineering capstone projects. In *2013 IEEE Frontiers in Education Conference (FIE)*, pages 1723–1729. ISSN 0190-5848.
- Fernandes, E., Oliveira, J., and Figueiredo, E. (2016). Investigating how features of online learning support software process education. In *2016 IEEE Frontiers in Education Conference (FIE)*, pages 1–8.
- Figueiredo, E., Pereira, J. A., Garcia, L., and Lourdes, L. (2014). On the evaluation of an open software engineering course. In *2014 IEEE Frontiers in Education Conference (FIE) Proceedings*, pages 1–8. ISSN 0190-5848.
- Fioravanti, M. L., Sena, B., Paschoal, L. N., Silva, L. R., Allian, A. P., Nakagawa, E. Y., Souza, S. R., Isotani, S., and Barbosa, E. F. (2018). Integrating project based learning and project management for software engineering teaching: an experience report. In *Proceedings of the 49th ACM Technical Symposium on Computer Science Education*, pages 806–811. ACM.
- Fuchs, M. and Wolff, C. (2016). Improving programming education through gameful, formative feedback. In *2016 IEEE Global Engineering Education Conference (EDUCON)*, pages 860–867. IEEE.
- Fukuyasu, N., Saiki, S., Igaki, H., Matsumoto, S., and Kusumoto, S. (2013). A case study of cloud-enabled software development pbl. In *2013 14th ACIS International Conference on Software Engineering, Artificial Intelligence, Networking and Parallel/Distributed Computing*, pages 499–504.



- Garcia, F., Pedreira, O., Piattini, M., Cerdeira-Pena, A., and Penabad, M. (2017). A framework for gamification in software engineering. *Journal of Systems and Software*, 132(C):21–40. ISSN 0164-1212.
- Gary, K. (2015). Project-based learning. *Computer*, 48(9):98–100. ISSN 0018-9162.
- Grant, S. and Betts, B. (2013). Encouraging user behaviour with achievements: an empirical study. In *Proceedings of the 10th Working Conference on Mining Software Repositories*, pages 65–68. IEEE Press.
- Hamari, J. (2017). Do badges increase user activity? a field experiment on the effects of gamification. *Computers in Human Behavior*, 71:469–478. ISSN 0747-5632.
- Hanakawa, N. (2015). Contest based learning with blending software engineering and business management: For students’ high motivation and high practice ability. In *2015 IEEE/ACM 37th IEEE International Conference on Software Engineering*, volume 2, pages 360–369. ISSN 0270-5257.
- Harms, S. and Hastings, J. (2016). A cross-curricular approach to fostering innovation such as virtual reality development through student-led projects. In *2016 IEEE Frontiers in Education Conference (FIE)*, pages 1–9.
- Herold, M. J., Lynch, T. D., Ramnath, R., and Ramanathan, J. (2012). Student and instructor experiences in the inverted classroom. In *Frontiers in Education Conference (FIE)*, pages 1–6. IEEE.
- Herranz, E., Colomo-Palacios, R., de Amescua Seco, A., and Yilmaz, M. (2014). Gamification as a disruptive factor in software process improvement initiatives. *j-jucs*, 20(6):885–906.
- Hesse-Biber, S. N. (2010). *Mixed methods research: Merging theory with practice*. Guilford Press.
- Hevner, A. and Chatterjee, S. (2010). Design science research in information systems. In *Design research in information systems*, pages 9–22. Springer.
- Hunicke, R., LeBlanc, M., and Zubek, R. (2004). Mda: A formal approach to game design and game research. In *Proceedings of the AAAI Workshop on Challenges in Game AI*, volume 4, page 1722.
- IEEE/ACM, T. J. T. F. o. C. C. A. (2015). Software engineering 2014: Curriculum guidelines for undergraduate degree programs in software engineering. Technical report, New York, NY, USA.



- Jazayeri, M. (2015). Combining mastery learning with project-based learning in a first programming course: An experience report. In *IEEE/ACM 37th IEEE International Conference on Software Engineering*, pages 315–318. ISSN 0270-5257.
- Kizaki, S., Tahara, Y., and Ohsuga, A. (2014). Software development pbl focusing on communication using scrum. In *2014 IIAI 3rd International Conference on Advanced Applied Informatics*, pages 662–669.
- Klock, A. C. T., Gasparini, I., and Pimenta, M. S. (2016). 5w2h framework: a guide to design, develop and evaluate the user-centered gamification. In *Proceedings of the 15th Brazilian Symposium on Human Factors in Computing Systems*, page 14. ACM.
- Knutas, A., Ikonen, J., Maggiorini, D., Ripamonti, L., and Porras, J. (2014). Creating software engineering student interaction profiles for discovering gamification approaches to improve collaboration. In *Proceedings of the 15th International Conference on Computer Systems and Technologies, CompSysTech '14*, pages 378–385, New York, NY, USA. ACM.
- Krusche, S., Reichart, B., Tolstoi, P., and Bruegge, B. (2016). Experiences from an experiential learning course on games development. In *Proceedings of the 47th ACM Technical Symposium on Computing Science Education, SIGCSE '16*, pages 582–587, New York, NY, USA. ACM.
- Kuhrmann, M. and Münch, J. (2018). Enhancing software engineering education through experimentation: An experience report. In *2018 IEEE International Conference on Engineering, Technology and Innovation (ICE/ITMC)*, pages 1–9. ISSN .
- Laskowski, M. (2015). Implementing gamification techniques into university study path - a case study. In *2015 IEEE Global Engineering Education Conference (EDUCON)*, pages 582–586. ISSN 2165-9559.
- Long, J., James, N., and Young, L. S. (2011). Multiplayer on-line role playing game style grading in a project based software engineering technology capstone sequence. In *American Society for Engineering Education*. American Society for Engineering Education.
- Macias, J. A. (2012). Enhancing project-based learning in software engineering lab teaching through an e-portfolio approach. *IEEE Transactions on Education*, 55(4):502–507. ISSN 0018-9359.



- Marques, M., Ochoa, S. F., Bastarrica, M. C., and Gutierrez, F. J. (2018). Enhancing the student learning experience in software engineering project courses. *IEEE Transactions on Education*, 61(1):63–73. ISSN 0018-9359.
- Marques, M. R., Quispe, A., and Ochoa, S. F. (2014). A systematic mapping study on practical approaches to teaching software engineering. In *2014 IEEE Frontiers in Education Conference (FIE) Proceedings*, pages 1–8. ISSN 0190-5848.
- Martin, J. G., López, C. L., and Martínez, J. E. P. (2014). Supporting the design and development of project based learning courses. In *2014 IEEE Frontiers in Education Conference (FIE) Proceedings*, pages 1–6. ISSN 0190-5848.
- MEC (2016). Catálogo nacional de cursos superiores de tecnologia. Technical report.
- Meira, S. (2015). Sistemas de informação e engenharia de software – cadê as escolas? *Computação Brasil*, 28:11–15.
- Mora, A., Riera, D., Gonzalez, C., and Arnedo-Moreno, J. (2015). A literature review of gamification design frameworks. In *2015 7th International Conference on Games and Virtual Worlds for Serious Applications (VS-Games)*, pages 1–8.
- Moreno, A. M., Sanchez-Segura, M. I., Medina-Dominguez, F., and Carvajal, L. (2012). Balancing software engineering education and industrial needs. *J. Syst. Softw.*, 85(7):1607–1620.
- Mäkiö, J., Mäkiö-Marusik, E., Yablochnikov, E., Arckhipov, V., and Kipriianov, K. (2017). Teaching cyber physical systems engineering. In *IECON 2017 - 43rd Annual Conference of the IEEE Industrial Electronics Society*, pages 3530–3535.
- Navarro, E. and van der Hoek, A. (2009). Multi-site evaluation of simse. *SIGCSE Bull.*, 41(1):326–330. ISSN 0097-8418.
- Nguyen, D. M., Truong, T. V., and Le, N. B. (2013). Deployment of capstone projects in software engineering education at duy tan university as part of a university-wide project-based learning effort. In *2013 Learning and Teaching in Computing and Engineering*, pages 184–191.
- Pedreira, O., García, F., Brisaboa, N., and Piattini, M. (2015). Gamification in software engineering – a systematic mapping. *Information and Software Technology*, 57:157–168. ISSN 0950-5849.



- Peixoto, D. C. C., Possa, R. M., Resende, R. F., and Pádua, C. I. P. S. (2011). An overview of the main design characteristics of simulation games in software engineering education. In *2011 24th IEEE-CS Conference on Software Engineering Education and Training (CSEET)*, pages 101–110. ISSN 1093-0175.
- Peixoto, D. C. C., Resende, R. F., and Pádua, C. I. P. S. (2014). Evaluating software engineering simulation games: The ugalco framework. In *2014 IEEE Frontiers in Education Conference (FIE) Proceedings*, pages 1–9. ISSN 0190-5848.
- Petri, G., von Wangenheim, C. G., and Borgatto, A. F. (2017). A large-scale evaluation of a model for the evaluation of games for teaching software engineering. In *2017 IEEE/ACM 39th International Conference on Software Engineering: Software Engineering Education and Training Track (ICSE-SEET)*, pages 180–189. IEEE.
- Pfleeger, S. L. and Kitchenham, B. A. (2001). Principles of survey research: Part 1: Turning lemons into lemonade. *SIGSOFT Softw. Eng. Notes*, 26(6):16–18. ISSN 0163-5948.
- Prikladnicki, R., Albuquerque, A. B., von Wangenheim, C. G., and Cabral, R. (2009). Ensino de engenharia de software: Desafios, estratégias de ensino e lições aprendidas. In *Anais do FEES09 Fórum de Educação em Engenharia de Software*.
- Radermacher, A., Walia, G., and Knudson, D. (2014). Investigating the skill gap between graduating students and industry expectations. In *Companion Proceedings of the 36th International Conference on Software Engineering*, pages 291–300, New York, NY, USA. ACM.
- Razali, R. and Chitsaz, M. (2010). Cases development for teaching software engineering. In *2nd International Conference on Education Technology and Computer (ICETC)*, volume 2.
- Rodrigues, P., Souza, M., and Figueiredo, E. (2018). Games and gamification in software engineering education: A survey with educators. In *2018 IEEE Frontiers in Education Conference (FIE)*, pages 1–9. IEEE.
- Runeson, P., Host, M., Rainer, A., and Regnell, B. (2012). *Case study research in software engineering: Guidelines and examples*. John Wiley & Sons.
- Rupakheti, C. R., Hays, M., Mohan, S., Chenoweth, S., and Stouder, A. (2017). On a pursuit for perfecting an undergraduate requirements engineering course. In *2017 IEEE 30th Conference on Software Engineering Education and Training (CSEET)*, pages 97–106.



- Sancho-Thomas, P., Fuentes-Fernández, R., and Fernández-Manjón, B. (2009). Learning teamwork skills in university programming courses. *Computers Education*, 53(2):517–531. ISSN 0360-1315.
- Schwaber, K. and Sutherland, J. (2016). The scrum guide (2013). Available at: <http://www.scrumguides.org/docs/scrumguide/v1/scrum-guide-us.pdf>.
- Shuto, M., Washizaki, H., Kakehi, K., Fukazawa, Y., Yamato, S., and Okubo, M. (2016). Learning effectiveness of team discussions in various software engineering education courses. In *2016 IEEE 29th International Conference on Software Engineering Education and Training (CSEE&T)*, pages 227–231.
- Singer, L. and Schneider, K. (2012). It was a bit of a race: Gamification of version control. In *2012 Second International Workshop on Games and Software Engineering: Realizing User Engagement with Game Engineering Techniques (GAS)*, pages 5–8. ISSN .
- Sommerville, I. (2010). *Software engineering*. New York: Addison-Wesley, 9th edition.
- Souza, M., Moreira, R., and Figueiredo, E. (2019a). Playing the project: Incorporando a gamificação em abordagens baseadas em projetos para a educação em engenharia de software. In *Anais do XXVII Workshop sobre Educação em Computação*, pages 71–80, Porto Alegre, RS, Brasil. SBC. ISSN 2595-6175.
- Souza, M., Moreira, R., and Figueiredo, E. (2019b). Students perception on the use of project-based learning in software engineering education. In *Proceedings of XXXIII SBES - Education Track*.
- Souza, M. R. A., Constantino, K. F., Veado, L. F., and Figueiredo, E. M. L. (2017a). Gamification in software engineering education: An empirical study. In *2017 IEEE 30th Conference on Software Engineering Education and Training (CSEE&T)*, pages 276–284.
- Souza, M. R. A., Veado, L., Moreira, R. T., Figueiredo, E., and Costa, H. (2018). A systematic mapping study on game-related methods for software engineering education. *Information and Software Technology*, 95:201–218. ISSN 0950-5849.
- Souza, M. R. A., Veado, L. F., Moreira, R. T., Figueiredo, E. M. L., and Costa, H. A. X. (2017b). Games for learning: bridging game-related education methods to software engineering knowledge areas. In *2017 IEEE/ACM 39th International Conference on Software Engineering: Software Engineering Education and Training Track (ICSE-SEET)*, pages 170–179.



- Stol, K. J., Ralph, P., and Fitzgerald, B. (2016). Grounded theory in software engineering research: A critical review and guidelines. In *2016 IEEE/ACM 38th International Conference on Software Engineering (ICSE)*, pages 120–131.
- Sunaga, Y., Shuto, M., Washizaki, H., Kakehi, K., Fukazawa, Y., Yamato, S., and Okubo, M. (2016). Which combinations of personal characteristic types are more effective in different project-based learning courses? In *2016 IEEE 29th International Conference on Software Engineering Education and Training (CSEET)*, pages 137–141.
- Sunaga, Y., Washizaki, H., Kakehi, K., Fukazawa, Y., Yamato, S., and Okubo, M. (2017). Relation between combinations of personal characteristic types and educational effectiveness for a controlled project-based learning course. *IEEE Transactions on Emerging Topics in Computing*, 5(1):69–76. ISSN 2168-6750.
- Team, C. P. (2010). Cmmi® for development, version 1.3. Technical report, Carnegie Mellon University.
- Thevathayan, C. (2018). Evolving project based learning to suit diverse student cohorts. In *Proceedings of the 22nd International Conference on Evaluation and Assessment in Software Engineering 2018*, pages 133–138. ACM.
- Uskov, V. and Sekar, B. (2014). Gamification of software engineering curriculum. In *2014 IEEE Frontiers in Education Conference (FIE) Proceedings*, pages 1–8. ISSN 0190-5848.
- von Wangenheim, C. G. and da Silva, D. A. (2009). Qual conhecimento de engenharia de software é importante para um profissional de software? In *Anais do II Fórum de Educação em Engenharia de Software (FEES)*, pages 1–8.
- von Wangenheim, C. G. and Shull, F. (2009). To game or not to game? *IEEE Softw.*, 26(2):92–94. ISSN 0740-7459.
- Warin, B., Talbi, O., Kolski, C., and Hoogstoel, F. (2016). Multi-role project (mrp): A new project-based learning method for stem. *IEEE Transactions on Education*, 59(2):137–146. ISSN 0018-9359.
- Werbach, K. and Hunter, D. (2012). *For the Win: How Game Thinking Can Revolutionize Your Business*. Wharton Digital Press. ISBN 978-1-613-63023-5.
- Wilhelm, W. J., Logan, J., Smith, S. M., and Linda, L. F. (2002). Meeting the demand: Teaching "soft" skills. *Institute of Education Sciences*, page 80.



- Winterfeldt, G. and Hahne, C. (2014). Controlling quad-copters a project-based approach in the teaching of application design. In *2014 IEEE Global Engineering Education Conference (EDUCON)*, pages 961–968. ISSN 2165-9559.
- Wohlin, C., Runeson, P., Hst, M., Ohlsson, M. C., Regnell, B., and Wessln, A. (2012). *Experimentation in Software Engineering*. Springer Publishing Company, Incorporated. ISBN 9783642290435.
- Wu, B. and Wang, A. I. (2012). A guideline for game development-based learning: A literature review. *Int. J. Comput. Games Technol.*, 2012. ISSN 1687-7047.
- Yamada, Y., Inaga, S., Washizaki, H., Kakehi, K., Fukazawa, Y., Yamato, S., Okubo, M., Kume, T., and Tamaki, M. (2014). The impacts of personal characteristic on educational effectiveness in controlled-project based learning on software intensive systems development. In *2014 IEEE 27th Conference on Software Engineering Education and Training (CSEE T)*, pages 119–128. ISSN 1093-0175.
- Zhi, G. C. (2016). A project-based blended learning mode for mobile applicaton development course. In *2016 11th International Conference on Computer Science Education (ICCSE)*, pages 757–762. ISSN 978-1-4799-3922-0.
- Zichermann, G. and Cunningham, C. (2011). *Gamification by Design - Implementing Game Mechanics in Web and Mobile Apps*. O'Reilly. ISBN 978-1-449-39767-8.
- Zorzo, A., Nunes, D., Matos, E., Steinmacher, I., Leite, J., Araujo, R., Correia, R., and Martins, S. (2017). Referenciais de formação para os cursos de graduação em computação. sociedade brasileira de computação (sbc). 153p, 2017. Technical report.







# Appendix A

## Primary Studies used in the Systematic Mapping

[S1] B. S. Akpolat and W. Slany, “Enhancing software engineering student team engagement in a high-intensity extreme programming course using gamification”, IEEE Conference on Software Engineering Education and Training, 2014.

[S2] A. L. D. Buisman and M. C. J. D. Van Eekelen, “Gamification in educational software development”, Computer Science Education Research Conference, 2014.

[S3] K. Berkling and C. Thomas, “Gamification of a software engineering course and a detailed analysis of the factors that lead to it’s failure”, International Conference on Interactive Collaborative Learning, 2013.

[S4] V. Uskov and B. Sekar, “Gamification of software engineering curriculum”, Frontiers in Education Conference, 2014.

[S5] M. Laskowski, “Implementing gamification techniques into university study path - A case study”, IEEE Global Engineering Education Conference, 2015.

[S6] J.N. Long and L. S. Young, “Multiplayer on-line role playing game style grading in a project based software engineering technology capstone sequence”, ASEE Annual Conference and Exposition, 2011.

[S7] C. Thomas and K. Berkling, “Redesign of a gamified Software Engineering course”, International Conference on Interactive Collaborative Learning, 2013.

[S8] W. Q. Qu, Y. F. Zhao, M. Wang and B. Liu, “Research on teaching gamification of software engineering”, International Conference on Computer Science Education, 2014.

[S9] D. F. Bacon, D. C. Parkes, Y. Chen, M. Rao, I. Kash, and M. Sridharan, “Predicting your own effort,” in Proceedings of the 11th International Conference on Autonomous Agents and Multiagent Systems (AAMAS’12), pp. 695–702, 2012.



[S10] J. Bell, S. Sheth and G. Kaiser, “Secret Ninja Testing with HALO Software Engineering”, International Workshop on Social Software Engineering, 2011.

[S11] A. A. de Melo, M. Hinz, G. Scheibel, C. D. M. Berkenbrock, I. Gasparini, and F. Baldo, “Version Control System Gamification: A Proposal to Encourage the Engagement of Developers to Collaborate in Software Projects,” presented at the Proceedings of the 6th Int. Conf. on Social Computing and Social Media (SCSM’2014), 2014.

[S12] S. Dencheva, C. R. Prause, and W. Prinz, “Dynamic Selfmoderation in a Corporate Wiki to Improve Participation and Contribution Quality,” presented at the Proceedings of the 12th European Conference on Computer Supported Cooperative Work (ECSCW’11), 2013.

[S13] A. Dorling, McCaffery, F., “The gamification of SPICE,” Communications in Computer and Information Science, vol. 290, pp. 295–301, 2012.

[S14] D. Duarte, C. Farinha, M. M. da Silva, and A. R. da Silva, “Collaborative Requirements Elicitation with Visualization Techniques,” Proceedings of the IEEE 21st International Workshop on Enabling Technologies: Infrastructure for Collaborative Enterprises (WETICE’12), 2012.

[S15] D. J. Dubois and G. Tamburrelli, “Understanding gamification mechanisms for software development,” Proceedings of the 9th Joint Meeting on Foundations of Software Engineering (ESEC/FSE’13), 2013, pp. 659–662.

[S16] J. Fernandes, D. Duarte, C. Ribeiro, C. Farinha, J. Madeiras Pereira, and M. Mira da Silva, “iThink: A Game-Based Approach Toward Improving Collaboration and Participation in Requirement Elicitation,” Procedia Computer Science, vol. 15, pp. 66–77, 2012 2012.

[S17] S. Grant and B. Betts, “Encouraging user behavior with achievements: an empirical study,” Proceedings of the 10th Working Conference on Mining Software Repositories (MSR’13), 2013, pp. 65–68

[S18] K. Januszewski, “Visual Studio Achievements Programs Brings Gamification to Development”, 2012.

[S19] T. D. LaToza, U. o. C. Dept. of Inf., Irvine, Irvine, CA, USA, W. Ben Towne, A. van der Hoek, and J. D. Herbsleb, “Crowd development,” Proceedings of the 6th Workshop on Cooperative and Human Aspects of Software Engineering (CHASE’13), pp. 85–88, 2013.

[S20] S. Nikkila, D. Byrne, H. Sundaram, A. Kelliher, and S. Linn, “Taskville: visualizing tasks and raising awareness in the workplace,” Proceedings of Conference on Human Factors in Computing Systems (CHI’2013), 2013.



[S21] S. Nikkila, S. Linn, H. Sundaram, and A. Kelliher, “Playing in Taskville: Designing a Social Game for the Workplace,” Proceedings of Int. Conf. on Computer Human Interaction (CHI’2011) – Workshop on Gamification: Using Game Design Elements in Non-Game Contexts, 2011.

[S22] C. R. Prause, J. Nonnen, and M. Vinkovits, “A Field Experiment on Gamification of Code Quality in Agile Development,” Proceedings of the 24th Annual Workshop of the Psychology of Programming Interest Group (PPIG’2012), 2012.

[S23] L. Singer, F. Figueira Filho, B. Cleary, C. Treude, M.-A. Storey, and K. Schneider, “Mutual assessment in the social programmer ecosystem: an empirical investigation of developer profile aggregators,” Proceedings of the Conference on Computer supported cooperative work (CSCW’13), pp. 103–116, 2013.

[S24] L. Singer and K. Schneider, “Influencing the adoption of software engineering methods using social software,” Proceedings of the International Conference on Software Engineering (ICSE’12), 2012.

[S25] L. Singer and K. Schneider, “It was a bit of a race: Gamification of version control,” Proceedings of the 2nd International Workshop on Games and Software Engineering (GAS’12), pp. 5–8, 2012.

[S26] W. Snipes, V. Augustine, A. R. Nair, and E. Murphy-Hill, “Toward recognizing and rewarding efficient developer work patterns,” Proceedings of the International Conference on Software Engineering (ICSE’13), pp. 1277–1280, 2013.

[S27] W. Snipes, A. R. Nair, and E. Murphy-Hill, “Experiences gamifying developer adoption of practices and tools,” Proceedings of the 36th International Conference on Software Engineering (ICSE’2014), 2014.

[S28] R. Sukale and M. S. Pfaff, “QuoDocs: improving developer engagement in software documentation through gamification,” Proceedings of the 32nd ACM Conference on Human Factors in Computing Systems (ACHI’2014), 2014.

[S29] J. Thom, D. Millen, and J. DiMicco, “Removing gamification from an enterprise SNS,” Proceedings of the ACM Conference on Computer Supported Cooperative Work (CSCW’12), pp. 1067–1070, 2012.

[S30] R. Troughton, “Gamification in Software Development and Agile,” Software Developer’s Journal, 2012.

[S31] B. Vasilescu, A. Serebrenik, P. Devanbu, and V. Filkov, “How social QA sites are changing knowledge sharing in open source software communities,” Proceedings of the 17th ACM conference on Computer supported cooperative work social computing (CSCW’2014), 2014.

[S32] P. G. F. Matsubara and C. L. C. da Silva, “Game Elements in a Software Engineering Study Group: A Case Study”, Proceedings of the 39th International Con-



ference on Software Engineering: Software Engineering and Education Track, 2017.

[S33] A. Bartel, P. Figas and G. Hagel, "Towards a Competency-based Education with Gamification Design Elements", Proceedings of the 2015 Annual Symposium on Computer-Human Interaction in Play, 2015.

[S34] I. Chow and L. Huang, "A Software Gamification Model for Cross-Cultural Software Development Teams", Proceedings of the 2017 International Conference on Management Engineering, Software Engineering and Service Sciences, 2017.

[S35] D. Elm, G. F. Tondello, D. L. Kappen, M. Ganaba, M. Stocco and L. E. Nacke, "CLEVER: A Trivia and Strategy Game for Enterprise Knowledge Learning", Proceedings of the 2016 Annual Symposium on Computer-Human Interaction in Play Companion Extended Abstracts, 2016.

[S36] S. Albertarelli, F. Dassenno, L. Galli and G. Pasceri, "The Rise of Serious Games and Gamified Application in Software Development", Proceedings of the Second ACM International Conference on Mobile Software Engineering and Systems, 2015.

[S37] S. K. Sripada, Y. R. Reddy and S. Khandelwal, "Architecting an Extensible Framework for Gamifying Software Engineering Concepts", Proceedings of the 9th India Software Engineering Conference, 2016.

[S38] S. -K. Thiel and U. Lehner, "Exploring the Effects of Game Elements in M-participation", Proceedings of the 2015 British HCI Conference, 2015.

[S39] G. C. Diniz, M. A. G. Silva, M. A. Gerosa and I. Steinmache, "Using Gamification to Orient and Motivate Students to Contribute to OSS Projects", Proceedings of the 10th International Workshop on Cooperative and Human Aspects of Software Engineering, 2017.

[S40] L. C. Stanculescu, A. Bozzon, R.-J. Sips and G.-J. Houben, "Work and Play: An Experiment in Enterprise Gamification", Proceedings of the 19th ACM Conference on Computer-Supported Cooperative Work Social Computing, 2016.

[S41] K. Kurihara, "Toolification of Games: Achieving Non-game Purposes in the Redundant Spaces of Existing Games", Proceedings of the 12th International Conference on Advances in Computer Entertainment Technology, 2015.

[S42] O. Korn, M. Funk and A. Schmidt, "Towards a Gamification of Industrial Production: A Comparative Study in Sheltered Work Environments", Proceedings of the 7th ACM SIGCHI Symposium on Engineering Interactive Computing Systems, 2015.

[S43] L. Lema Moreta; A. C. Gamboa; M. G. Palacios, "Implementing a Gamified application for a Risk Management course", IEEE Ecuador Technical Chapters Meeting (ETCM), 2016.



[S44] F. Kifetew; D. Munante; A. Perini; A. Susi; A. Siena; P. Busetta, "DMGame: A Gamified Collaborative Requirements Prioritisation Tool", IEEE 25th International Requirements Engineering Conference (RE), 2017.

[S45] P. Busetta; F. M. Kifetew; D. Munante; A. Perini; A. Siena; A. Susi, "Tool-Supported Collaborative Requirements Prioritisation", IEEE 41st Annual Computer Software and Applications Conference (COMPSAC), 2017.

[S46] F. Steffens; S. Marczak; F. F. Filho; C. Treude; C. R. B. de Souza, "A Preliminary Evaluation of a Gamification Framework to Jump Start Collaboration Behavior Change", IEEE/ACM 10th International Workshop on Cooperative and Human Aspects of Software Engineering (CHASE), 2017.

[S47] T. Dal Sasso; A. Mocci; M. Lanza; E. Mastrodicasa, "How to gamify software engineering", IEEE 24th International Conference on Software Analysis, Evolution and Reengineering (SANER), 2017.

[S48] T. Barik; E. Murphy-Hill; T. Zimmermann, "A perspective on blending programming environments and games: Beyond points, badges, and leaderboards", IEEE Symposium on Visual Languages and Human-Centric Computing (VL/HCC), 2016.

[S49] J. Brito; V. Vieira; A. Duran, "Towards a Framework for Gamification Design on Crowdsourcing Systems: The G.A.M.E. Approach", 12th International Conference on Information Technology - New Generations, 2015.

[S50] A. Bernik; D. Radošević; G. Bubaš, "Introducing gamification into e-learning university courses", 40th International Convention on Information and Communication Technology, Electronics and Microelectronics (MIPRO), 2017.

[S51] A. Bartel; G. Hagel, "Gamifying the learning of design patterns in software engineering education", IEEE Global Engineering Education Conference (EDUCON), 2016.

[S52] C. B. Chirila; R. Raes; A. Roland, "Towards a generic gamification of sorting algorithms", 12th IEEE International Symposium on Electronics and Telecommunications (ISETC), 2016.

[S53] N. Unkelos-Shpigel; I. Hadar, "Gamifying Software Engineering Tasks Based on Cognitive Principles: The Case of Code Review", IEEE/ACM 8th International Workshop on Cooperative and Human Aspects of Software Engineering, 2015.

[S54] F. M. Kifetew; D. Munante; A. Perini; A. Susi; A. Siena; P. Busetta; D. Valerio, "Gamifying Collaborative Prioritization: Does Pointsification Work?", IEEE 25th International Requirements Engineering Conference (RE), 2017.

[S55] L. Piras; P. Giorgini; J. Mylopoulos, "Acceptance Requirements and Their Gamification Solutions", IEEE 24th International Requirements Engineering Confer-



ence (RE), 2016.

[S56] N. Unkelos-Shpigel; I. Hadar, "Inviting everyone to play: Gamifying collaborative requirements engineering", IEEE Fifth International Workshop on Empirical Requirements Engineering (EmpiRE), 2015.

[S57] M. Fuchs; C. Wolff, "Improving programming education through gameful, formative feedback", IEEE Global Engineering Education Conference (EDUCON), 2016.

[S58] M. Z. H. Kolpondinos; M. Glinz, "Behind Points and Levels — The Influence of Gamification Algorithms on Requirements Prioritization", IEEE 25th International Requirements Engineering Conference (RE), 2017.



## Appendix B

# Questionnaire for the Evaluation of the Framework

This Appendix presents the questionnaire used in the survey study described in Chapter 6.



## Section 1 - Participant's Background

- What program are you attending?
  - ☐ Information System (Undergrad.)
  - ☐ Computer Science (Undergrad.)
  - ☐ Computer Science (Grad.)
- Age
  - ☐ 20 or less
  - ☐ 21 - 25
  - ☐ 26 - 30
  - ☐ 31 - 35
  - ☐ 36 or greater
- Professional experience in software engineering or software development (including trainee and internship programs)
  - ☐ None
  - ☐ Less than 1 year
  - ☐ 1 – 2 years
  - ☐ 3 – 5 years
  - ☐ More than 5 years
- How often do you play games?
  - ☐ Never
  - ☐ Rarely
  - ☐ Monthly
  - ☐ Weekly
  - ☐ Daily

## Section 2 – Relevancy and adequacy of the practical assignment

- How important is “practice” to the learning process of this course?

Completely irrelevant:	<input type="radio"/> <input type="radio"/> <input type="radio"/> <input type="radio"/> <input type="radio"/>	Highly important:
This course should	1   2   3   4   5	Practice is imperative
focus only in theory.		for this course.

- How appropriate was the practical assignment to this course?

Completely inadequate	<input type="radio"/> <input type="radio"/> <input type="radio"/> <input type="radio"/> <input type="radio"/>	Completely adequate
	1   2   3   4   5	



### Section 3 – About the gamification

- Evaluate each of the following affirmative

Challenge	Totally disagree	Disagree	Neutral	Agree	Totally Agree
Q1. The tasks of the assignment were adequately challenging	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
Q2. Having a roadmap of activities made me confident about the execution of the assignment	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>

Confidence	Totally disagree	Disagree	Neutral	Agree	Totally Agree
Q3. The interaction with the lecturer kept me confident on what needed to be done	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
Q4. Receiving feedback made me more confident that I was in the right path.	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>

Engagement	Totally disagree	Disagree	Neutral	Agree	Totally Agree
Q5. I felt engaged in the execution of the tasks	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
Q6. I wanted to complete the maximum number of tasks possible.	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>

Motivation	Totally disagree	Disagree	Neutral	Agree	Totally Agree
Q7. The format of the practical assignment kept motivated in the execution of the project	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
Q8. The possibility of having my team acknowledged as “the best” motivated me in pursuing better results	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
Q9. The possibility of receiving rewards motivated me in pursuing better results	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
Q10. The possibility of receiving feedback to improve my results motivated me to complete tasks in antecedence	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>



<b>Collaboration</b>	Totally disagree	Disagree	Neutral	Agree	Totally Agree
Q11. The practical assignment promoted moments of collaboration among students	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
Q12. The practical assignment promoted improved interaction between students and lecturers	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>

<b>Recognition</b>	Totally disagree	Disagree	Neutral	Agree	Totally Agree
Q13. I felt that my efforts were acknowledged	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
Q14. Having my team ranked as “the best” is a form of personal recognition	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>

<b>Learning</b>	Totally disagree	Disagree	Neutral	Agree	Totally Agree
Q15. The practical assignment contributed to my learning process	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
Q16. The practical assignment was efficient for my learning process	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
Q17. The practical assignment helped me in contextualizing theory in professional practice	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
Q18. The classroom activities contributed to my progress in the project	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
Q19. The feedback from lecturers was important for my learning process	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>

<b>Satisfaction</b>	Totally disagree	Disagree	Neutral	Agree	Totally Agree
Q20. Completing tasks gave me the feeling of accomplishment	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
Q21. I felt satisfied with the results of my team	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
Q22. I felt satisfied with what I learned in the practical assignment	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>



<b>Relevance</b>	Totally disagree	Disagree	Neutral	Agree	Totally Agree
Q23. The project and its tasks were relevant to my personal interests	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
Q24. The link between the assignment and the course is clear	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
Q25. This format of assignment is adequate for this course	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
Q26. I prefer to learn using this method than using other learning methods	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
Q27. I would like to have more courses using this learning method	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>

<b>Fun</b>	Totally disagree	Disagree	Neutral	Agree	Totally Agree
Q28. The format of the assignment was fun	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
Q29. There was a fun situation during the assignment	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
Q30. The practical assignment promoted moments of fun in the competition	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>



### Section 3 – Skill Development

- Evaluate the contribution of the practical assignment towards the development of the following specific skills:

Significant negative contribution (-2); Small negative contribution (-1); No contribution (0); Small positive contribution (1); Significant positive contribution (2)

PRO	(-2)	(-1)	(0)	(1)	(2)
Understand and apply the Configuration Management process from MPS.BR	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
Understand and apply the Measurement process from MPS.BR	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
Adapt and apply tools to support process improvement in specific organizational contexts	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
Adapt process to the needs of specific organizational contexts	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>

SWE	(-2)	(-1)	(0)	(1)	(2)
Elicit and document software requirements	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
Plan a software project	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
Establish a strategy for software testing	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>

SQM	(-2)	(-1)	(0)	(1)	(2)
Define and plan criteria for quality evaluation of a software product	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
Apply and analyze measurements for assessing quality aspects of software product	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
Propose corrective actions to improve the internal quality of a software product	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
Understand the concept of code smells and apply techniques for their identification and treatment.	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
Use tools for static analysis of source code	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>

WEB	(-2)	(-1)	(0)	(1)	(2)
Selection and use of technologies for Web applications	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
Specification, componentization, and prototyping of Web development projects	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
Use of version control systems (Git)	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
Use of GitHub features for planning and documenting the project evolution	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>



- Evaluate the contribution of the practical assignment towards the development of the following general skills:

Significant negative contribution (-2); Small negative contribution (-1); No contribution (0); Small positive contribution (1); Significant positive contribution (2)

General Skills	(-2)	(-1)	(0)	(1)	(2)
Professional Knowledge: develop mastery of software engineering knowledge and skills and of the professional standards necessary to begin practice as a software engineer.	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
Technical Knowledge: Demonstrate an understanding of and apply appropriate theories, models, and techniques that provide a basis for problem identification and analysis, software design, development, implementation, verification, and documentation.	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
Teamwork: Work both individually and as part of a team to develop and deliver quality software artifacts. Design Solutions in Context: Design appropriate solutions in one or more application domains using software engineering approaches that integrate ethical, social, legal, and economic concerns.	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
Perform Trade-Offs: Reconcile conflicting project goals, finding acceptable compromises within the limitations of cost, time, knowledge, existing systems, and organizations.	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
End-User Awareness: Demonstrate an understanding and appreciation of the importance of negotiation, effective work habits, leadership, and good communication with stakeholders in a typical software development environment.	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
Continuing Professional Development: Learn new models, techniques, and technologies as they emerge and appreciate the necessity of such continuing professional development.	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>



#### **Section 4 – Positive and Negative aspects**

- What did you perceived as POSITIVE in the educational method used in the practical assignment?

[Long answer text, with minimum character count of 10]

- What did you perceived as NEGATIVE in the educational method used in the practical assignment?

[Long answer text, with minimum character count of 10]

- Additional comments (compliments, critics, suggestions) [Optional]

[Long answer text, with minimum character count of 10]