Balancing between Creativity and Efficiency in Software Engineering Project Courses

Ruoqing Wang
Technical University of Munich
ruoqing.wang@tum.de

Snezhina Milusheva
Technical University of Munich
snezhina.milusheva@in.tum.de

Stephan Krusche
Technical University of Munich
krusche@in.tum.de

***Abstract*—Practical software engineering courses incorporate industrial clients to present a more realistic environment for the students. Clients introduce problem statements to students in the predevelopment phase. These documents describe the project's vision, scope, requirements, and acceptance criteria. An unreasonable trade-off between efficiency and creativity can lead to unfulfilled client expectations or very constrained creative space for students. In problem statements, a good balance between freedom and efficiency helps foster such a creative-friendly environment that facilitates creative thinking and innovative approaches.**

This paper describes a case study in a software engineering project course among 17 projects (with over 100 students) in the past two semesters with real industrial clients. We develop criteria to classify problem statements into three main types. Innovation and creativity are measured through self-evaluation surveys and project documentation. The findings show that a problem statement with a known problem, fewer than 15 requirements, and fewer than 30 constraint word occurrences has the highest potential to strike the balance to encourage creativity and achieve project success.

***Index Terms*—Creativity, Software Engineering, Project Courses, Problem Statement, Requirements Engineering, Analysis**

I. Introduction

Many studies in software engineering education have shown that project courses with external industrial clients have a positive effect on students' motivation and understanding of theoretical knowledge [6, 8, 10, 11].

The clients write a document called a problem statement to introduce the project to students at the beginning of the practical project course. It contains a problem, visionary scenarios, functional and non-functional requirements, target environment, development environment, client acceptance criteria, deliverables, and schedule. It establishes the first asynchronous communication between the client and the development team and is the starting point of the requirements engineering process. In the absence of proper guidance, problem statements are likely to cause one of two undesirable situations.

The first situation is focusing too much on the efficiency of the project and limiting the developers' freedom and therefore creativity. There are two possible reasons behind this. One is that clients clearly know what the product should look like, and they even have thought of a complete proposed solution, depriving the creativity of the developers. The other one is that the product required by the clients is very complex. In both cases, clients write down a long detailed list of initial requirements. Due to time pressure's negative impact on creative cognition [2], students tend to follow the rules and satisfy the requirements rather than approach the problem in a creative way.

The second situation is focusing too much on creativity which can result in inefficiency. Different from the first situation, the only demand from clients is for the students to come up with something new and creative. Even for experienced developers, it takes a lot of time to brainstorm, discuss, and finally agree on the vision of the product. Most of the students who participate in project courses interact for the first time with a real team project and a real industry client. Therefore, the time spent on implementation would be limited if too much time is spent during requirements engineering. This could result in analysis paralysis and endanger the success of the project, even if valuable creative ideas are generated.

In the nowadays competitive world, time consumption has a major impact on the value of creative ideas. However, the idea of risk-taking and time-consuming has been notably absent in current creative education areas [18]. Balancing between the two extremes described in the above paragraph is yet to be studied.

This paper mainly focuses on creativity and innovation in software engineering, including two main perspectives: creative problem solving and creative problem finding [4]. The value of creative problem solving can be seen as: previously unsolved problems are solved, the results of existing solutions are improved, or the existing solution's efficiency is enhanced. Creative problem discovery refers to finding a completely new problem that has not yet been discovered, and solving this new problem is both valuable and realistic.

The goal of this paper is to investigate combining efficiency and creativity into requirements engineering:

H1: The amount of requirements affects creativity.

H2: Problem statements with unknown problems allow for more creative project results.

The paper is structured as follows. Section 2 describes the related work about the process model, creativity, and requirement engineering. Section 3 introduces a way of classifying problem statements based on the problem, the number of requirements, and constraints word occurrences. Section 4 explains how to measure and evaluate creativity. Section 5 shows a case study conducted on 17 projects among over one

2640-0715/22/$31.00 ©2022 IEEE
DOI 10.1109/APSEC57359.2022.00075

hundred students. In the end, Section 6 describes limitations, and Section 7 summarizes conclusions and recommendations.

II. Related Work

There are many and varied definitions of creativity, depending on the different focuses. A general definition that has been cited over 2000 times is that "Creativity is the interaction among aptitude, process, and the environment by which an individual or group produces a perceptible product that is both novel and useful as defined within a social context." [29]

A more specified definition is described in a 4P innovation model proposed by Runco [30]. In 4P innovation model, there are several major approaches leading to creativity: person(or personality), process, product, and place(or press). One way to evaluate creativity is through measurements from the perspective of the above dimensions [18]. Person includes personality, attitudes, self-concept, and behavior. The process includes motivation, perception, thinking, learning, and communication. Press or Place includes the relationship with the environment. The product includes the idea that is transformed into a tangible form.

Agile software development process with Unified Process, Scrum, and their adaptations have been commonly used in many software engineering project courses [33, 6, 20, 34, 32]. During the first development phase, the identification of scenarios, functional requirements, and non-functional requirements is important.

The problem statement is the first step toward requirements elicitation for the development team. We look into two branches of research on requirements engineering. The first one is anchored in education and the second one investigates the creativity aspect. For both areas, we step onto literature reviews to outline the existing research and missing aspects.

Daun and others [9] review the existing literature on requirements engineering education. They summarize the trends in the field that lean towards including real clients which parallel with the course we describe in the case study. However, creativity is not mentioned among the key teaching trends.

Next in order, Lemos and others [27] focus on creativity in requirements engineering in an industry context. The argument for fostering creativity during requirements engineering is strongly supported by the existing literature. The papers propose various techniques and approaches to requirements elicitation from the client. The responsibility of gathering and understanding the requirements lies within the development team; and then doing that in a creative way. We take a look at the very initial communication tool - the problem statement, and how the client plays a significant role in the predevelopment phase.

One of the biggest challenges for beginner programmers is to apply basic programming knowledge to practice [26]. For project courses that are focusing on improving students' practical skills, there already exist many studies discussing ways to enhance innovation during the process. Decreasing pressure to succeed in every step has been proven to be helpful to creativity [23]. Encouraging pair programming [14, 28] is helpful to motivate students to build closer team bonds, actively participate in the development and improve implementation skills. These methods have been widely applied to various project course teaching.

III. Problem Statement Classification

The classification of problem statements first focuses on the number of requirements and whether problems are specified. A problem is defined as a situation with a goal and some potential obstacles have to be conquered to achieve the goal. A requirement can be defined as a feature that the system must realize or a constraint that the system must satisfy so that the client will accept the system.

There are two different types of requirements in software engineering: functional requirements, and non-functional requirements. Requirements can be directly proposed and listed by the clients or elicited from scenarios and use cases [5]. In problem statement template, clients are required to write down the initial requirements which have been identified as important to them.

- A known problem with high requirements level.
- A known problem with low requirements level.
- Discover unknown problems in some areas.

Among the above three problem statement types, the first two types aim at solving an existing and valuable problem that has been identified by the clients and the last one is about exploring unknown problems. The difference between the first two types is the number of requirements. The distribution of the number of requirements in each document basically follows a normal distribution. The threshold for classification is one standard deviation σ bigger than the mean. Therefore, the majority, around 84%, of the documents, are within the range smaller than the threshold.

The last type is about finding an unknown and valuable problem inside the scope and solving that problem. This type of problem statement is usually about an open topic, for example, creating an app with creative features that support electronic automobile driving. In this case, students need to find a problem that satisfies the client's request, propose some creative ideas, and communicate with clients to define the scenarios and the requirements.

IV. Creativity Measurement Methods

Innovation can be measured by combining analysis of subjective participant perceptions with analysis of objective documentation products. The measurement method measures creativity and innovation through the data of the 4P dimensions at different time periods.

A. Self-evaluation survey from participants

A survey is designed as a subjective way to measure creativity and innovation in the software engineering practical course. A questionnaire survey is an effective tool for large-scale measurement among participants. The respondents of the questionnaire are selective, that is, the participants of the course. The questionnaire is highly recommended to be

anonymous to eliminate the concerns of some participants and collect more sufficient and true survey results. For each option of the single-choice questions, the 5-Point Likert Scale model [21] is used: It is one of the most reliable ways to measure a user's opinions and capture the range of those opinions.

Questions focused more on participants' experiences and self-feelings during the course. Traditional 5L options are: Strongly disagree; Disagree; Neither agree nor disagree; Agree; Strongly agree. They are more tended to guide participants' opinions on an issue, which is not appropriate for questions related to subjective feelings. In order to make it easier and more accurate for participants to find a suitable range, the targeted adaptation options are adverbs: Not at all; Slightly; Moderately; Very; Extremely. In this way, participants can more intuitively relate their initial responses in the question context to corresponding options accordingly through different adverbs, so that the collected information is also more accurate and valuable.

In the meantime, conditional open questions are introduced after the participant picks one answer to a single-choice question. Participants of the survey are encouraged to further conduct self-analysis and reflect based on intuitive feelings, and then give feedback in terms of reasons behind such answers to the instructors. For example, when a participant expresses a negative opinion on being creative, an optional open-ended question asking about the reason why he or she thinks himself or herself is not creative will be added below. The participant will re-consider the previous answer and find reasons to support the answer with more details there. The advantage of this is that more valuable and customized feedback can be collected based on not undermining the enthusiasm of users to participate in the questionnaire.

B. Evaluation from documentation

There are several products that should be delivered to clients at the end of the project. They are a functional application, presentation slides, and documentation. The documentation is written by the students as a white book not only to introduce clients to how to set up and use the application but also to explain the design and implementation details. The documentation contains the purpose of the system, user interface, scenarios, system design, future work, and administrator manual.

By using natural language process analysis on the documentation, the words, and sentences which are related to creativity and innovation are counted to evaluate product creativity. It helps to objectively measure the performance and originality of the system and covers more about the related creative details. This evaluation method is also especially helpful when measuring some previous projects where surveys are impossible to carry out.

V. Case Study

This section describes a case study in a software engineering practical course. The case study contains 17 projects, over 100 students, and 10 clients in the winter semester of 2021 and the summer semester of 2022. In this course, students first learn to master essential knowledge for iOS development and then are divided into teams to develop mobile applications. The topics range from medical treatment, smart home devices, and automotive driving to grocery shopping.

A. Setting

Students are distributed into several teams based on a set of team composition criteria. The team distribution takes the number of students and equipment, programming skills, project interests, and personal factors such as languages, personality traits, and interests into consideration [12, 35]. Apart from student developers, each team is equipped with a project leader and a coach at the same time. Together, they guide the team through the software development process.

The role of the project leader is similar to a product owner in agile development. The project leader makes sure that the product development is aligned with the client's requirements. Similar to practice from Creativity-Support Learning Environment(CSLE) [3], this course does not have teaching roles who give instructions and knowledge directly. Instead, project leaders only facilitate cooperation.

The role of the coach is similar to a scrum master in agile development. The coach is usually a student who has taken the course before and is familiar with the infrastructure and organizational matters. Coach learns management skills by observing project leader and attending meetings with other coaches where senior staff mentors them. Apart from assisting the project leader with management, the coach also acts as a communication channel between the project leader and developers.

Each semester, there are several course-wide workflows independent of all project teams. The workflows are course-wide expert platforms that provide deepening knowledge and help. The workflow experts prepare self-learning materials, presentations, and workshops about respective fields.

In this case study, there are four workflow teams: Modeling; Release; Usability; Development. The workflow experts are taken over by coaches. Usually, three coaches are working on one workflow. Inside each project team, there are corresponding workflow managers who act as liaisons to workflow experts. The workflow experts are responsible to answer workflow managers' questions and help workflow managers resolve problems in that domains.

Similar to Scrum, the team performs regular Sprint planning, Sprint review, and Sprint retrospective inside each Sprint and creates at least one release at the end of each Sprint. Each Sprint also usually takes two weeks. Instead of using traditional daily scrum meetings, weekly team meetings aim to deal with the situation that students participate in this course as part-time developers.

Figure 1 describes the lifecycle of the project course. It shows the average effort of each workflow(Requirements Elicitation, Analysis, Design, Implementation, Test, Project Management, and Release Management) throughout the whole

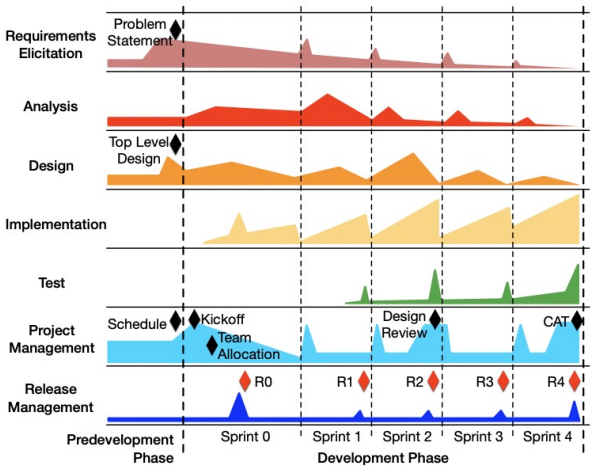

Fig. 1. The lifecycle model of the project course [7, 22]

timeline of the project course. Figure 1 also marks several important events, documents, and releases.

Requirements elicitation starts from the predevelopment phase and continues throughout the whole project course. During the course of the project, some non-documented requirements will arise from meetings and other formal or informal communication. This fluid information, together with solid documented requirements, are both worth considering [31]. Therefore, we also encourage students to use informal models[13] to communicate with customers and exchange ideas within the team. Using formal and informal models in real projects increases students' engagement and understanding of those software engineering concepts, especially modeling, that students often do not like [25].

There are two important meetings marked as milestones during each semester's project course. Both are course-wide events for students to present their work publicly to all clients, all teams, and also their friends.

The first milestone, Design Review, happens around two-thirds time of the course. Each team presents its understanding of the problem, the requirements, the design, and the current status of the system. The team also shows a 60-seconds they shoot in Sprint 0 to demonstrate the application in an attractive way. A demo can be performed by the team in form of a short theater play [24, 36], which can still include workarounds and mocks.

The second milestone, CAT (Client Acceptance Test), takes place at the end of the course, usually three months after Kick-off. Each team presents the requirements and the architecture of the system combined with another demo. Different from the first milestone's demo, what is shown in the demo should be implemented without workarounds and mocks.

This case study is conducted on-site. Students are free to choose wherever they want to work or meet. There are several rooms open to all students that they can use for regular team meetings, workshops, meet-ups, and workflow help desks in the university. All room setup is similar to Schild's Creativity Room 555 [17], an informal and pleasing environment where students can actively discuss and get distracted less by devices. All students are equally offered all-day access to those rooms.

B. Methodology

Here we describe the criteria we use in our case study for problem statement classification and creativity evaluation.

1) Problem statements classification: There are 17 problem statements in total, which are classified into 3 types. The threshold to classify requirements level, 15, is calculated by adding mean, 12, and standard deviation, 3. A problem statement containing no more than 15 requirements is seen as low requirements level, and one with more than 15 requirements is seen as high requirements level.

TABLE I
Problem statements requirements classification results

Category	Projects
Known problems and low requirements level	11
Known problems and high requirements level	5
Unknown problems	1

Table I shows the number of problem statements inside each requirement type.

Unknown problems type is a relatively rare problem statement type. This type appeared only once among the 17 problem statements that we collected.

Inside known problems and low requirements level type, it is further split into two categories based on the number of constraints vocabulary occurrences. The constraint word list contains: must; be able to; can; should; have to. Similar to requirements level classification, constraints vocabulary occurrences classification also uses the same way to pick threshold, which is 30 after calculation.

TABLE II
Low requirements level problem statements constraints classification results

Category	Projects
Constraints words $<$ 30 occurrences	9
Constraints words $\geq$ 30 occurrences	2

Table II shows the number of problem statements of two different constraint types.

2) Creativity measurement: The measurement of creativity is based on a self-evaluation questionnaire and documentation. We invited all 22S semester course participants, around 60 students, to take this questionnaire. Considering everyone can have very different understandings of creativity and innovation, we explained creativity and innovation at the very beginning of the survey. "In this survey, creativity and innovation focus on finding new and useful ways to solve known problems as well as discovering unknown and valuable unknown problems."

The questionnaire contains 9 mandatory single-choice questions and 4 conditional open questions. The design of the questions, inspired by the 4P model, covers person(personality, attitudes, self-concept, and behavior), process(thinking and communication), and product(the idea that is transformed into a tangible form), these three creativity dimensions. The fourth dimension, place(press), is eliminated as all teams are offered the same environment. Table III describes all the questions and their categories in the questionnaire.

TABLE III
Questionnaire questions

Perspective	Question	Title
Person	**Q1**	Before starting the project, do you think you are creative or innovative?
	Q2	After completing the project, do you think you are creative or innovative?
	Q8	Why do you think you are (not) creative or innovative?
Process	**Q3**	Before starting the project, do you feel empowered to propose new ideas in your study or work that are considered creative or innovative?
	Q4	During working on the project, do you feel empowered to propose new ideas or solutions that are considered creative or innovative?
	Q5	After completing the project, do you feel empowered to propose new ideas in your study or work that are considered creative or innovative?
Product	**Q6**	Before starting the project, do you think the project is creative or innovative?
	Q7	After completing the project, do you think the project is creative or innovative?
	Q9	Why do you think the product is (not) creative or innovative?

We analyzed the documentation of all 17 projects. The feature analysis adopts a trait list [15] of more than 600 adjective characteristic words. Using the word vector from spacy [19], en_core_web_lg, which includes 685k unique vectors, the most similar 20 words to ”innovative” are chosen among all 600 traits, such as ”creative”, ”sophisticated”, ”imaginative”, and so on. Next, the documentation is pre-processed by using tokenization and removing stopwords. Then through lemmatization from Spark, all words are assigned to the base forms. In the last step, the CountVectorizer model reads each processed documentation and generates the occurrence of the innovative trait words which we obtained in the first step. The innovation factor of each documentation is inferred by the number of occurrences of innovative-related trait words.

C. Results

All projects were successfully completed in one semester. Clients all expressed their acceptance of the final results of the project. So from an overall measurement, the final outcomes and length of time spent on each project can be considered the same. In other words, the overall efficiency of each project is the same.

The results below include statistics from self-assessment survey answers and the number of creative traits in documentations.

1) Survey results: Among the 60 surveys we sent, 31 complete responses are received. The answers to single-choice questions are grouped into the 3 creativity dimensions: Person, Process, and Product. All answers in each group are about the same focus on one dimension with different time periods: Before starting the project and after completing the project. The answers to the single-choice questions are assigned to specific projects according to the first question asking which project team this participant belongs to. Then the answers of all projects are distributed to the problem statement categories they were previously classified as, such as low requirements level and high requirements level, low requirements level with fewer than 30 constraints words and low requirements level with over 30 constraints words.

Since the number of items under each category is very different and each category contains more than 6 complete responses, the percentage of all answers within a category rather than the number of responses can represent some characteristics of the category more clearly. The counts are transformed into percentage distribution in the final results.

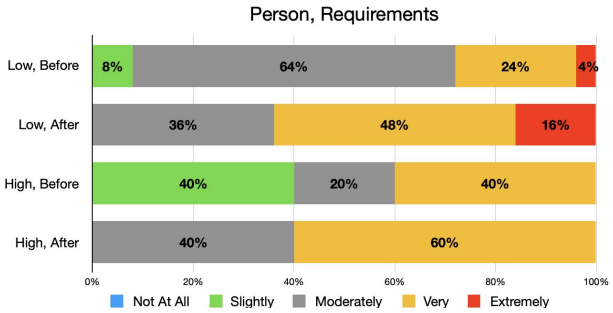

Fig. 2. Do you think you are creative or innovative?

Figure 2 depicts the answer distribution of Q1 and Q2 focusing on person dimension in low requirements level and high requirements level before starting and after completing the projects.

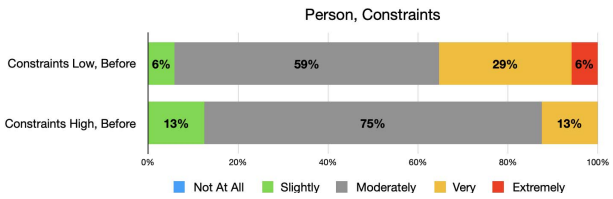

Fig. 3. Before starting the project, do you think you are creative or innovative?

Figure 3 shows the answer distribution of Q1 focusing on person in fewer than 30 constraints word counts type and over 30 constraints word counts type under the low requirements level category.

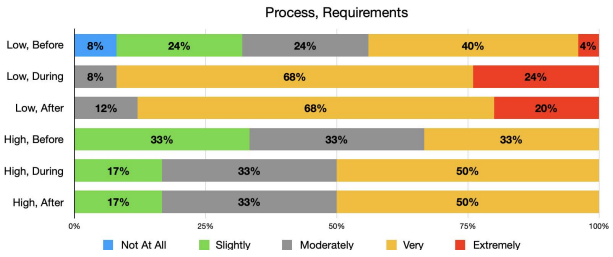

Fig. 4. Are you empowered to propose new ideas in your study or work that are considered creative or innovative?

Figure 4 depicts the answer distribution of the Q3, Q4, and Q5 focusing on process perspective in low requirements level and high requirements level before starting, during development, and after completing the projects.

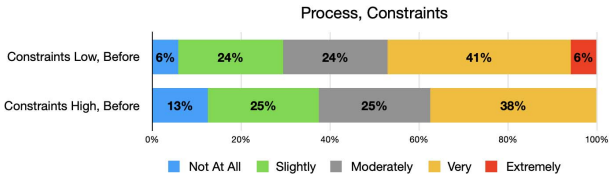

Fig. 5. Before starting the project, do you feel empowered to propose new ideas in your study or work that are considered creative or innovative?

Figure 5 shows the answer distribution of Q3 focusing on process perspective in fewer than 30 constraints word counts type and over 30 constraints word counts type under the low requirements level category.

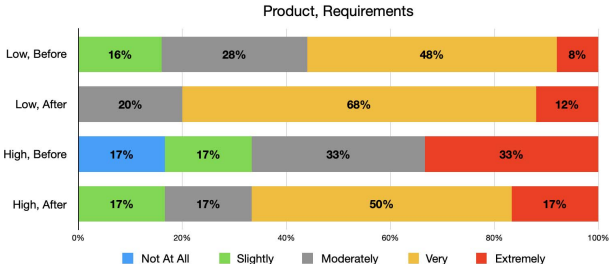

Fig. 6. Do you think the product is creative or innovative?

Figure 6 depicts the answer distribution of Q6 and Q7 focusing on product perspective in low requirements level and high requirements level before starting and after completing the projects.

Figure 7 shows the answer distribution of Q6 focusing on process perspective in fewer than 30 constraints word counts type and over 30 constraints word counts type under the low requirements level category.

For optional open questions, we received 11 answers for each.

The answers to "Why do you think you are creative or innovative" include the following examples.

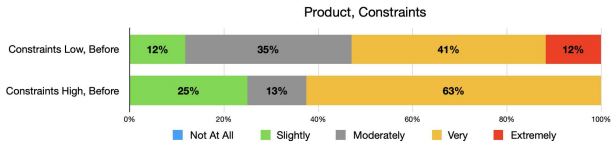

Fig. 7. Before starting the project, do you think the product is creative or innovative?

- There were some things that I proposed adding (sometimes added without proposing) to the app that was actually appreciated by the client, project lead, and team members.
- Always trying to improve everything and make it efficient.
- Non-programming tasks open more opportunities, UI, and software theatre.
- I can think out of the box if needed, and the coaches also positively enforced brainstorming for new ideas that took our project in a different direction. Normal solutions would have made the project very dull.

The answers to "Why do you think the project is creative or innovative" include the following examples.

- The idea of the project was quite futuristic, but not unreal. I can definitely see our app fulfilling the demands of people living in a future, where electric cars are very common.
- The project combines both navigation and AR, there are existing technologies for both, but the combination is still quite new. There are other works out there that combine them, but not a lot.
- It brings about new approaches to dealing with common problems. This system can potentially take care of your medical setup, actively replacing a sales representative, while being easy to use, accessible and portable. It saves time and effort, that's while reaching the same goal, which is innovative and even creative at times.

2) Documentation results: Table IV shows the documentation analysis results of creative or innovation traits on 7 documentations. The innovative traits occurrences are seen as the innovative evaluation factor. The more innovative traits words occurred in one documentation, the more innovative the project is. For known problems and low requirement levels projects, each of their documentation contains 4.25 innovative traits words occurrences on average. For high-requirement levels projects, each contains 3 innovative traits occurrences. And for the unknown problems project type, each documentation has 6 innovative traits words occurrences.

TABLE IV
Documentation traits words occurrences result

Category	Traits
Known problems and low requirement levels	4.25
Known problems and high requirement levels	3
Unknown problems	6

D. Findings

The following findings are generated after comparing the results in different categories and different periods.

> **Finding 1:** All problem statements end up with better creativity and innovation results than before.

In Figure 2, from person perspective, by comparing the before and after course results, we can find the following improvements. For the low requirements level category, the very or extremely parts have risen from 28% to 64%. For moderately and slightly parts, which were the majority before, have decreased from 72% to 36%. Similarly, in the high requirements level category, the parts that exceeds moderately have increased from 40% to 60%, and the parts below moderately have decreased from 60% to 40%. Both types of people who felt that they were only slightly creative before starting the projects, believe that they are at least moderately creative after completing projects.

In Figure 4, from process perspective, after finishing the course, students are more confident to share creative ideas and solutions than before. For the low requirements level category, the beyond moderately empowered parts have risen from 44% to 92%, while the below moderately empowered parts have decreased from 56% to 8%. Those who were afraid of proposing creative ideas and solutions now feel comfortable sharing. In the end, almost all students feel very or extremely empowered to propose creative ideas and solutions. For the high requirements level category, the part that exceeds moderately, which is the very part, has also increased a bit from 33% to 50%, while the moderately and slightly parts which were the majority before, have decreased from 67% to 50%.

In Figure 6, from product perspective, most students find the product more creative than in the beginning. For the low requirements level category, the beyond moderately creative parts, have risen from 56% to 80%. For the high requirements level category, the part that exceeds moderately has increased from 33% to 67%. Those who thought the product is not creative at all changed their opinions after completing the projects.

To sum up, the creativity and innovation measurements in all three dimensions for all problem statements have been improved after the project is completed.

> **Finding 2:** Software engineering students' innovative self-awareness can be improved through external affirmation from the team, appropriate encouragement, and guidance, and a broader space which is not just limited to programming.

From the result of Q8 where students explained the reason why they feel more creative, most students claim that their confidence in their creativity comes from external affirmation, some appropriate encouragement and enforcement, and also the discovery of their creative ability in some non-programming tasks that are not usually found in computer science exercises. After the course, more than 50% of the participants agreed that innovation can be improved through learning.

> **Finding 3:** Low requirements problem statements have a higher potential to achieve higher creative or innovative results.

In Figure 2, the low requirements level problem statements in the category have some participants who consider themselves extremely creative before and after the project. However, nobody from the high requirements level projects feels extremely creative or innovative ever.

In Figure 4, in the low requirements level problem statements category, there are students feeling extremely empowered to propose creative ideas and solutions before, during, and after the project, but there is still nobody in the high requirements level category. After completing the project, 92% of low requirements level projects feel very or extremely empowered to propose creative ideas or solutions while in high requirements level projects only 50% feel empowered to the same extent.

In Figure 6, although before starting, a higher proportion, 33% of participants from the high requirements level type thought their products were extremely creative or innovative, only 17% remain the same extremely creative evaluation in the end. After completing projects, 80% of low requirements level projects think the product is beyond moderately creative or innovative, while for high requirements levels only 67% think so and there are also 17% who think the product is just slightly creative.

To sum up, combining the three innovation dimensions, low requirements problem statements performs better than high requirements problem statements in the most innovative part.

> **Finding 4:** Low requirements level problem statements that contain fewer constraint words make students feel more creative about themselves, the process, and the products before starting development.

In Figure 3, before starting projects, among students of low constraint words occurrences type, only 6% think themselves are below moderately creative, while for students of over 30 constraint words occurrences type, 13% feel below moderately creative. In the meantime, 6% of the fewer constraint words occurrences type think they are extremely creative or innovative, while for over 30 constraint words occurrences type, none think they are extremely creative.

In Figure 5, before starting projects, 9% more students in the low constraint words occurrences type feel more than moderately empowered in over 30 constraint words occurrences type.

In Figure 7, before starting projects, 12% in low constraint words occurrences type think the products are less than moderately creative, whereas in high constraint words occurrences type the proportion is 25%. For extremely creative part comparison, 12% from low constraint type think the products

are extremely creative but nobody from high constraint type thinks so.

To sum up, fewer constraint words have a positive effect on students' creativity and how they think about the product's creativity before starting the projects.

> **Finding 5:** Problem statements with unknown problems create more creative or innovative products.

In Table IV, among these three types, the unknown problem type has the highest innovative traits occurrences, followed by the low requirements level type, and finally the high requirements level type. From this, we can infer that the unknown problem type products are more likely to contain more innovative elements, and the products are more likely to be more creative.

E. Discussion

Creativity depends on being encouraged and affirmed, so creating an environment that welcomes creativity is crucial in teaching. From the survey results, students' self-perception of creativity largely comes from the approval of their creative ideas by leaders and team members. At the end of the project, students generally believe that creativity is a learnable skill, and most of them also believe that they are more creative than before. More importantly, the students feel that they are also more confident and more comfortable bringing up creative ideas so there will be fewer barriers to creativity.

Hypothesis H1: The amount of requirements that affect creativity, is supported by Finding 3. With fewer requirements defined at the beginning of the practical project course, there will be more potential for creativity. Clients should manage to control the number of initial requirements as few as possible, so the initial requirements can indicate the needs and in the meantime, leave more space for creativity.

Hypothesis H2: Problem statements with unknown problems allow for more creative project results, supported by Finding 5. Whether a specific problem is defined also influences creativity. When clients are not sure about the product features, it is an option to leave the initial project description as an open topic and let it to students. It is worth mentioning that since the unknown problem type has only one project, the generality of the impact of this type also deserves continued attention in the future.

The way of expressing the initial description of the project also matters. A more open, suggestive tone to express ideas and reducing the use of constraint words can better encourage innovation.

In addition, we strongly suggest monitoring further communication between the clients and the team. The problem statement is just one gateway to a month-long collaboration between both parties. It proposes a communication and control standard for the project, that could be loosened or strengthened with time.

In our study, since all projects were carried out in the same environment, the influence of environment on creativity was not taken into account. In the future, we also recommend further investigating how much different factors, such as the number of requirements, constraints words, and environment influence creativity.

VI. Limitations

Threat to internal validity: Since the questionnaire is carried out in the end, the feelings in the early stage may be blurred in the participant's memory. The students may tend to present themselves in a better light because they are afraid of getting a worse grade. We, therefore, collected the data anonymously.

Threat to external validity: The case study covers all problem statement types discussed in this paper. However, it is more common to have problem statements defined with a specific problem. There is only 1 project belonging to the unknown problem type among the 17 projects. Additional observations for the upcoming semesters are needed to better understand this type better. Project participants include not only students but also stakeholders and customers. Different roles result in different perspectives and may also perceive innovation differently, so measurements from more roles will also be valuable in future research.

Threat to construct validity: Although we tried to design the questionnaire questions in a simple and general format using a customized Likert scale, the results can still be subject to distortion [16]. Creativity has been proven to be influenced by entrepreneurship [1]. Different stakeholders can also lead to different impacts on creativity and efficiency.

VII. Conclusion

Balancing creativity and efficiency is crucial in teaching practical software engineering courses. In order to better stimulate potential creativity, it is very helpful to create a creative-friendly atmosphere at the beginning of the course.

When a problem has been well identified by clients, reducing the number of initial requirements to less than 15 and the use of constraint words to less than 30 is one approach. Another approach is to grant students the freedom to explore the unknown problem space of a specific topic. We measured and evaluated the creativity of related problem statement types in a case study consisting of 17 projects. Both approaches are proven to have enhanced students' creative skills to a higher level in terms of self-awareness, communication, and implementation.

References

[1] Teresa A Amabile and Mukti Khaire. "Creativity and the role of the leader". In: (2008).

[2] Teresa M Amabile, Jennifer S Mueller, William B Simpson, Constance N Hadley, Steven J Kramer, Lee Fleming, et al. "Time pressure and creativity in organizations: A longitudinal field study". In: (2002).

[3] Mikko Apiola, Matti Lattu, and Tomi A Pasanen. "Creativity-Supporting Learning Environment—CSLE". In: *ACM Transactions on Computing Education (TOCE)* 12.3 (2012), pp. 1–25.

[4] Wayne Brookes. “On creativity and innovation in the computing curriculum”. In: *Proceedings of the 20th Australasian Computing Education Conference*. 2018, pp. 17–24.

[5] Bernd Bruegge and Allen H Dutoit. “Object–oriented software engineering. using uml, patterns, and java”. In: *Learning* 5.6 (2009), p. 7.

[6] Bernd Bruegge, Stephan Krusche, and Lukas Alperowitz. “Software engineering project courses with industrial clients”. In: *ACM Transactions on Computing Education (TOCE)* 15.4 (2015), pp. 1–31.

[7] Bernd Bruegge, Stephan Krusche, and Martin Wagner. “Teaching Tornado: from communication models to releases”. In: *Proceedings of the 8th edition of the Educators’ Symposium*. 2012, pp. 5–12.

[8] Nergiz Ercil Cagiltay. “Teaching software engineering by means of computer-game development: Challenges and opportunities”. In: *British Journal of Educational Technology* 38.3 (2007), pp. 405–415.

[9] Marian Daun, Alicia M Grubb, Viktoria Stenkova, and Bastian Tenbergen. “A systematic literature review of requirements engineering education”. In: *Requirements Engineering* (2022).

[10] Marian Daun, Andrea Salmon, Bastian Tenbergen, Thorsten Weyer, and Klaus Pohl. “Industrial case studies in graduate requirements engineering courses: The impact on student motivation”. In: *27th Conference on Software Engineering Education and Training (CSEE&T)*. IEEE. 2014, pp. 3–12.

[11] Marian Daun, Andrea Salmon, Thorsten Weyer, Klaus Pohl, and Bastian Tenbergen. “Project-based learning with examples from industry in university courses: an experience report from an undergraduate requirements engineering course”. In: *29th International Conference on Software Engineering Education and Training (CSEET)*. IEEE. 2016, pp. 184–193.

[12] Dora Dzvonyar, Lukas Alperowitz, Dominic Henze, and Bernd Bruegge. “Team composition in software engineering project courses”. In: *International Workshop on Software Engineering Education for Millennials (SEEM)*. IEEE. 2018, pp. 16–23.

[13] Dora Dzvonyar, Stephan Krusche, and Lukas Alperowitz. “Real Projects with Informal Models.” In: *EduSymp@ MoDELS*. 2014, pp. 39–45.

[14] Raymond Flood and Bob Lockhart. “Teaching programming collaboratively”. In: *Proceedings of the 10th annual SIGCSE conference on Innovation and technology in computer science education*. 2005, pp. 321–324.

[15] Nikhil Garg, Londa Schiebinger, Dan Jurafsky, and James Zou. “Word embeddings quantify 100 years of gender and ethnic stereotypes”. In: *Proceedings of the National Academy of Sciences* 115.16 (2018), E3635–E3644.

[16] Ron Garland. “The mid-point on a rating scale: Is it desirable”. In: *Marketing bulletin* (1991), pp. 66–70.

[17] Timo Göttel and Jonas Schild. “Creativity room 5555: Evoking creativity in game design amongst CS students”. In: *Proceedings of the 16th annual joint conference on Innovation and technology in computer science education*. 2011, pp. 98–102.

[18] Wouter Groeneveld, Brett A Becker, and Joost Vennekens. “How Creatively Are We Teaching and Assessing Creativity in Computing Education: A Systematic Literature Review”. In: *Proceedings of the 53rd ACM Technical Symposium on Computer Science Education V. 1*. 2022, pp. 934–940.

[19] Matthew Honnibal and Ines Montani. “spaCy 2: Natural language understanding with Bloom embeddings, convolutional neural networks and incremental parsing”. In: *To appear* 7.1 (2017), pp. 411–420.

[20] Ivar Jacobson, Grady Booch, and James Rumbaugh. “The unified process”. In: *IEEE Software* 16.3 (1999), p. 96.

[21] Ankur Joshi, Saket Kale, Satish Chandel, and D Kumar Pal. “Likert scale: Explored and explained”. In: *British journal of applied science & technology* 7.4 (2015), p. 396.

[22] Stephan Krusche. “Rugby-a process model for continuous software engineering”. PhD thesis. Technische Universität München, 2016.

[23] Stephan Krusche, Bernd Bruegge, Irina Camilleri, Kirill Krinkin, Andreas Seitz, and Cecil Wöbker. “Chaordic learning: A case study”. In: *39th International Conference on Software Engineering: Software Engineering Education and Training Track (ICSE-SEET)*. IEEE. 2017, pp. 87–96.

[24] Stephan Krusche, Dora Dzvonyar, Han Xu, and Bernd Bruegge. “Software theater—teaching demo-oriented prototyping”. In: *ACM Transactions on Computing Education (TOCE)* 18.2 (2018), pp. 1–30.

[25] Stephan Krusche, Nadine von Frankenberg, Lara Marie Reimer, and Bernd Bruegge. “An interactive learning method to engage students in modeling”. In: *42nd International Conference on Software Engineering: Software Engineering Education and Training*. IEEE. 2020, pp. 12–22.

[26] Essi Lahtinen, Kirsti Ala-Mutka, and Hannu-Matti Järvinen. “A study of the difficulties of novice programmers”. In: *Acm sigcse bulletin* 37.3 (2005), pp. 14–18.

[27] Joao Lemos, Carina Alves, Leticia Duboc, and Genaina Nunes Rodrigues. “A systematic mapping study on creativity in requirements engineering”. In: *Proceedings of the 27th Annual ACM Symposium on Applied Computing*. 2012, pp. 1083–1088.

[28] Charlie McDowell, Brian Hanks, and Linda Werner. “Experimenting with pair programming in the classroom”. In: *Proceedings of the 8th annual conference on Innovation and technology in computer science education*. 2003, pp. 60–64.

[29] Jonathan A Plucker, Ronald A Beghetto, and Gayle T Dow. “Why isn’t creativity more important to educational psychologists? Potentials, pitfalls, and future directions in creativity research”. In: *Educational psychologist* 39.2 (2004), pp. 83–96.

[30] F Preckel, H Holling, M Weise, R Richards, DK Kinney, M Benet, APC Merzel, AC Sligh, FA Conners, B Roskos-Ewoldsen, et al. “Creativity: Theories and Themes: Research, Development, and Practice by Mark A. Runco, Elsevier Academic Press, 2007, 492 pp. ISBN 13: 978-0-12-602400-5..” In: *Human Physiology* 26 (), pp. 516–522.

[31] Kurt Schneider, Kai Stapel, and Eric Knauss. “Beyond documents: visualizing informal communication”. In: *Requirements Engineering Visualization*. IEEE. 2008, pp. 31–40.

[32] Ken Schwaber. *Agile project management with Scrum*. Microsoft press, 2004.

[33] Ken Schwaber and Mike Beedle. *Agile software development with scrum. Series in agile software development*. Vol. 1. Prentice Hall Upper Saddle River, 2002.

[34] Ken Schwaber and Jeff Sutherland. “The scrum guide”. In: *Scrum Alliance* 21.19 (2011), p. 1.

[35] Masashi Shuto, Hironori Washizaki, Katsuhiko Kakehi, Yoshiaki Fukazawa, Shoso Yamato, Masashi Okubo, and Bastian Tenbergen. “Relationship between the five factor model personality and learning effectiveness of teams in three information systems education courses”. In: *18th International Conference on Software Engineering, Artificial Intelligence, Networking and Parallel/Distributed Computing (SNPD)*. IEEE. 2017, pp. 167–174.

[36] Han Xu, Stephan Krusche, and Bernd Bruegge. “Using software theater for the demonstration of innovative ubiquitous applications”. In: *Proceedings of the 10th Joint Meeting on Foundations of Software Engineering*. 2015, pp. 894–897.