Teaching software design patterns: An experience using the Jigsaw classroom

Lina Garcés
Laboratório de Engenharia de Software, Instituto de Ciências Matemáticas e de Computação, Universidade de São Paulo - LabES/ICMC/USP.
São Carlos, SP, Brazil
linagarces@usp.br

Brauner R. N. Oliveira
Núcleo de Excelência em Tecnologias Sociais, Universidade Federal de Alagoas.
Maceió, AL, Brazil
brauner@alumni.usp.br

ABSTRACT

Software design patterns have been one of the most important topics taught in computer science courses. However, several challenges remain for teaching design patterns in the classroom, such as misconceptions about their utility and students' lack of motivation for learning them. The Jigsaw classroom is a teaching method based on cooperative learning, where students, as principal actors, collaborate to construct their knowledge about a subject. Jigsaw has demonstrated improvements in students' cognitive, affective, and psychosocial skills; however this instructional method has not been properly explored in software engineering teaching to provide evidence of its benefits and drawbacks for education in this area. This study presents an experience on using Jigsaw classroom for teaching software design patterns in an undergraduate course. The preliminary results confirm the benefits and some application challenges the Jigsaw method presents in other areas. Lessons learned from this experience are shared with educators interested in applying the Jigsaw method in software engineering-related courses.

KEYWORDS

design patterns, software engineering education, jigsaw, active learning, experience

1 Introduction

In the foreseeable future, no significant shift is expected in the core skill set required of software engineering (SE) professionals, even with the increasing integration of advanced artificial intelligence models into software development practices [8]. Foundational knowledge in requirements analysis, software design, and testing continues to be essential to form future professionals [17]. This scenario is supported by Akdur [1], who surveyed 628 experienced software practitioners and investigated areas of knowledge that are most frequently applied in daily software development tasks. The results revealed that programming, software configuration management, requirements engineering, and software design are the four most commonly used areas in practice [1]. At the same time, considering those SE areas, there remains a gap between academic curricula and the practical demands of the software industry, particularly in software design [17]. Addressing these disparities will better prepare graduates for the realities of professional software development.

Specifically, design patterns serve as proven, reusable solutions to common problems encountered in software design [15]. They encapsulate best practices and facilitate communication among developers by providing a shared vocabulary [9]. Their relevance extends beyond software design, influencing and enhancing practices in the other most-required knowledge areas.

In programming, design patterns can contribute directly to code quality and maintainability [14, 35]. Patterns such as Strategy, Observer, and Factory Method promote flexibility, encapsulation, and reuse, enabling developers to write cleaner and more adaptable code. Understanding design patterns allows programmers to anticipate changes, reduce code duplication, and align their implementation with established architectural practices [9, 15].

In software configuration management, design patterns can support modular and decoupled design, which is critical for managing source code across teams, branches, and versions. For example, patterns that encourage the separation of concerns simplify version control by minimizing merge conflicts and isolating changes to specific components. This enhances traceability and simplifies rollback and deployment processes. Moreover, identifying the need to apply a pattern during code refactoring can improve code readability and maintainability, contributing to the code quality and software evolution [23].

In requirements engineering, design patterns can help bridge the gap between requirements and implementation by offering predefined solutions that can be mapped to functional and non-functional requirements. When requirements indicate the need for extensibility, responsiveness, or dynamic behavior, certain patterns can be proposed during the design phase to meet those criteria effectively. Moreover, using design patterns can improve communication with stakeholders by making design decisions more transparent and justifiable.

Considering its relevance for software development practice, software design patterns have been a mandatory component in the curriculum of computer science-related undergraduate courses as proposed by ACM/IEEE [24] and SBC [37] academic societies. By teaching this content to future software engineers, they are expected to obtain the skills to define well-designed software systems, assess best design solutions for a problem, and maintain, evolve, or refactor software in production [37].

Although software design patterns are a well-established and demanding topic in most computer science curricula, their instruction has not received adequate attention. From the students' perspective, software design, particularly the comprehension and application of design patterns, ranks among the most difficult subjects encountered during undergraduate studies [29]. From the educators' standpoint, teaching this content presents significant challenges, especially in student engagement and in the development of instructional strategies [30]. Therefore, switching from traditional

lecture classes and exploring different instructional methodologies is necessary to better teach design patterns in undergraduate courses.

From another perspective, the Jigsaw method is a well-established cooperative and active learning strategy that has been employed for over four decades across various educational levels. It is categorized as an active learning approach because it places students at the center of the learning process, encouraging them to take responsibility for their own knowledge acquisition. Furthermore, it promotes cooperative learning, as students collaboratively construct knowledge on specific topics through peer interaction. At the cognitive level, the Jigsaw method has been shown to enhance students' retention, comprehension, critical thinking, cognitive processing, and memory, while also mitigating knowledge loss [27]. In addition to these cognitive benefits, the method contributes to the development of affective and psychomotor skills, fostering improvements in communication and teaching abilities [40]. The Jigsaw classroom has been effectively implemented in a wide range of disciplines, including medicine, physics, mathematics, and the humanities. It has received high levels of acceptance from both students and instructors, who have recognized its effectiveness in promoting both hard and soft skill development. [11].

This study presents an account of the experience of designing and implementing a Jigsaw classroom for teaching design patterns in a computer science undergraduate course. The remainder of this paper is organized as follows. Section 2 provides the background on design patterns, the Jigsaw classroom learning method, and its application in software engineering teaching. Section 3 describes related experience reports on teaching design patterns. Section 4 presents the Jigsaw activity design and intended learning objectives of its application as an instructional method for teaching design patterns. Results of this cooperative activity are reported in Section 5 and the lessons learned of it are described in Section 6. Finally, the conclusion and future work are outlined in Section 7.

2 Background

2.1 Software design patterns

At their essence, software design patterns represent formalized and reusable solutions to recurring design challenges encountered within specific software development contexts [15]. They are not concrete implementations but abstract blueprints or templates that capture proven architectural insights and best practices cultivated through extensive practical application. These patterns establish a shared lexicon and conceptual framework among software engineers, thereby facilitating more precise and effective communication and collaboration [9]. Applying design patterns within the software development life-cycle yields significant advantages contributing to software systems' quality, code reusability and maintainability, and team communication and efficiency.

Design patterns are traditionally categorized into three primary groups [15]: creational, structural, and behavioral. Each category addresses a distinct aspect of object-oriented design and serves a specific architectural concerns. In short, **creational patterns** abstract the process of object instantiation, allowing systems to be decoupled from the concrete classes they utilize. This enhances design flexibility and scalability by promoting the use of interfaces over concrete implementations. Some examples of creational patterns are [15, 35]: singleton, factory method, abstract factory, builder, and prototype. **Structural patterns** address the composition of classes and objects to form larger, more complex structures. They define efficient and flexible relationships among entities. Patterns in this category are [14, 15, 35]: adapter, bridge, composite, decorator, facade, flyweight, and proxy. Finally, **behavioral patterns** are concerned with the assignment of responsibilities among interacting objects. They enable effective communication and control flow within a system by encapsulating complex behavior. Examples of these patterns are [14, 15, 35]: observer, strategy, command, chain of responsibility, mediator, state, template method, iterator, and visitor.

2.2 The Jigsaw classroom

Jigsaw is a cooperative learning method created in 1978 by Aronson et al. [4] as an alternative to conventional teaching methods. Jigsaw was influenced by the social constructivist theory, which emphasizes the significance of learners' learning when they construct their knowledge through interaction, collaboration, and group work [2].

In Jigsaw, the classroom organization is centered on the students, rather than the teacher. Jigsaw promotes collaborative learning by forming smaller, interdependent groups. Each student is assigned a specific segment of the overall topic, which they study individually. Upon completion, the group members integrate their respective contributions to construct a comprehensive and cohesive understanding of the subject matter, thereby assembling the pieces of a jigsaw puzzle [4, 5].

Figure 1 illustrates a basic Jigsaw classroom design. Students are initially divided into small (4 to 6 members) heterogeneous "expert groups", with members assigned a subtopic related to the broader subject under study. Following this, students leave their home groups and join "teaching groups," composed of peers from other home groups with different but related subtopics under their responsibility. Within the expert group, each student teaches their colleagues the subtopic he/she know. All students in the teaching group collaborate to analyze, discuss, and master all the taught content. This structure ensures that all students contribute to the group's collective understanding while deepening their knowledge through teaching and peer interaction.

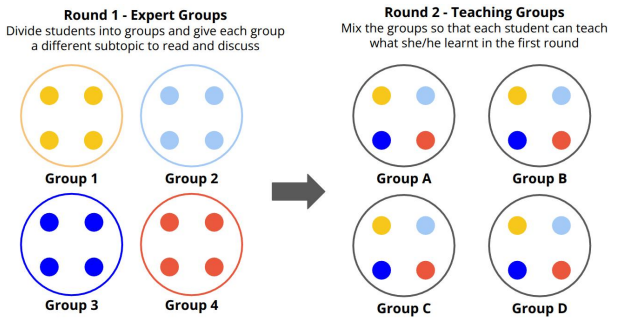

Figure 1: Groups dynamics in the Jigsaw classroom.

Recent studies show that Jigsaw improves academic performance [27], by improving knowledge retention and reducing the decline

effects on their learning. Another positive impact of Jigsaw is that students enhance their learning, comprehension, critical thinking, cognition, and memory due to the cognitive effort required to understand a topic and explain it to their group's colleagues [2, 27, 36, 40].

Considering psychosocial skills, the Jigsaw method has beneficial effects on students' engagement, learning motivation, self-efficacy (i.e., feeling of competence), self-esteem, communication skills, overcoming shyness and hesitation in class [11, 13, 25, 40]. Motivation is increasing in this kind of activity, as students need to contribute to forming the knowledge for their group. Being part of a group with a shared purpose enhances students' engagement in the activity. Students feel competent when their colleagues understand his/her explanation about the topic [12].

2.3 Jigsaw in computer sciences and software engineering education

Recent research [40] reported that most contributions about the application of the Jigsaw approach has been obtained from instructions conducted in areas of medicine, physics, and mathematics.

To understand the evidence of applying the Jigsaw method in computer science courses, and specifically in software engineering, a search was conducted on academic databases, Scopus, and Google Scholar. For this, the following string was used: *(TITLE (jigsaw) AND ALL (("software engineering" OR "computer sciences") AND (education OR teaching OR learning))).*

Based on the results of database searches, and to the best of our knowledge, the Jigsaw method has been barely applied as a learning method in computer sciences or software engineering classes. This cooperative learning strategy has been reported to teach topics of programming [10], computational thinking [34, 42], class diagrams [31, 33], and software functional size through function points measurements [32].

Studies in programming and computational thinking have reported better results in terms of students' performance and engagement when using the Jigsaw method compared to other approaches, such as traditional teacher-centered lectures, unplugged activities, or traditional group activities [10, 34, 42].

The studies in [32, 33] reported that students who initially held misconceptions about the usefulness and importance of a subject (i.e., class diagrams or function points) demonstrated improved performance after completing the cooperative learning activity. Moreover, the same author of [32, 33] conducted an additional study to understand the knowledge retention of students after a semester. The results of a post knowledge test demonstrated that students who participated in the Jigsaw activity six months prior performed better than students who learned the same content in a traditional lecture class [31].

Related work identifies several limitations of the Jigsaw teaching approach in computer science courses: (i) the effectiveness of the method relies heavily on student accountability, which may lie beyond the instructor's control [10]; (ii) implementing the Jigsaw strategy requires a substantial investment of the instructor's time for careful planning, including determining appropriate group sizes, identifying subtopics, and assigning topics to students [33]; (iii) depending on the class size, the participation of teaching assistants may be necessary to effectively monitor group dynamics [34]; and (iv) depending on the scope of the subject matter, students may feel overwhelmed by the responsibility of both learning and teaching complex content. Consequently, instructors may encounter challenges in dividing the main topic into sufficiently granular and manageable subtopics for student distribution [42].

Despite these challenges, the Jigsaw method is still regarded as a novel instructional strategy in computer science education, particularly in software engineering (SE) courses. It is therefore essential to collect and analyze practical experiences with this approach in order to better understand its potential benefits and limitations for SE instruction.

3 Teaching of software design patterns

Jiménez-Díaz et al. [22] propose a pedagogical methodology for teaching design patterns that emphasizes active learning by involving students in structured role-playing sessions designed to facilitate software refactoring practices. As a result, the authors concluded that theoretical preparation is essential before the role-play sessions, as it helps students grasp design problems and context. Therefore, such preparation makes later role-play sessions more meaningful and effective.

Jeremic et al. [21] propose a learning environment that integrates diverse educational systems for teaching software design patterns. The teacher can use the platform to set different contexts and scenarios for pattern application. The platform also helps educators measure students' understanding of design patterns based on the patterns selected in a specific situation.

Intelligent tutors had also been used for teaching software design patterns. The intelligent tutor proposed in [6] observes students' coding work and detects possible snippets of code where a pattern could be applied. The tutor suggests to the student the application of the pattern and explains the rationale behind its application. This tutor aims to guide novice students to understand why and how diverse design patterns can be employed.

In [38], educators reported the use of the Just-in-Time Teaching (JiTT) method to teach software design patterns in two classrooms with a total of 130 students. The authors noted that the application of JiTT had a significant impact on teaching and learning software design patterns. As a matter of clarification, the JiTT, originally proposed by Novak et al. [28], is a teaching strategy in which instructors pose questions to students about pre-class work, such as readings or videos. Students respond to these questions before class, and instructors review their answers to identify important themes to be addressed in in-class activities, making the face-to-face time more focused on students' comprehension and areas of confusion.

Intending to teach the application of software design patterns in various types of systems, some educators had challenged the students to apply their software design knowledge in diverse contexts. For instance, [19] uses the context of a game development course to engage and motivate students to understand patterns implemented in a toy example game. Therefore, students must apply design patterns in a new game project they have idealized. Another example is the work in [41], where students designed object-oriented embedded systems using the Raspberry Pi as a platform for their projects. The report in [41] highlights the interesting results of this kind of

challenge, particularly regarding the increase in team dynamics and students' creativity and engagement when using the Raspberry Pi platform and working with sensors and actuators.

Similarly, in [20], a learning model was proposed to assist novice developers in understanding and selecting the most suitable design patterns for a specific problem. The model guides the developer through a sequence of three phases, comprising the identification of the design strategy, the specification of the design scope, and the recognition of the design intention. By following these phases, the developer can select a suitable design pattern for their problem, while also acquiring the skills necessary to understand patterns and make their own autonomous decisions.

More recently, in [26], the authors experimented with 60 students to evidence the improvement of software design patterns learning by using serious video games. The authors perceived improvements in retention and understanding of the topic, and in motivation and satisfaction from learning with the video game.

Finally, in [18], the author evaluates the use of ChatGPT by students to learn software design patterns. He analyzed the students' interaction with ChatGPT as a tutor. He focused on the questions the students asked about it while studying this topic. Additionally, students' learning was assessed through an exam to determine whether they comprehend and retain the theory, using ChatGPT as a tutor. As a result, the author recognized the value of ChatGPT as a learning tool. However, it requires providing students with structured guidance on how to formulate questions that encourage critical thinking about software design patterns in real-world scenarios.

Despite the growing advocacy for active and cooperative learning in computer science education, there remains a noticeable scarcity of documented experiences regarding the use of such methodologies in the specific context of teaching software engineering courses, and specifically software design patterns. While the pedagogical literature increasingly emphasizes student-centered approaches, there is limited empirical or practical evidence demonstrating how constructivist methods, like the Jigsaw classroom, can be effectively integrated into the teaching of this topic. This study contributes to filling this gap by reporting on the use of the Jigsaw method by highlighting both the benefits and limitations encountered, as well as lessons learned during implementation. This work offers valuable insights that can support other instructors in adapting and applying the method in their own teaching contexts.

4 Activity Design

The primary objective of this activity was to effectively teach software design patterns to 59 third-year students enrolled in the Information Systems program. In planning the activity, the following constraints were taken into account: (i) the activity should not exceed two in-class sessions, each lasting 1.7 hours; (ii) the majority of students are employed full-time and attend classes after their work; and (iii) the class is held during evening hours.

4.1 Learning objectives

Educational goals were defined in the cognitive, affective, and psychomotor domains.

4.1.1 Cognitive goals. The revised Bloom's taxonomy [3] was used as a frame for the cognitive domain. The cognitive goals are presented as follows:

LG01 - Analyze: The students distinguish important from unimportant parts in the material that presents a design pattern.

LG02 - Understand: The students comprehend and construct together the main knowledge (e.g., context, purpose, problem, solution, application, and trade-offs) about nine software design patterns: singleton, builder, prototype, adapter, proxy, facade, mediator, observer, and chain of responsibility. Additionally, the students compare design patterns in the same and in different pattern categories: structural, behavioral, and creational.

LG03 - Recall: The students retrieve relevant design patterns knowledge from long-term memory.

4.1.2 Affective goals. Considering the affective domain, the learning objectives were:

LG04 - Receive: The students learn through the explanations brought by their colleagues about a new design pattern.

LG05 - Respond: The students react to the new information they received from their colleagues, showing involvement.

4.1.3 Psychomotor goals. Finally, for the psychomotor domain, the established educational objectives were:

LG06 - Precision: The students explain design patterns with accuracy and control.

4.2 Jigsaw activity design

The activity consisted of four rounds: individual studies of a design pattern, group sessions of 'one pattern experts', group sessions of 'teaching patterns in one category', and a final 'teaching all-categories' group session. Each round is explained below.

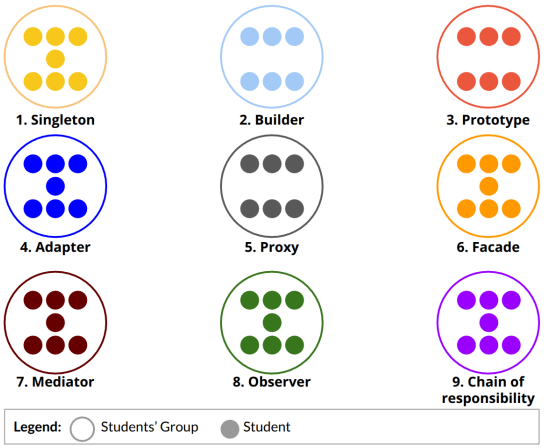

Figure 2: Expert group session. Students grouped by pattern in the in-person class session.

Individual studies. Fifty-nine (59) students were enrolled in the class. The teacher assigned each student a design pattern one week before the in-class session. The selected design patterns were: Singleton (7 students), Builder (6 students), Prototype (6 students),

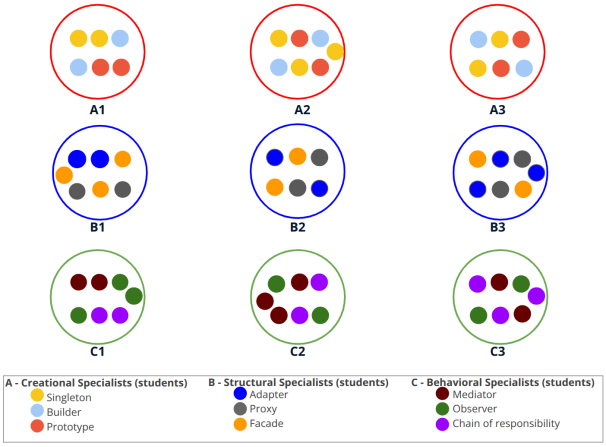

Figure 3: First teaching group session. Students groups discuss patterns in the same category in the in-person class session.

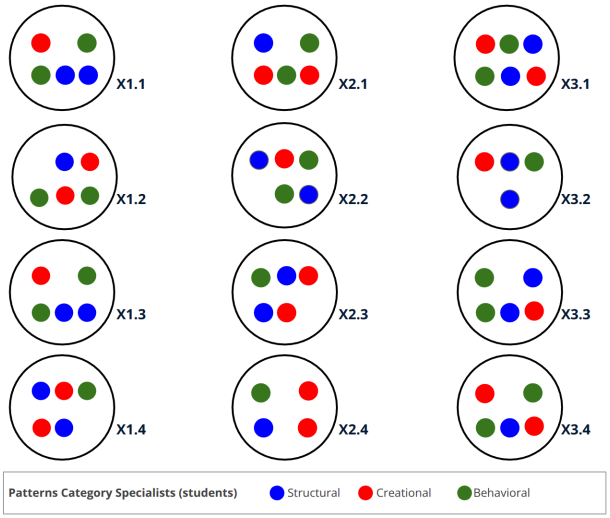

Figure 4: Second teaching group session. Student groups discuss patterns in all categories in the in-person class session.

Adapter (7 students), Proxy (6 students), Facade (7 students), Mediator (7 students), Observer (7 students), and Chain of responsibility (6 students).Those nine patterns were selected from the twenty-one patterns proposed in [15], the reason for that is the teacher considered them sufficient to achieve the learning objectives intended for the Jigsaw activity. From the teacher's perspective, focusing on a selected subset of design patterns is more efficient than briefly covering all of them, especially given the limited time available to explore this topic.

Students were advised that they would explain the pattern to colleagues in the in-class session. For that, students should prepare a 10-minute oral presentation. The students were also oriented to study the designated design pattern before the in-class session. The students should consider, for instance, the pattern's purpose, the problem it solves, the proposed solution, the application use cases, the benefits and drawbacks of using such a pattern, and code implementation examples. For this, the design patterns catalog available in [35] and the design patterns chapter in [39] were recommended.

Artifact: As a result of the individual studies, the students sent to the teacher an individual report before starting the in-class session.

Expert group session. In a second moment, with the students in the classroom, the educator explained the Jigsaw activity and what was expected of them. After this, the students were divided into expert groups, i.e., those who prepared the same pattern. Figure 2 illustrates the nine groups of experts by pattern and the expected number of students in each group.

In this 40-minute session, expert groups were asked to discuss the studied pattern, socializing their understanding. The idea was that the shared information would construct a more robust knowledge of the pattern. In case some students had doubts or lack knowledge, the colleagues, as a group, could fill content gaps. The teacher also solved additional doubts.

Artifact: The group should prepare a unified material for teaching the pattern to colleagues in other groups. All individuals in the expert group should be confident in explaining the content to other groups colleagues.

First teaching group session. On the same class day, new groups of students were composed. On this occasion, groups were formed to discuss design patterns by category. This session last 40 minutes. Figure 3 shows organized groups for categories of organizational, structural, and behavioral patterns. As illustrated, three groups were formed for each category. At least two representatives of the pattern expert groups participated in a category group.

Students experts in a pattern explained the patterns to their colleagues. For instance, in groups A1 - A3, students taught and discuss singleton, builder, and prototype creational patterns.

After reviewing the three patterns, the group discussed and compared patterns in the same category, for instance by discussing patterns' use cases, similarities, differences, benefits, and drawbacks.

Artifact: The students' group prepared a document with a comparative table summarizing the results of such discussions.

Second teaching group session. This last session was performed in another class day. All groups shared the produced material in a shared directory to be accessible to all students. New groups were formed. On this moment, members came from different categories. For instance, in Figure 4, the group X1.1 was formed by past members of groups A1, B1, and C1 (See Figure 2). The main task was to share their knowledge about pattern comparison in each category.

Artifact: The students' groups produced a document with the comparisons between patterns in a category and a discussion about pattern categories, e.g., the meaning of a pattern being creational, structural, or behavioral, and the importance of each category.

4.3 Learning assessment

One week after the activity, the students conducted a test. This test aimed to measure their cognition of the design patterns worked on in the Jigsaw activity. The assessment comprised five single-choice questions, each worth two points. The maximum score was ten

points. Moreover, a self-assessment questionnaire was administered to students to understand the affective and psychomotor skills they trained during the Jigsaw classroom.

5 Activity Results

This section analyses the accomplishment of cognitive, affective, and psychomotor learning objectives that outlined the Jigsaw activity.

5.1 Cognitive goals

As defined by Anderson and Krathwohl in [3], cognitive learning objectives focus on developing students' intellectual, critical thinking, and problem-solving skills. In the proposed Jigsaw activity, the next cognitive learning objectives were intended: analyzing, understanding, and remembering.

5.1.1 Analyzing. This skill was exercised by the students in individual studies. In this moment, students prepared individual patterns, a task they performed before the in-person session. The teacher asked to students sending a report summarizing the knowledge they obtained from individual research. Students had one week to study and prepare the individual report. Additionally, students were asked to add a reference list for the material used in their research.

This activity was thought to encourage the development of the analysis level of Bloom's Taxonomy[3] by requiring students to engage critically with information related to a specific design pattern. In order to complete the task, students searched for and identified relevant sources (e.g., books, articles, websites, videos, etc.), extracted essential concepts, and differentiated between core structural elements and contextual applications of the pattern. This process involves decomposing information into meaningful knowledge components and understanding the relationships between them, which are considered central aspects of analytical thinking [3]. Furthermore, by synthesizing the gathered content into a coherent report, students organized and structured the material reflecting their understanding of the pattern.

Figure 5 presents the knowledge components about design patterns that students considered relevant to include in the individual report. From the 59 students enrolled in the course, 50 of them delivered the report. All students (n=50) included in their report information about the purpose, solution, application, benefits, and drawbacks of using a specific design pattern. 90% (45/50) described the problem that a pattern solves. 80% (40/50) considered relevant to report the application of the pattern and possible use cases. In a less proportion, 30% (15/50) of students included examples of patterns implementation in code. Only 10% (5/50) of the students included information about relationships with other design patterns, examples of patterns use in realistic software systems, and best practices to be considered at using and implementing a pattern. No student considered relevant to explain the pattern through UML's class diagrams, even when in well-known literature and famous websites, such diagrams are commonly presented.

5.1.2 Understanding. This cognitive skill was exercised throughout all three group sessions: the pattern specialist session, the category specialist session, and the all-categories specialist session. It is noteworthy that 50 out of 59 students were present during these activities.

During the expert group session, students engaged in discussions focused on a single design pattern. The objective was to share and complement the knowledge they acquired during their studies. This collaborative exchange enabled students to address gaps in their understanding of the designated design pattern. As a result, each student could supplement their individual report with information obtained from their peers.

Students were instructed to develop comparative charts as an outcome of the first and second teaching group sessions. During the first teaching groups session, each group created a chart that compared design patterns within a single category. For example, groups focusing on creational patterns produced comparisons among the Singleton, Builder, and Prototype patterns. In the second teaching groups session, students expanded their analysis to include comparisons across all nine patters from the three categories, namely, creational, structural, and behavioral.

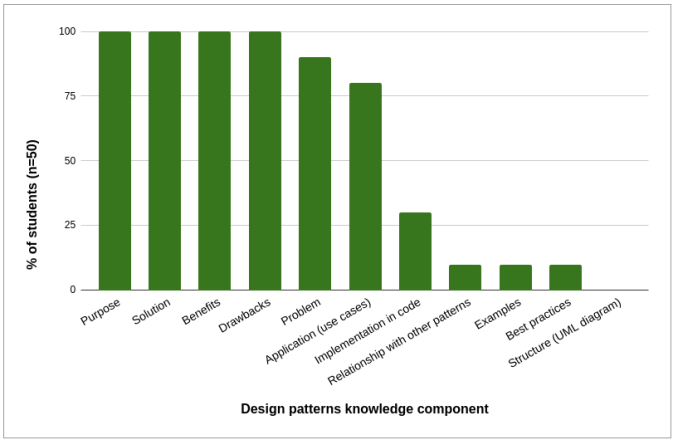

Figure 5: Students that reporting a specific knowledge component about patterns in individual reports.

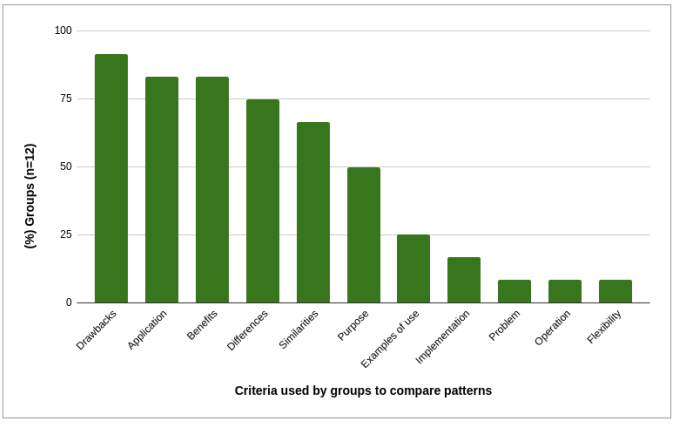

Figure 6: Criteria used by groups to compare design patterns.

Figure 6 shows the criteria that students' groups selected to analyze and compare design patterns in the same category and between categories. Twelve groups participated in the group sessions in total (See groups settings in Figure 4). 92% (11/12) of the groups considered pattern drawbacks as important comparison criteria. The application and benefits were reported by 83% (10/12) of groups for comparing patterns. Differences and similarities between patterns were also considered as comparison criteria by 75% (9/12) and 67% (8/12) of groups, respectively. The purpose of the pattern was addressed by 50% of the groups. Other criteria such as examples of use, implementation, problem, operation, and flexibility of patterns were used as comparative baseline by 25% (3/12) of the groups.

The cognitive level of understanding was incentivized among students in the following way. Students engaged in peer discussions in each group session and collaboratively produced summaries or comparative charts of design patterns. During peer discussions, students either explained the design patterns they had studied or learned from their peers. This required them to organize their thoughts, communicate using accessible language, and respond to their colleagues' questions. Such activities foster cognitive engagement, as students were not merely recalling memorized content but also interpreting and articulating their understanding to share it with others.

5.1.3 Recalling. To understand the students' recall of design patterns, the test in Table 1 was executed ten days after the Jigsaw activity concluded. No previous advice about the test was gave to students. A total of 50 students individually answered the test and all of them participated of the Jigsaw classroom sessions. The test was executed in an in-person class.

The resultant grades for n=50 students were: average grade = 9/10 points, mean grade = 10/10 points, mode = 10/10 points, and stdv = 1.27. Only one student did not pass the test with a grade of 4 points. Overall, it was possible to conclude that students' recall was satisfactory, as shown in Figure 7.

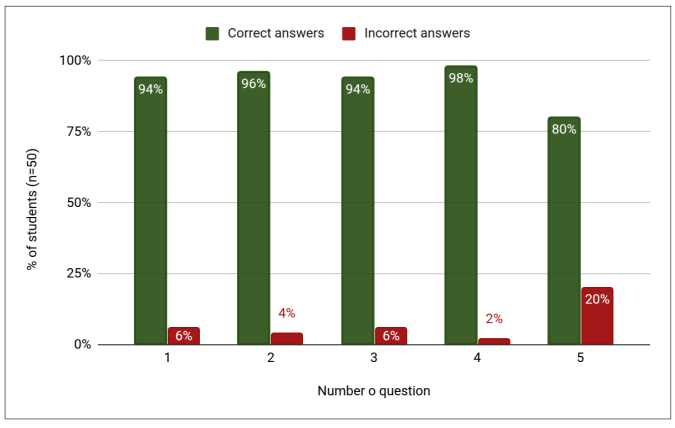

Figure 7: Students' grades in the post-Jigsaw activity test.

For the test question #1, 94% (47/50) of the students were able to identify the purpose of each pattern category. Three students confused (interchanged) the goals of behavioral and structural patterns.

Identifying a pattern belonging to a specific category was also well-recalled by the students, since 96% (48/50) of them correctly answered question #2. Two students got this question wrong, associating the Adapter or the Facade pattern with software behavior, but not with its structure.

Additionally, from question #3, 94% (47/50) of the students correctly remembered implementation details about the Singleton pattern. Few students (3/50) were confused about implementing this pattern using multiple inheritance or a normal class constructor that allows creating multiple instances of objects.

The functioning of the Facade pattern was correctly recalled by 98% (49/50) of the students. The student who failed the question #4 related this pattern with creating a unique object instance (i.e., the purpose of the Singleton pattern).

Finally, question #5 assessed the capacity of the students to understand the applicability of a specific pattern in an unseen context. By presenting an unknown situation to the student, he/she must understand the problem and select, from the known patterns, the best option to solve such a problem. This question exercises not only the students' memory (recall) but also their analytical skills for decision-making. 80% (40/50) of the students answered it correctly, i.e., the most adequate pattern for the problem is the Observer. 14% (7/50) related the new software context to the Chain of Responsibility pattern, while 6% of the students answered that the best option for the problem was the Mediator pattern.

5.2 Affective and psychomotor goals

Following the Jigsaw activity, a self-assessment questionnaire was administered to the students. The purpose of this questionnaire was to gather data on students' perceptions of their own performance, both during the individual preparation phase prior to class and throughout the collaborative group interactions during the in-class session. The questionnaire included the following five statements:

- ***(Q1)*** The preparation of the design pattern under my responsibility prior to the classroom activity was highly satisfactory;
- ***(Q2)*** I felt very confident when explaining my assigned pattern to my peers;
- ***(Q3)*** The participation of my group mates was highly satisfactory;
- ***(Q4)*** I believe I learned a great deal about design patterns I was previously unfamiliar with through my peers' explanations; and
- ***(Q5)*** I greatly appreciated this type of activity (Jigsaw) for learning design patterns and prefer it over traditional lectures.

Students answered each affirmation following the Likert scale: *1 - Totally disagree, 2 - Disagree, 3 - I do not know, 4 - Agree, and 5 - Strongly agree.*

Participation in the questionnaire was voluntary and did not impact students' grades. Twenty-two out of 50 students who participated in the Jigsaw activity completed the questionnaire. Results are depicted in Figure 8.

Regarding affective goals, our aim was to assess students' openness to learning new content from their peers, specifically, design patterns they had not studied prior to class. Based on the responses

Table 1: GoF Design Patterns - Single Choice Quiz

Question	Description	Answer Options	Rationale
1	The GoF design patterns are grouped into three categories: creational, structural, and behavioral. Which of the options below correctly describes these categories?	A) Creational patterns deal with code organization, structural patterns handle object creation, and behavioral patterns define how classes are organized into packages. **B)** Creational patterns deal with object instantiation, structural patterns handle the composition of classes and objects, and behavioral patterns define the interaction and communication between objects. C) Creational patterns define how objects interact, structural patterns deal with multiple inheritance, and behavioral patterns assist with code reuse. D) Creational patterns are related to algorithm execution, structural patterns handle database modeling, and behavioral patterns deal with data persistence.	This question checks if students understand the fundamental classification of GoF patterns and can distinguish their purposes accurately.
2	Which of the following options corresponds to a behavioral pattern according to the GoF design pattern classification?	A) Singleton B) Adapter **C)** Mediator D) Facade	This question assessed students' ability to classify specific patterns correctly within the GoF categories.
3	Which of the following characteristics is essential for correctly implementing the Singleton design pattern?	A) The use of inheritance to allow multiple controlled instances of the class. B) A public constructor to allow the creation of multiple instances whenever needed. **C)** A static method that returns a unique instance of the class and a private constructor to restrict the creation of new objects. D) The creation of a new object every time the instance is requested, ensuring independence between calls.	This question evaluates understanding of a pattern's core implementation mechanism and purpose.
4	The Facade design pattern is used to facilitate interaction with complex systems. Which of the following best describes the purpose and operation of this pattern?	A) The Facade encapsulates object creation to ensure that only one instance is created and reused throughout the system. **B)** The Facade provides a simplified interface to a set of classes and subsystems, reducing complexity and making usage easier. C) The Facade allows new behaviors to be added dynamically to objects without modifying their original structure. D) The Facade defines a family of interchangeable algorithms that can be selected dynamically at runtime.	This question checks whether students can identify the intent and usage of the pattern within a system architecture.
5	In an online multiplayer game, there is a notification system that tracks player status. Whenever a player joins or leaves a match, several parts of the system need to be updated automatically, such as: - The scoreboard, which must display the list of active players. - The mini-map, which must add or remove the player's position. - The chat system, which displays messages about player activity. Given this scenario, which behavioral design pattern is being applied?	**A)** Observer B) Mediator C) Chain of Responsibility D) No behavioral pattern can be used	This scenario-based question tests the students' ability to map real-world behavior to a pattern.

to questions Q3 and Q4, 95% of the students agree or strongly agreed that their classmates' explanations of the other patterns were satisfactory. Furthermore, 90% of the students agreed that they were able to learn effectively through the insights shared by their group members.

With regard to the psychomotor aspect, responses to question Q1 indicate that the majority of students (85%) actively engaged in studying the design pattern for which they were responsible. In addition to completing the individual report assigned by the instructor, some students also prepared slide presentations to support their explanations to group members. The remaining 15% reported feeling unsatisfied with their preparation prior to the in-class session. Furthermore, by Q2, 90% of the students expressed a high level of confidence in sharing their knowledge with peers on multiple occasions.

5.3 Students' opinions on the Jigsaw classroom

When asked, in Q5, about their perception of the Jigsaw activity for learning design patterns, 85% of the students agreed that this method was more effective than traditional lecture-based classes. Meanwhile, 10% were unsure how to respond, and 5% expressed a preference for lectures over cooperative learning approaches such as Jigsaw.

The questionnaire also included an open-ended section in which students could share their thoughts on the experience. According to several responses, the most motivating aspect of the activity was either explaining a pattern to peers or engaging with classmates to gain a better understanding of the other patterns.

"I felt that the coolest part of the dynamic was the teaching aspect. Even though this impression might come from a personal preference, I noticed that the topics I retained best were the ones I actively engaged with — either by teaching or by responding to more interactive explanations that some classmates prepared."

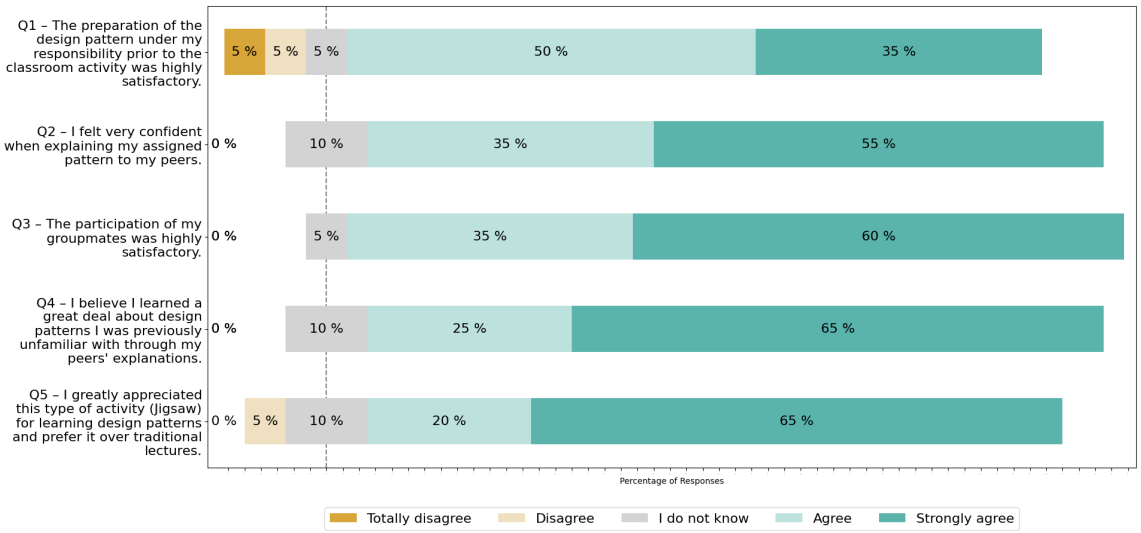

Figure 8: Students' self-assessment after the Jigsaw activity.

Students also shared suggestions for improving future Jigsaw activities, focusing mainly on issues caused by student absences. Some key areas for improvement include:

"I found this type of activity interesting, but I think it is necessary some strategy to encourage more in-person participation from the class."

"I liked the format, but the downside is that there's a chance we might miss out on some content if our classmates are absent from class (which isn't uncommon)."

"I liked the activity, but unfortunately some people from my (pattern) group were absent."

Other comments pointed to a lack of time for discussing the patterns in more detail. For example:

"I believe there was a lack of time for the second part of the activity. I felt we had to rush through the explanations, which affected the time for questions or clearing up doubts."

6 Discussion

Educators interested in applying the Jigsaw classroom in software engineering courses could consider the following perceived benefits and drawbacks, and lessons learned from this experience.

6.1 Perceived benefits

The Jigsaw method is a cooperative learning strategy that fits well for large student groups (over 40) [11, 40]. In addition to the Jigsaw's benefits reported by educational researchers [2, 11–13, 25, 27, 36, 40], and presented in Section 2, observations from this experience confirmed the high level of student engagement during group sessions, with only a small number of students (2 out of 50) adopting a more passive role in discussions.

As reported in Section 5, the Jigsaw method proved to be a practical instructional approach for fostering cognitive, affective, and psychomotor skills across the entire class. Furthermore, based on the author's experience and in comparison with previous applications of active learning strategies in software engineering education [7], the Jigsaw approach revealed latent student abilities that had not been identified through other methods. For example, students' body language often conveyed a sense of satisfaction when peers actively engaged with the material they presented. Several students also expressed to the instructor their appreciation for the opportunity to teach their classmates, noting that they felt confident when responding to questions.

6.2 Perceived limitations

The main limitation of using Jigsaw as a learning method for software design patterns is related to the scope of Bloom's cognitive levels [3] explored with it. As designed in this experience, only the remembering, understanding, and analysis levels could be exercised.

When considering higher cognitive levels, e.g., applying, evaluating, and creating, the Jigsaw design and application increases in complexity. In the case the educator consider such higher levels as essential learning objectives for design patterns teaching, other active learning methods could better option, for instance, problem-based or project-based learning. Through those approaches the instructor can incentivize students to deal with patterns in different real or realistic software projects, for instance, by coding using patterns, refactoring code by applying patterns, or selecting patterns to solve a diversity of real problems.

Additionally, we can consider that design patterns is not a subject to study isolated in a specific course. Therefore, patterns can be taught (reinforced) in different courses of the undergraduate curriculum, for instance, in courses of object-oriented programming, object-oriented analysis and design, software architecture, maintenance, refactoring, configuration management, software quality, and integrated projects. Those courses are commonly spread over undergraduate courses curriculum, then, enforcing this subject in

diverse courses could improve the effectiveness of students patterns learning.

6.3 Lessons Learned

Lesson #1: Division of topics and individual allocation of contents to students. The subject or topic selected can be naturally broken into distinct subtopics. Each student must get a unique subtopic for individual preparation. This will facilitate the junction and relationship of the jigsaw pieces for the students.

The allocation of subtopics to students must be fair. Each subtopic must be the same (or very similar) complexity for all students and must be adequate to their abilities.

Lesson #2: Contingency plan for students' absenteeism. Student absenteeism in in-person classroom sessions is a common occurrence and must be anticipated when designing Jigsaw activities. Instructors should develop contingency plans to address potential disruptions caused by the absence of expert group members. One effective strategy involves assigning the same subtopic to multiple students within the expert groups. For example, if the instructor intends to form four teaching groups, the same subtopic can be assigned to six students within the expert groups. This redundancy ensures that, even in the case of student absences, the subtopic can still be adequately covered by the remaining expert group members. Additionally, instructors may mitigate the impact of absences by sharing the learning artifacts produced during group sessions, such as slides, summaries, or reports, with the entire class, thereby reinforcing the material and promoting continuity in the learning process.

Lesson #3: Continuous observation of group members' interaction The instructor is required to circulate and observe group interactions, mainly to clarify doubts, support teams on discussion management, mainly to avoid that one student monopolizes the discussion, to support disagreement resolutions, or incentivize quiet students to participate.

Lesson #4: Deal with unprepared students The instructor must incentivize the students' accountability for preparing the individual topic, highlighting their role as 'specialists' and the importance of their participation in contributing to their colleagues' learning. Moreover, it is important to suggest to them basic studying material (e.g., book chapters, websites, etc.) for preparing the subtopic. Another strategy is to allow the specialist in the same subtopic to share their reports and discuss the subject to fill gaps in their knowledge. Finally, a grade can be provided to individual reports to increase the students' dedication to preparing the material for teaching the subtopic under their responsibility.

Lesson #5: Time control Controlling time during group discussions in a Jigsaw activity helps to keep the activity focused and complete all planned group sessions. The total session time must consider the time for each student's intervention, new group formation (moving from one group to another), doubt resolution, reflection, summarization, artifact production, and resolution of unforeseen situations. Ideally, in each group, roles can be assigned to students, for instance, for taking notes, time keeping, intervention orchestrating, or communicating with the instructor.

Lesson #6: Prepare materials previously: The instructor must produce artifact templates for students to use during discussion sessions. Ideally, the material can be physically printed to avoid students' distractions by using notebooks, tablets, or smartphones. Moreover, it is important to explain to students the Jigsaw dynamics before starting the activity, as this will avoid questions not related to the subject being learn.

Lesson #7: Select the right physical space for group formation Students must be able to move through the physical space without problems. Set the initial space for groups, explaining where each student must sit in the different group sessions. This can be done in a projected slide with the list of group members visible to all students. Additionally, the physical space should be selected with the aim of preventing groups from disturbing each other during parallel discussions.

7 Conclusion

The application of the Jigsaw method in software engineering education remains relatively underexplored. This report offers educators in this area a case study to inspire their instructional design based on Jigsaw.

Beyond design patterns, the Jigsaw method seems practical for teaching other software engineering topics that can be decomposed into subtopics and benefit from group discussions. The Jigsaw's collaborative structure can allow a more profound understanding through active student engagement and knowledge collaborative construction.

While the jigsaw approach supports the development of foundational knowledge and collaborative skills, higher-order cognitive objectives in teaching software design patterns may require complementary instructional strategies. For instance, project-based Learning (PjBL) can be particularly effective when integrated into software maintenance courses, where students evaluate and apply design patterns during refactoring activities [22, 23]. Such pedagogical combinations may better support critical thinking, synthesis, and practical application skills, ultimately enhancing the overall learning experience in software engineering education.

ARTIFACT AVAILABILITY

Artifacts related to data collection and analysis used in this study are available in [16].

ACKNOWLEDGMENTS

The author gratefully acknowledges the support of the Brazilian National Council for Scientific and Technological Development (CNPq) (Grant number: 440425/2024-7), and the PRPI-USP (Grant number: 22.1.09345.01.2).

REFERENCES

[1] Deniz Akdur. 2022. Analysis of Software Engineering Skills Gap in the Industry. *ACM Trans. Comput. Educ.* 23, 1, Article 16 (Dec. 2022), 28 pages. https://doi.org/10.1145/3567837

[2] V. I. Akpan, U. A. Igwe, I. C. Mpamah, and C. O. Okoro. [n. d.]. Social Constructivism: Implications on Teaching and Learning. https://www.eajournals.org/wp-content/uploads/Social-Constructivism.pdf. Accessed: 2025-05-01.

[3] Lorin W. Anderson and David R. Krathwohl (Eds.). 2001. *A Taxonomy for Learning, Teaching, and Assessing: A Revision of Bloom's Taxonomy of Educational Objectives.* Allyn Bacon, Boston, MA. Part of the Pearson Education Group.

[4] E. Aronson. 1978. *The Jigsaw Classroom.* SAGE Publications. https://books.google.com.br/books?id=EUOfAAAAMAAJ

[5] Elliot Aronson and Shelley Patnoe. 2011. *Cooperation in the Classroom: The Jigsaw Method* (3 ed.). Pinter & Martin, London.

[6] Luis Berdun, Analia Amandi, and Marcelo Campo. 2011. An intelligent tutor for teaching software design patterns. *Computer Applications in Engineering Education* 22, 4 (2011), 583 – 592. https://doi.org/10.1002/cae.20582 Cited by: 5; All Open Access, Green Open Access.

[7] blind author. blind year. blind title. In *blind conference* (blind location). blind publisher, blind city, 00000. https://doi.org/blinddoi

[8] Grady Booch. 2018. The History of Software Engineering. *IEEE Software* 35, 5 (2018), 108–114. https://doi.org/10.1109/MS.2018.3571234

[9] Frank Buschmann, Regine Meunier, Hans Rohnert, Peter Sommerlad, and Michael Stal. 1996. *Pattern-Oriented Software Architecture, Volume 1: A System of Patterns.* John Wiley & Sons.

[10] Jie-Qi Chen and Yu Huang. 2024. International Journal of STEM Education. *International Journal of STEM Education* 11 (2024), 36. https://doi.org/10.1186/s40594-024-00495-2

[11] Deepti Chopra, Gagandeep Kwatra, Bharti Bhandari, Jaspreet K. Sidhu, Jayant Rai, and C.D. Tripathi. 2023. Jigsaw Classroom: Perceptions of Students and Teachers. *Medical Science Educator* 33, 4 (2023), 853 – 859. https://doi.org/10.1007/s40670-023-01805-z Cited by: 5; All Open Access, Green Open Access.

[12] Fatemeh Darabi, Zahra Karimian, and Alireza Rohban. 2025. Putting the pieces together: comparing the effect of jigsaw cooperative learning and lecture on public health students' knowledge, performance, and satisfaction. *Interactive Learning Environments* 33, 1 (2025), 495 – 512. https://doi.org/10.1080/10494820.2024.2351157

[13] Océane Cochon Drouet, Vanessa Lentillon-Kaestner, and Nicolas Margas. 2023. Effects of the Jigsaw method on student educational outcomes: systematic review and meta-analyses. *Frontiers in Psychology* 14 (2023). https://doi.org/10.3389/fpsyg.2023.1216437

[14] Eric Freeman and Elisabeth Robson. 2004. *Head First Design Patterns.* O'Reilly Media, Inc.

[15] Erich Gamma, Richard Helm, Ralph Johnson, and John Vlissides. 1994. *Design Patterns: Elements of Reusable Object-Oriented Software.* Addison-Wesley Professional.

[16] Lina Garcés. 2025. Dataset Teaching software design patterns: An experience using the Jigsaw classroom. XXXIX Simpósio Brasileiro de Engenharia de Software (SBES), Recife, PE, Brazil. https://doi.org/10.5281/zenodo.16161087 Data set.

[17] Vahid Garousi, Görkem Giray, Eray Tüzün, Cagatay Catal, and Michael Felderer. 2019. Aligning software engineering education with industrial needs: A meta-analysis. *Journal of Systems and Software* 156 (2019), 65–83. https://doi.org/10.1016/j.jss.2019.06.044

[18] Çağdaş Evren Gerede. 2024. Are We Asking the Right Questions to ChatGPT for Learning Software Design Patterns? *UBMK 2024 - Proceedings: 9th International Conference on Computer Science and Engineering* (2024), 1092 – 1097. https://doi.org/10.1109/UBMK63289.2024.10773596 Cited by: 0.

[19] Paul Gestwicki and F. Sheldon Sun. 2008. Teaching design patterns through computer game development. *ACM Journal on Educational Resources in Computing (JERIC)* 8, 1 (March 2008), 2:1–2:21. https://doi.org/10.1145/1348713.1348715

[20] Masita Abdul Jalil, Nurul Azarina Abd. Rahman, Noraida Hj. Ali, Shahrul Azman Mohd Noah, Noor Maizura Mohamad Noor, and Fatihah Mohd. 2020. Development of A Learning Model on Software Design Pattern Selection for Novice Developers. *ACM International Conference Proceeding Series* (2020), 108 – 113. https://doi.org/10.1145/3383923.3383966 Cited by: 1.

[21] Zoran Jeremic, Jelena Jovanovic, and Dragan Gasevic. 2011. An environment for project-based collaborative learning of software design patterns. *International Journal of Engineering Education* 27, 1 PART 1 (2011), 41 – 51. https://www.scopus.com/inward/record.uri?eid=2-s2.0-79958847658&partnerID=40&md5=a38b2018c8be90caa7ea05704da3b118 Cited by: 15.

[22] Guillermo Jiménez-Díaz, Mercedes Gómez-Albarrán, and Pedro A. González-Calero. 2008. Teaching GoF Design Patterns through Refactoring and Role-Play. *International Journal of Engineering Education* 24, 4 (2008), 717–728. https://www.ijee.ie/articles/Vol24-4/s9_Ijee2082.pdf

[23] Joshua Kerievsky. 2004. *Refactoring to Patterns.* Pearson Higher Education.

[24] Amruth N. Kumar, Rajendra K. Raj, Sherif G. Aly, Monica D. Anderson, Brett A. Becker, Richard L. Blumenthal, Eric Eaton, Susan L. Epstein, Michael Goldweber, Pankaj Jalote, Douglas Lea, Michael J. Oudshoorn, Marcelo Pias, Susan Reiser, Christian Servin, Rahul Simha, Titus Winters, and Qiao Xiang. 2023. *Computer Science Curricula 2023.* ACM Press, IEEE Computer Society Press, and AAAI Press, New York, NY, USA. https://doi.org/10.1145/3664191

[25] Soohyun Nam Liao, William G. Griswold, and Leo Porter. 2018. Classroom experience report on jigsaw learning. In *Proceedings of the 23rd Annual ACM Conference on Innovation and Technology in Computer Science Education* (Larnaca, Cyprus) *(ITiCSE 2018).* Association for Computing Machinery, New York, NY, USA, 302–307. https://doi.org/10.1145/3197091.3197118

[26] Alan R.L. Loyola Alvarez and Segundo E. Cieza-Mostacero. 2024. Serious Video Game to Improve the Learning of Software Design Patterns. *TEM Journal* 13, 3 (2024), 2557 – 2567. https://doi.org/10.18421/TEM133-81 Cited by: 0; All Open Access, Gold Open Access.

[27] Hira Moin, Sadaf Majeed, Tatheer Zahra, Sarim Zafar, Amna Nadeem, and Sidra Majeed. 2024. Assessing the impact of jigsaw technique for cooperative learning in undergraduate medical education: merits, challenges, and forward prospects. *BMC Medical Education* 24, 1 (2024). https://doi.org/10.1186/s12909-024-05831-2 Cited by: 2; All Open Access, Gold Open Access, Green Open Access.

[28] G. M. Novak. 2011. Just-in-time teaching. *New Directions For Teaching & Learning* 2011, 128 (2011), 63–73.

[29] Damla Oguz and Kaya Oguz. 2019. Perspectives on the Gap Between the Software Industry and the Software Engineering Education. *IEEE Access* 7 (2019), 117527–117543. https://doi.org/10.1109/ACCESS.2019.2936660

[30] Sofia Ouhbi and Nuno Pombo. 2020. Software Engineering Education: Challenges and Perspectives. In *2020 IEEE Global Engineering Education Conference (EDUCON).* 202–209. https://doi.org/10.1109/EDUCON45650.2020.9125353

[31] Jose Antonio Pow-Sang. 2015. Replacing a traditional lecture class with a jigsaw class to teach analysis class diagrams. *Proceedings of 2015 International Conference on Interactive Collaborative Learning, ICL 2015* (2015), 389 – 392. https://doi.org/10.1109/ICL.2015.7318059 Cited by: 8.

[32] José Antonio Pow-Sang. 2017. Experiences using the Jigsaw learning technique to teach IFPUG function points. *EDUNINE 2017 - IEEE World Engineering Education Conference: Engineering Education - Balancing Generalist and Specialist Formation in Technological Carriers: A Current Challenge, Proceedings* (2017), 76 – 79. https://doi.org/10.1109/EDUNINE.2017.7918186 Cited by: 3.

[33] José Antonio Pow-Sang Portillo and Pedro G. Campos. 2009. The Jigsaw Technique: Experiences Teaching Analysis Class Diagrams. In *2009 Mexican International Conference on Computer Science.* 289–293. https://doi.org/10.1109/ENC.2009.31

[34] Yilong Pu, Libing Zhang, and Heng Luo. 2024. Using Jigsaw Pedagogy to Promote Learning in Unplugged Information Technology Class: An Experimental Study. *Proceedings - 2024 6th International Conference on Computer Science and Technologies in Education, CSTE 2024* (2024), 347 – 351. https://doi.org/10.1109/CSTE62025.2024.00072 Cited by: 0.

[35] Alexander Shvets. 2021. Catálogo dos Padrões de Projeto. https://refactoring.guru/pt-br/design-patterns/catalog Accessed: 2025-05-03.

[36] Helena Silva, José Lopes, Eva Morais, and Caroline Dominguez. 2023. Fostering Critical and Creative Thinking through the Cooperative Learning Jigsaw and Group Investigation. *International Journal of Instruction* 16, 3 (2023), 261 – 282. https://doi.org/10.29333/iji.2023.16315a Cited by: 4; All Open Access, Gold Open Access.

[37] Sociedade Brasileira de Computação. 2017. *Referenciais de Formação para os Cursos de Graduação em Computação 2017.* Sociedade Brasileira de Computação, Porto Alegre, RS, Brasil. https://books-sol.sbc.org.br/index.php/sbc/catalog/view/134/586/904

[38] Ye Tao, Guozhu Liu, Jürgen Mottok, Rudi Hackenberg, and Georg Hagel. 2015. Just-in-Time-Teaching experience in a Software Design Pattern course. In *2015 IEEE Global Engineering Education Conference (EDUCON).* 915–919. https://doi.org/10.1109/EDUCON.2015.7096082

[39] Marco Tulio Valente. 2020. *Engenharia de Software Moderna: Princípios e Práticas para Desenvolvimento de Software com Produtividade.* Editora Independente. https://engsoftmoderna.info/ Accessed: 2025-05-03.

[40] Eva Vives, Céline Poletti, Anaïs Robert, Fabrizio Butera, and Pascal Huguet. 2024. Learning With Jigsaw: A Systematic Review Gathering All the Pieces of the Puzzle More Than 40 Years Later. *Review of Educational Research* (2024). https://doi.org/10.3102/00346543241230064 Cited by: 3; All Open Access, Green Open Access.

[41] Chad Williams and Stan Kurkovsky. 2017. Raspberry Pi creativity: A student-driven approach to teaching software design patterns. *Proceedings - Frontiers in Education Conference, FIE* 2017-October (2017), 1 – 9. https://doi.org/10.1109/FIE.2017.8190735 Cited by: 5.

[42] Zehui Zhan, Tingting Li, and Yaner Ye. 2024. Effect of jigsaw-integrated task-driven learning on students' motivation, computational thinking, collaborative skills, and programming performance in a high-school programming course. *Computer Applications in Engineering Education* 32, 6 (2024). https://doi.org/10.1002/cae.22793 Cited by: 1.