Design Decisions Matter: Conveying the Importance of Software Engineering Best Practices through Hybrid PBL

Niyousha Raeesinejad, Mohammad Moshirpour, Laleh Behjat, Yalda Afshar
niyousha.raeesinejad@ucalgary.ca, mmoshirp@ucalgary.ca, laleh@ucalgary.ca, yalda.afshar@ucalgary.ca
Department of Electrical and Software Engineering, University of Calgary

***Abstract*—This Research Full Paper presents the implementation of a hybrid Project-Based Learning (PBL) model in a Software Engineering (SE) course to balance the focus on teaching fundamental knowledge and fostering of applied software development skills through a real-world project, accompanied by contextualized learning and Just-In-Time (JIT) teaching to develop students' scalable knowledge of how to intelligently design with respect to SE best practices. The data is collected from 2 semesters spanning over 2019 and 2020. Based on quantitative and qualitative analysis, this study examines the effectiveness of using the hybrid PBL approach in conveying to students the importance of SE best practices such as the SOLID principles which are deemed as timeless. Results support the claim that JIT lectures help students better evaluate their design decisions and ensure they're on the right track for following optimal design patterns and best practices, and that contextualized learning may be used to develop a notion of why design decisions matter outside of the classroom. Although incorporating these pedagogies in hybrid PBL allows for students' conviction of the significance of SE best practices in academic projects, there still exists room to better convey their significance in industry.**

***Index Terms*—Software Engineering (SE), object-oriented programming (OOP), SE best practices, SOLID principles, project-based learning (PBL), hybrid PBL, just-in-time (JIT) teaching, contextualized learning, scalable knowledge, lifelong learning**

I. Introduction

There exists a field of tension in which SE educators must establish learning frameworks to foster students' abilities to adapt their knowledge and skills to the wide spectrum of development approaches used by companies [1]. Regardless of whether a student enters industry to work on an enterprise-level system complying with strict standards or a minimum viable product through pragmatic procedures, the responsibility lies with educators to highlight these different methodologies as well as the contexts in which they are applied. However, mere awareness of current industry-relevant procedures and technologies is not enough to maintain a stable SE career due to tech companies' continuous adoption of new tools and frameworks.

Shaw [2] addresses these constant variations in industry with explicit consideration for Software Engineers even before SE courses were offered separately from Computer Science (CS) curricula; SE students must regularly update their skills and mastery of new technology even when provided with undergraduate education of the highest level. It is important to note that this circumstance does not imply that SE students are entirely left to manage their own learning with the supplied tools; it is equally imperative for SE educators to provide guidance on how to learn as it is to teach modern software development topics. As elaborated by Heggen [3], "scalable knowledge will consistently be favored over immediate learning such as current industry standards". Therefore, to continuously facilitate lifelong learning for current and future SE students – so that they may not only adapt but contribute positively to the rapid advancement of the tech industry – SE educators must adhere to scalable knowledge of SE principles as a fixed learning outcome.

Throughout recent years, SE programs have been subject to several adjustments to their traditional course delivery, with many advances being made to integrate modern tools and technologies, such as Open Source Software (OSS) [4] and cross-platform mobile programming [5], innovative pedagogies – notably flipped and fragmented classrooms [6], [7], just-in-time (JIT) and project-based learning (PBL) [8]–[11], and hands-on learning strategies like Game Development Based Learning (GDBL) [12]. With the aim of generating more active and collaborative learning environments, such efforts mainly yield positive results depicted by instructor reports of further engagement in classrooms and students claiming to have become more motivated and better oriented for self-learning and retaining technical concepts [13]–[15].

While several of these recognized pedagogies may provide fundamental elements to be incorporated in SE learning frameworks for the decades to come, it is important to acknowledge that their practicality in facilitating scalable knowledge of SE principles and, just as importantly if not more so, delivering their significance to students of more modern classrooms, has not yet been identified to a full extent; there still exists a gap in SE educators' current understanding of how the significance of fundamental SE learning outcomes are perceived according to the learning expectations of SE students.

The primary objective of this paper is to contribute towards establishing the importance of SE best practices at an early stage for students in undergraduate SE classrooms. This research is within the context of emergent active learning frameworks such as PBL and JIT to optimize them to better prepare students for industry, with the hope of gaining further

insight into their perceptions and learning expectations. The case study is a recently redesigned undergraduate SE course at the University of Calgary. The data collected from this course reflects two semesters' (one year apart) worth of course artifacts followed by an end-of-term survey. After conducting quantitative and qualitative analysis, the main contributions are as follows:

- Application of the hybrid PBL model accompanied by other active learning approaches in SE curricula.
- Common learning expectations and workflow patterns of students within a hybrid PBL environment.
- Common design and implementation decisions of students with respect to SE best practices.
- Effectiveness of the hybrid PBL framework in teaching the significance of SE best practices to students.

The remaining structure of this paper is as follows. Section II presents SE principles that are deemed as timeless and their conveyance in academia in contrast with industry. Section III introduces the case study and current challenges. Section IV establishes the research questions and methodology. Section V discusses the results, and Section VI notes threats to validity. Finally, conclusions from the findings are summarized with recommendation for future research in Section VII.

II. Background

A. SE Best Practices in Industry vs. Academia

In industry, Osterweil [16] reveals that while the high-level aspects of a design are not the main concerns of many programmers, they end up having to worry about them anyway during the code design phase. Similarly, Martin [17] insists on the acceptance that "design lives in the code". Due to the interwoven relationship between high-level design and detailed design, programming – in essence – is often perceived as a design activity. The role of "design-implement experiences" is especially crucial in SE education as strong drivers of progressive and deep learning [18]. The importance of design is further verified through a recent systematic literature review of 17 primary studies and publications [19]. The study reveals SE best practices that have been reported to be most frequently taught in academia, in which high-level design is listed as a widely accepted "best practice" which should be taken into consideration within the context of SE education. Although further recurring best practices are discovered, the study acknowledges the insufficient degree to which their relevance is elaborated in actual SE curricula.

During the same year, Kuhrmann et al. [1] investigated the extent to which current higher education institutions (HEIs) cover and coincide with SE practices applied in industry. Their findings, based on 67 SE courses and industry-related research, reveal that while students are reasonably adopting current methods and practices, there are still deviations in their usage style. In fact, "coding standards" is demonstrated to be the most practiced in industry, with more than 90% of the examined software companies actively applying these standards. However, this is only pedagogically exercised in nearly 60% of HEIs and implemented in even less than half.

These findings address an underlying concern in SE curricula; despite how design and code go hand-in-hand, their application and relevance are not entirely reflected. In the meantime, industry continues to acknowledge coding standards as an essential set of guidelines for programmers to consistently implement safe, reliable, and maintainable software systems. As a result, recent graduating students are falling short of employer expectations in significant technical areas. Several of these areas, in addition to certain soft skills, were discovered through a systematic literature review [20]; one of the most important areas is design, which is identified in the study as a "knowledge deficiency" based on students' system designs being "minimalist with important details left out" or otherwise "incomplete". The next area is development and improvement process, wherein it was revealed that students "did not receive an adequate amount of exposure to process standards". These findings are the inherent result of the current gap between academic and production code.

Based on past studies of applied practices in HEIs and industry [1]–[3], [16], [19], one of the main reasons for which SE curricula enforce different coding standards is deployment environments; academic code is oftentimes deployed in a brief safe-to-fail setting, with the purpose of experimentation or proof of concept. In the context of most SE undergraduate courses, a student's program typically must only satisfy functionality and readability requirements. The most severe consequence of a system failure manifests itself in a letter grade. Conversely, production code resides in live deployment and must fulfil modularity and scalability in addition to the previous criteria. This also serves as the reason for the quality gap; since there is far less incentive for academic code to fulfil additional criteria, focusing more on proof of isolated concepts in lieu of a combination of concepts, it is effectively less stable than production code.

B. SOLID Principles and Code Smells

As stated by Wright [21], SE principles provide a vocabulary for describing artifacts of the design and development processes for complex software systems. This is derived into a set of the most prominent axiomatic principles based on their fundamental nature through decades of evolution in software architecture. Despite not being ordered, the first principle in this set is the Open-Closed Principle (OCP) which Martin [22] esteems as most important in object-oriented programming (OOP). He explains further that violation of OCP is the leading cause of rigidity, which renders software systems resistant to change of any magnitude. In contrast to OCP, the Single Responsibility Principle (SRP) is deemed timeless as the root of the other principles in the OOP realm i.e., the Liskov Substitution (LSP), Interface Segregation (ISP), and Dependency Inversion (DIP) principles. Adherence to the latter three often prevents violation of SRP, ensuring high cohesion and low coupling; the system's modules are isolated yet work together as a logical whole [17].

Most software systems are subject to risks involved with the inability to adapt to constant change; however, this risk

is mitigated for those following a clean and simple design. This is confirmed through the SOLID principles, founded on the theory of high cohesion and low coupling. In effect, SOLID allows for software systems of any scale to become more modularized and simplified in terms of both development and maintenance [21]. In addition to violating the SOLID principles, there exist further code smells which, similarly, do not always manifest as bugs but rather problematic design choices, such as poor code readability in the form of long or incohesive methods and lack of proper naming conventions.

C. Conveyance of SE best practices to Students

When making pedagogical decisions, SE educators often must deal with the trade-off between practical relevance and resource allocation; this is directly correlated with a balance between immediate and long-term knowledge [2]. However, resource allocation issues, such as course scope and time constraints, may be resolved once educators shift the focus of their learning outcomes towards acquiring scalable knowledge as opposed to a distorted combination of immediate and long-term knowledge. SE principles – notably SOLID – are deemed as timeless agents of scalable knowledge due to their fundamental nature, structured to allow for their application across various levels of detail and abstraction [21]. This allows for building the foundation of a lifelong learner with an understanding of how to apply principles in various stages and, in effect, follow SE best practices throughout their career.

An overview of core traits of an incoming group of CS and IT students was discovered in 2011 [23]. It was found that this group showed a strong inclination towards only achieving the minimum of expectations, opting for solutions that worked instead of seeking out the "best" ones. The justification for their lack of commitment in producing quality code was either not appreciating its significance or the reasoning behind learning certain concepts. This study verifies whether findings of this nature hold true for students more recently, through contributions of workflow patterns, design and implementation decisions, and perceptions of students with respect to hybrid PBL and SE best practices, with the aim of improving SE curricula to better foster students' growth as lifelong learners.

III. Case Study

Principles of Software Development (ENSF 409) is a core undergraduate course for students in the SE and Electrical Engineering (EE) departments at the University of Calgary. Students in the EE department must be pursuing a Minor in Computer Engineering (CE) to be eligible to register in ENSF 409. This course covers software design and development topics, notably key features of Java and elements of object-oriented design such as inheritance and polymorphism.

ENSF 409 is the first software development course for most students to learn how to write larger applications. Its prerequisite courses focus exclusively on syntax, algorithms, and memory management. With this in consideration, the professor has identified a few challenges with teaching ENSF 409 over the past 7 years. Since it is a software development course, it is crucial to emphasize SE best practices. However, students are eager to code and create applications and may view these practices as obstacles which can slow down their progress. This is particularly the case for small applications built for "cookie-cutter" assignments, where students are not able to witness the implications of poor design choices. Furthermore, one of the biggest challenges is that although students are "pro students" i.e., experts in receiving information, they are not frequently encouraged to relate that knowledge to external contexts; past courses in their degrees often emphasize parroting concepts over student-centric learning. In addition, students are not fully convinced of the importance and practicality of design decisions in OOP. Upon reflecting over past practices and results, the professor mainly attributes this to using a passive learning approach where SE best practices were discussed only after teaching the traditional course content; unfortunately, they believe this has influenced students to regard SOLID principles as more of an "after-thought".

Following recent restructuring, the curriculum now utilizes a hybrid approach to PBL to teach SE best practices more effectively. This framework is not fully PBL in the sense that it still provides traditional lectures and labs – albeit in a JIT fashion – however in addition to assessing individual concepts through assignments, students are once again assessed in their understanding and application of cumulative concepts through a term project. The main purpose of completing lab assignments and a term project is to allow students to see the implications of their design choices more clearly. The lecture material is consistent with students' current tasks through a JIT approach to promote active learning; new topics are immediately reinforced in assignments, allowing for students to gain proficiency beyond theoretical knowledge.

The main learning outcomes after implementing this new structure for ENSF 409 are two-fold. The first is for students to be able to create an object-oriented design of their application based on user requirements. The second goal is for students to be able to evaluate their designs based on SE best practices. For example, it is not enough to learn how inheritance and polymorphism work, but students must understand if their hierarchy is consistent with LSP. These two main learning outcomes were evaluated throughout completion of assignments and the term project which account for 60% of the students' total final grades. Table 1 shows the breakdown of marks used to evaluate the students' final term projects in 2019. "Functionality" and "Constraints" were used to evaluate the two main learning outcomes, with criteria like "SOLID principles followed" and "Documentation and general coding style " highlighting the importance of writing clean, readable, and manageable code.

Throughout the course, the professor stressed to students the importance of learning how to design and develop future-proof software programs that are maintainable and can effectively manage change to prevent disruptiveness and promote sustainability. This was primarily conveyed through the SOLID principles. To highlight the benefit of learning these concepts, the professor often provided anecdotes from industry

TABLE I
MARKING SHEET – 2019 PROJECT MILESTONE III

Category	Criteria	Mark
Functionality	List all tools	/2
	Search by tool name	/2
	Search by tool id	/2
	Check item quantity	/3
	Decrease item quantity after sale	/3
Constraints	Implemented client-server application properly	/5
	Server supports multiple clients	/5
	SOLID principles followed	/3
	A database is used	/3
	Overall flow and user-friendliness of the GUI is reasonable	/1
	Documentation and general coding style	/1
Bonus	Customer GUI and backend	/10
	Client server communicating over a network	/5

reflecting how companies use these concepts as metrics to evaluate potential hires. This approach is widely known as contextualized learning, through which concept is put into context for students to construct their own understanding and interpret its significance outside of the classroom [24], [25].

The significance of SE best practices was further stressed in JIT lectures allocated towards learning how to identify common "code smells" that violate them; this includes problematic design choices i.e., not adhering to the SOLID principles as well as poor code readability through lack of documentation, poor naming conventions, etc. This skill was immediately put into practice through a peer review exercise, during which students conducted code reviews of each other's programs developed as part of a lab assignment and reflected on lessons learned after receiving objective feedback from their peers in the form of identified code smells. Code reviews were mainly facilitated by introducing the concept and significance of pull requests (PRs), code review templates and access to external material such as programming articles, and lastly, hosting online discussion boards for students to share other external resources and discuss common industry practices. These are significant examples of how JIT course material and resources were leveraged to assess the extent to which students appreciate and are able to evaluate their code based on SE best practices i.e., the second main learning outcome.

The end-of-term project is a client-server application for a tool shop and course registration system in the 2019 and 2020 semesters, respectively. These applications were previously designed as standalone desktop applications halfway through the semesters for a lab assignment. As part of the project, students were required to transform their system architecture into a distributed system. The main objective of this full stack comprehensive project is for students to gain further practical experience in recently introduced programming concepts, such as client-server architectures (mainly MVC), multi-threading, and GUI. The project is comprised of three milestones to simulate the iterative development process used for real-life software systems. In addition, students were given a set of functional requirements with no other information aside from a problem statement, which encouraged them to design, develop, and present their solutions with minimal guidance. The first milestone involves producing a set of design class diagrams, including both a high-level system design and several low-level package designs. These diagrams were further discussed in lectures prior to implementing the remaining two milestones, which involve writing code based on these initial designs. The aim of these JIT discussions was to better convey the importance of optimal design decisions to students when directly compared with their own.

IV. METHODOLOGY

The aim of this study is to contribute to the ongoing efforts of educators to optimize teaching frameworks for fostering scalable knowledge and lifelong learning to better prepare the next wave of Software Engineers for industry. This study evaluates the effectiveness of the emergent active learning framework – hybrid PBL – towards teaching students on how to intelligently design and develop with respect to SE best practices as well as their practical importance. Effectiveness is evaluated based on the workflow patterns, performance evaluations, and perceptions of this cohort. To achieve these objectives, this study works towards answering the following sub-questions in the order by which they are presented.

[RQ_1] What are some workflow patterns of students in current SE classrooms? More specifically, this study aims to identify common habits exhibited by SE students towards their own learning within active learning environments. This includes preferred approaches towards completing project milestones and assignments with hard deadlines.

[RQ_2] What are the design and implementation decisions of students with respect to SE best practices? This refers to students' implementation decisions when developing their high-level designs for a software system based on user requirements as well as to what extent SE best practices are upheld based on course requirements. After identifying common design and implementation decisions, the intention of this paper is to verify whether they are founded on any of the workflow patterns identified in RQ_1.

[RQ_3] What are the general perceptions of students in current SE classrooms? Based on student feedback on course structure, this study aims to determine general perceptions on upholding SE best practices; this pertains to designing and developing with careful consideration of the SOLID principles as well as avoiding code smells. Furthermore, this study intends to compare these perceptions in academia with industry to gain a better understanding of the effectiveness of contextualized learning. Lastly, it aims to verify these perceptions based on common design and implementation decisions and workflow patterns identified in RQ_2 and RQ_1 respectively.

Coursework and artifacts were extracted from consenting students' online submissions throughout the 2019 and 2020 Winter semesters. This includes select weekly lab assignments as well as the end-of-term project. Additionally, an end-of-term survey was distributed to students after their final

grades were posted.[1] The survey consists of Likert-style (based on a 5-point scale), multiple-choice (MC), and open-ended questions. The open-ended questions address students' overall experiences throughout the course, while the Likert and MC questions address students' self-evaluations of their skills, knowledge, and learning preferences. For instance, students were asked to what extent their knowledge and skills in certain areas have increased after completing the course as well as their level of preparedness to take on the final project when accompanied by JIT lectures.

The MC questions address students' general work patterns towards completing the project milestones. One question asked whether the project was completed individually or in a group, in which up to three members were allowed. The remaining MC questions were designed similarly to a Likert-style format with a modified scale to represent the students' span of work towards meeting hard deadlines. To address RQ_1, this study presents the distributions of responses in these fields. Students' common design and implementation decisions are derived based on their initial design diagrams and final code submissions for the project. To gain deeper insight into students' levels of preparedness for designing high-level class diagrams, it also discusses certain peer review reflections and responses to open-ended survey questions regarding their learning experiences. Qualitative observations of these course artifacts and open-ended responses are used to address RQ_2.

To investigate students' perceptions towards SE best practices, their responses for two of the Likert questions and one open-ended question are presented. First, the survey asked students to what extent they agree with the statements "my knowledge of Software Engineering best practices (SOLID principles) has increased" and "my industry-relevant skills have developed". Since the number of survey respondents is not large and students' responses to these two Likert questions were measured on an ordinal scale with many tied ranks, a Kendall's tau-b test was conducted to determine the direction and strength of the nonparametric correlation between these two variables [26], [27]. Based on literature, a strong linear correlation is indicated by a minimum correlation coefficient of $|0.7|$ [28], [29]. IBM SPSS Statistics 26 was used for this statistical analysis. The null hypothesis is as follows:

> H_0 : There is no statistically significant association between students' perceptions of improving SE best practices and developing industry-relevant skills.

Next, students were requested to provide open-ended responses regarding which aspects of ENSF 409 they found most useful or valuable. To better identify the frequency of mentioned topics in students' responses, all words were first normalized by removing punctuation and converting to lowercase. These normalized words were then classified into the following topics which were used to generate a word cloud using the Python wordcloud library: 1) course structure, 2) design, 3) SE best practices, 4) soft skills, 5) OOP concepts, and 6) other theoretical concepts. Sentiment analysis was also conducted on students' self-reflections following code peer reviews to identify the contexts behind common perceptions towards upholding SE best practices. For this analysis, Google Cloud's Natural Language API was used to generate sentiment scores and magnitudes for each set of related phrases and keywords associated with the most frequently mentioned SE best practices in the reflections. Based on the context of each word and phrase as well as the impact of their sentiment scores and magnitudes, a number was assigned as a threshold parameter to help categorize each sentiment within the context of student' reflections. These statistical and sentiment analyses are used along with previous findings to address RQ_3.

[1]Survey Link: https://survey.ucalgary.ca/jfe/form/SV$_0SqnOq27ss1wOoZ$

V. Results and Discussion

The total numbers of respondents to the end-of term surveys are 43 and 41 for the 2019 and 2020 semesters, with response rates of 33% and 26%, respectively. 3 incomplete responses are not considered in further analyses. It is important to note that the demographic focus of this study is the most recent cohort of SE students and respondents who were not in the typical year during which the course was taken i.e., second and third year within the Faculty of Engineering are also excluded. As a result, the final dataset consists of 76 participants. Approximately 90% of the survey respondents identified as second-year students, whereas 10% identified as third years. The same statistic was found for SE and EE students, respectively. The EE students were all pursuing a minor in CE and gave a minimum rank of 3 out of 5 for their level of interest in pursuing a SE career.

A. Workflow Patterns

Only 2 students reported to have worked individually on the end-of-term project. This complements previous studies characterizing recent cohorts of undergraduate students as exhibiting more team-oriented thinking and being naturally inclined towards team-based activities [3], [23], [30]. Figure 1 compares workflow patterns towards completing each milestone of the project. More than 60% of students started early and worked consistently until the deadlines to complete milestones II and III, both of which involved implementing their designs in code. In contrast, there is not one main workflow for completing the initial design diagrams for milestone I; most students either worked consistently from an early starting point or began near the deadline instead. This lack of consensus may be attributed to a certain portion of students relying on supplementary clarifications during JIT lectures prior to starting their designs; immediately after announcing the project, the professor delivered JIT lectures covering Java packages and tips on optimal design decisions for completing milestone I, while leaving enough room for students to consider their own design choices. Although most students started working early on all milestones, few reported to have finished them early. This may be similarly justified by students who started implementing their solutions early and opted to wait until after attending topic-relevant lectures for last-minute revisions of their designs prior to submission.

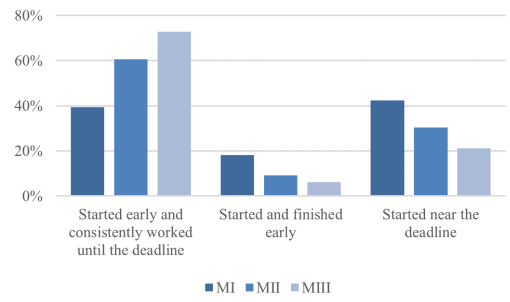

Fig. 1. Workflow of survey respondents for project milestones I, II and III

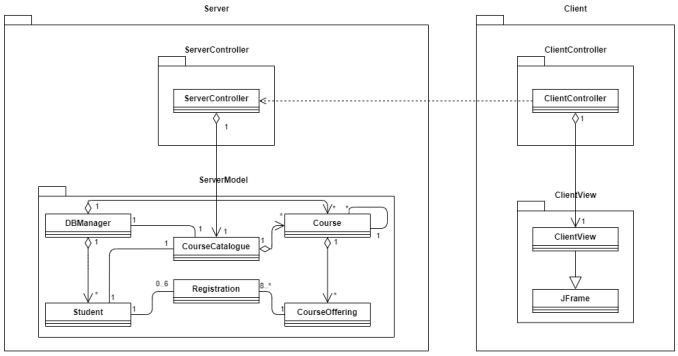

Fig. 2. A student group's milestone I design diagram following MVC architecture for Server and Client packages.

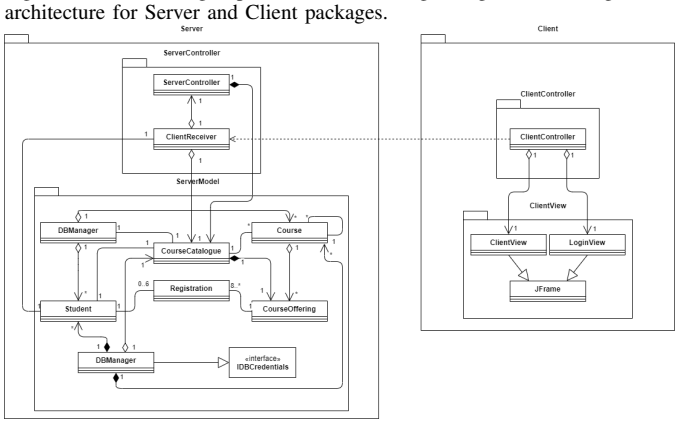

Fig. 3. Milestone III package diagram derived from the student group's code implementation.

B. Design and Implementation Decisions

After manually comparing milestone I diagrams with their corresponding milestone III submissions – with respect to coupling and cohesion – it was found that no student entirely implemented their original designs in code. However, the high-level structure of the package diagrams remained largely unchanged by following an MVC architecture according to the professor's provided examples and tips during the JIT lectures. The server models stayed relatively the same since much of the logic was implemented and satisfied functional requirements as part of a previous lab assignment; server models were generally altered only to manage data in a database and rarely due to changing any business logic. In contrast, classes belonging to other packages were often broken down into further classes in code implementations; students' designs for their client view and client and server controller packages underwent the greatest number of changes, especially for adding connections through sockets and handling GUI events for the frontend portion of their applications.

Figure 2 depicts the high-level design of an arbitrarily chosen student group's milestone I submission and Figure 3 depicts the authors' manually derived package diagram based on that group's milestone III code submission. As shown, both client and server packages were altered following milestone I to mainly establish a client-server connection and implement the GUI. This was done more so by breaking down or adding more classes, as seen in their modified `ServerController`, `ServerModel`, and `ClientView` packages. Repeated observations across further submissions suggest that in the beginning, students were either not upholding SRP by designing classes with excessive responsibilities or underestimating their workload for later milestones.

C. Student Perceptions

Although the professor succeeded in evoking a sense of trust within students for high-level frameworks by getting them to design their own MVC architectures from scratch, there is still room for improvement regarding students' abilities to view a software system through the three mediums of code, memory, and Unified Modeling Language (UML). This may rationalize why some students were not able to foresee or prevent violation of SOLID principles in their designs. One student addressed this by stating in their code review reflection that "the SOLID principles [...] were hard to identify across multiple classes from just looking at code". In further reflections, some students justified not having followed additional SE practices that were encouraged by the professor but not explicitly outlined in the requirements of their reviewed lab assignments, such as adhering to Java documentation guidelines. One student stated that the "assignment requirements [...] had clear expectations [which they] followed when writing the code" and although a lack of documentation was noted by a fellow peer reviewer, they defended that "the assignment did not require it and there was no indication that the code would be reviewed in the future".

Figure 4 presents relative frequencies of key topics mentioned in students' open-ended survey responses regarding which aspects they found most useful or valuable in the course. The most mentioned topic is course structure; students mostly commented on the value of working on lab assignments supported by relevant examples provided in JIT lectures; they generally showed an appreciation for learning and implementing concepts side-by-side by recognizing weekly lab assignments as "a chance to learn concepts by using and implementing them in actual code" and "essential to developing and learning the material taught in lectures". For the project, however, many students preferred to learn new concepts ahead of time rather than delivered through a JIT approach for each milestone.

Fig. 4. Aspects of ENSF 409 which students found most useful/valuable.

One student elaborated that "some of the concepts near the end of the course which were quite relevant to the final project felt like they were never really covered in depth, and [their] team struggled due to this". Theoretical concepts that were implemented solely in the project include GUI and Java database connections, for which students either requested more time or material to be allocated as a "more in-depth approach" to better prepare them before the first milestone, such as incorporating "some labs on [them] rather than going straight to the project".

Students explicitly identified certain practices they would have liked to learn outside the scope of design practices and SOLID principles. It is important to note that this revelation is with consideration for the value that such practices would bring towards completing the term project more so than in their careers. The most frequent is best version control practices which some students revealed they were not familiar with as they "didn't really understand the best way to use GitHub" and suggested that "a more in-depth GitHub tutorial lab would have been very valuable to do the final project in a group". Students also noted professional skills as another valuable aspect in the course, especially through working on the project, for which they found that "learning to cooperate and communicate was very useful". One student expressed the project was the component they enjoyed the most, remarking that "it felt nice to work together as a group, work on specific parts and compile all the code [they] wrote as a group into one final project". It was widely recognized by students that developing as part of a team allowed for them to "fill in for [each other's] weaknesses".

The second most frequently mentioned topic is SE best practices, largely relating to the SOLID principles which some students explicitly noted as "rules" to follow when programming. Within half of these responses, the SOLID principles are listed as the only useful or valuable aspect of the course. In the remaining responses, however, students frequently associate the significance of the principles with design patterns and OOP concepts, with specific consideration for learning how to properly implement their projects. Most students recognized organization and planning skills as a "crucial part of software design [which] can have drastic consequences" and specifically identified the SOLID principles as a "great steppingstone to understand organization [towards] knowing what to code before you start writing code". Figure 5 depicts students' condensed responses for two Likert-style survey questions addressing the extent to which they believe their knowledge of SE best practices – namely SOLID principles – and industry-relevant skills have developed after completing ENSF 409. Nearly 90% of respondents have given a minimum rank of 4 for both categories. The most frequent response-pair consists of a rank of 5 and 5 for knowledge of SE best practices and industry-relevant skills. This indicates a consensus in students' perceptions towards having increased these two competencies. However, it does not guarantee a direct cause-and-and effect relationship between developing the two competencies.

As a result of performing a Kendall's tau-b correlation test, there is a medium, positive association between students' perceptions of improving their knowledge of SE best practices and developing industry-relevant skills, which is statistically significant ($\tau_b = 0.583$, $\rho = 0.000$). This result rejects the null hypothesis. Although it does not indicate a perfect positive monotonous relation, it is sufficient in proving the effectiveness of hybrid PBL in instilling the importance of SE best practices to students. There do however remain some gaps in their conviction of the industry-relevant significance of increasing their knowledge of the SOLID principles. For instance, one survey respondent identified the SOLID principles as the most valuable aspect of the course for understanding organization and code standards; however, when asked what they would improve about the course in the subsequent open-ended question, they suggested to incorporate "more industry focused content", expressing the desire to learn "what programming will be like in the real world after [their] degree".

Table 2 presents the average sentiment score and magnitude for the most prominently discussed SE best practices in students' peer code review reflections. The first practice pertains to documentation with Javadoc and code readability through comments, formatting, and naming conventions for variables, methods, and classes. The next practice refers to upholding

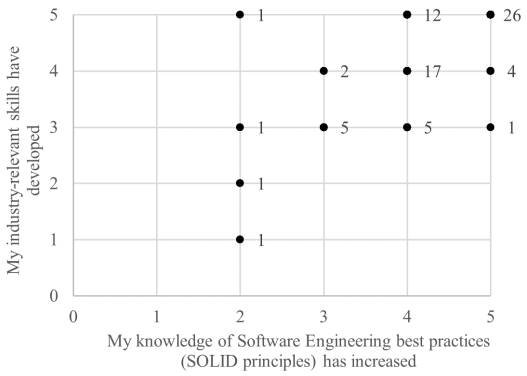

Fig. 5. Scatterplot of student survey responses for 2 Likert-style questions.

TABLE II
AVERAGE SENTIMENT SCORE AND MAGNITUDE

SE Best Practices	Positive		Negative	
	Score	*Magnitude*	*Score*	*Magnitude*
Documentation and Readability	0.2	0.3	-0.5	0.6
SOLID Principles and Design Decisions	0	0.1	-0.4	0.7

the SOLID principles and making optimal design decisions with respect to modularity, expandability, and abstraction. From examining the context behind the generated list of sentiment scores and magnitudes for each keyword, a threshold parameter of -0.1 was assigned. A "positive" sentiment corresponds to a score higher than or equal to the threshold, signifying those students found the activity of identifying code smells and violations of best practices helpful towards improving the quality of their code and developing good habits for future coursework. A score lower than the threshold represents a "negative" sentiment, generally because of students listing dire consequences of violating certain principles or practices in any of the given two categories. A portion of these negatively scored reflections also touch on the difficulty some students reported in identifying code smells – mainly violations of the SOLID principles – either due to discerning from code or not being explicitly prompted by assignment requirements on which practices to follow.

Based on the frequency of sentiment scores, the dominating sentiment of students is positive for both categories of SE best practices. The overall perception towards upholding the SOLID principles and making optimal design decisions is 95% positive, with students recognizing the importance of breaking down overcomplicated logic in their code to reduce coupling and increase cohesion. Many students identified SRP and OCP as the most violated principles in their assignments. Through this process, students stated to have "learned that any function whose job is not clear should be broken down into smaller parts" and that "overly complicated logic in code not only poses as an issue for the expandability or future-proofing of the code, but also makes it harder for other developers to understand and utilize". Students also commented on the value of the peer code review activity towards further clarifying the definitions of the SOLID principles. As a result, many stated to have learned "how important it is to prevent the violation of SOLID principles" as well as gained "an understanding of how [their] code could've been modified to allow for the principles to be followed and make it easier to extend".

VI. THREATS TO VALIDITY

In experimental design, 'history' refers to events occurring between two stages of data collection and 'maturity' denotes the processes within subjects which act as a function of the passage of time [31], [32]. The surveys were distributed all at once at the end of their respective semesters and the questions required participants to reflect on both past and current experiences. Collection of survey responses was completed roughly 3 months after the code peer review exercise. During this period, students continued to build on their theoretical knowledge in further lectures and completed two more lab assignments in addition to the end-of-term project. Naturally, they became more well-versed in following SE best practices and as such, their perceptions are not entirely consistent between these two stages, and students' recollections of past perceptions may not be completely authentic. Furthermore, the peer code review reflection was part of a lab assignment and thus counted towards students' final grades, whereas the survey was optional and distributed after each semester ended, which may also cause variances in students' levels of honesty reflected in the two datasets. To mitigate this threat, the strategy used was to avoid supporting evidence from survey responses when conducting data analysis for peer review reflections, as the survey was distributed afterwards.

VII. CONCLUSION

This paper reports on the effectiveness of teaching and conveying the importance of SE best practices to students through a hybrid PBL framework. When it comes to workflow patterns within the hybrid PBL environment, students mainly rely on JIT lectures for learning and applying concepts in weekly lab assignments; however, for projects of larger scale with strict deadlines, they prefer to have relevant concepts delivered more in depth and ahead of time. For certain students, reliance on JIT teaching towards completing a large-scale project is at the expense of organizational and project management skills such as anticipating and planning future workload. In addition, students tend to avoid upholding certain SE best practices if their significance is relatively less reinforced in lectures or they fall outside the scope of minimum functional requirements in assignments. This reflects previous findings of CS students demonstrating a strong inclination towards achieving minimum expectations in lieu of seeking out the "best" solutions [23].

Incorporating JIT and contextualised learning within a hybrid PBL framework proved to be beneficial towards conveying the importance of SE best practices by getting students to implement and evaluate their designs based on the SOLID principles and general OOP concepts. This supports previous findings from generating more active and collaborative learning environments - namely JIT and PBL - in which students become better oriented for self-learning and retaining technical concepts [13]–[15]. Nevertheless, there remain some gaps in their conviction for the significance of applying the SOLID principles in industry, as students typically only acknowledge the consequences of violating them within the context of academic projects. For future work, further contributions are encouraged to the existing pool of learning habits and perceptions of SE students in response to other emergent pedagogical frameworks applied in physical and remote classrooms. In addition, SE educators are encouraged to consider alternative methods of teaching and conveying the practical significance of SE best practices, such as industry-sponsored projects.

REFERENCES

[1] M. Kuhrmann, J. Nakatumba-Nabende, R.-H. Pfeiffer, P. Tell, J. Klünder, T. Conte, S. G. MacDonell, and R. Hebig, "Walking through the method zoo: Does higher education really meet software industry demands?" in *2019 IEEE/ACM 41st International Conference on Software Engineering: Software Engineering Education and Training (ICSE-SEET)*, 2019, pp. 1–11.

[2] M. Shaw, "Software engineering education: A roadmap," in *Proceedings of the Conference on The Future of Software Engineering*, ser. ICSE '00. New York, NY, USA: Association for Computing Machinery, 2000, p. 371–380. [Online]. Available: https://doi.org/10.1145/336512.336592

[3] S. Heggen and M. Cody, "Hiring millennial students as software engineers: A study in developing self-confidence and marketable skills," in *2018 IEEE/ACM International Workshop on Software Engineering Education for Millennials (SEEM)*, 2018, pp. 32–39.

[4] G. Pinto, C. Ferreira, C. Souza, I. Steinmacher, and P. Meirelles, "Training software engineers using open-source software: The students' perspective," in *2019 IEEE/ACM 41st International Conference on Software Engineering: Software Engineering Education and Training (ICSE-SEET)*, 2019, pp. 147–157.

[5] P. Muyan-Özçelik, "A hands-on cross-platform mobile programming approach to teaching oop concepts and design patterns," pp. 33–39, 2017.

[6] N. M. Paez, "A flipped classroom experience teaching software engineering," in *2017 IEEE/ACM 1st International Workshop on Software Engineering Curricula for Millennials (SECM)*, 2017, pp. 16–20.

[7] W. Billingsley, "The case of the fragmented classroom," in *2019 IEEE/ACM 41st International Conference on Software Engineering: Software Engineering Education and Training (ICSE-SEET)*, 2019, pp. 74–83.

[8] A. Martinez, "Use of jitt in a graduate software testing course: An experience report," in *2018 IEEE/ACM 40th International Conference on Software Engineering: Software Engineering Education and Training (ICSE-SEET)*, 2018, pp. 108–115.

[9] R. Juárez-Ramírez, S. Jiménez, and C. Huertas, "Developing software engineering competences in undergraduate students: A project-based learning approach in academy-industry collaboration," in *2016 4th International Conference in Software Engineering Research and Innovation (CONISOFT)*, 2016, pp. 87–96.

[10] S. C. d. Santos, Santana, L. Santana, P. Rossi, L. Cardoso, U. Fernandes, C. Carvalho, and P. Tôrres, "Applying pbl in teaching programming: an experience report," in *2018 IEEE Frontiers in Education Conference (FIE)*, 2018, pp. 1–8.

[11] D. Shoham, R. Paul, and M. Moshirpour, "Student perceptions of project-based learning in a software engineering course," in *The 16th International CDIO Conference,*, vol. 1, 2020, pp. 268–279.

[12] M. R. De Almeida Souza, L. Furtini Veado, R. Teles Moreira, E. Magno Lages Figueiredo, and H. A. X. Costa, "Games for learning: bridging game-related education methods to software engineering knowledge areas," in *2017 IEEE/ACM 39th International Conference on Software Engineering: Software Engineering Education and Training Track (ICSE-SEET)*, 2017, pp. 170–179.

[13] S. Krusche, B. Bruegge, I. Camilleri, K. Krinkin, A. Seitz, and C. Wöbker, "Chaordic learning: A case study," in *2017 IEEE/ACM 39th International Conference on Software Engineering: Software Engineering Education and Training Track (ICSE-SEET)*, 2017, pp. 87–96.

[14] R. Holmes, M. Allen, and M. Craig, "Dimensions of experientialism for software engineering education," in *2018 IEEE/ACM 40th International Conference on Software Engineering: Software Engineering Education and Training (ICSE-SEET)*, 2018, pp. 31–39.

[15] Z. Shakeri Hossein Abad, M. Bano, and D. Zowghi, "How much authenticity can be achieved in software engineering project based courses?" pp. 208–219, 2019.

[16] L. Osterweil, *What is software?:*, 06 2018, pp. 59–76.

[17] R. C. Martin and J. O. Coplien, *Clean code: a handbook of agile software craftsmanship*. Upper Saddle River, NJ [etc.]: Prentice Hall, 2009.

[18] C. Brito, M. Ciampi, R. Vasconcelos, L. Amaral, H. Santos, and V. Barros, "Rethinking engineering education," 10 2017, pp. 1–5.

[19] M. Marques and J. Robledo, "What software engineering â€œbest practicesâ€ are we teaching students - a systematic literature review," in *2018 IEEE Frontiers in Education Conference (FIE)*. Los Alamitos, CA, USA: IEEE Computer Society, oct 2018, pp. 1–8. [Online]. Available: https://doi.ieeecomputersociety.org/10.1109/FIE.2018.8658576

[20] A. Radermacher and G. Walia, "Gaps between industry expectations and the abilities of graduates," in *Proceeding of the 44th ACM Technical Symposium on Computer Science Education*, ser. SIGCSE '13. New York, NY, USA: Association for Computing Machinery, 2013, p. 525–530. [Online]. Available: https://doi.org/10.1145/2445196.2445351

[21] D. Wright, "Towards a theory of software design: Timeless principles of software system design," 01 2007, pp. 320–325.

[22] R. C. Martin, *Design Principles and Design Patterns*, 2000. [Online]. Available: http://www.objectmentor.com/resources/articles/Principles_and_Patterns.pdf

[23] C. Roberson, "Aligning generations to improve retention in introductory computing courses," *J. Comput. Sci. Coll.*, vol. 26, no. 6, p. 30–36, Jun. 2011.

[24] R. G. Berns and P. M. Erickson, "Contextual teaching and learning: Preparing students for the new economy. the highlight zone: Research @ work no. 5." 2001.

[25] F.-T. Leow and M. Neo, "Students' perceptions of a constructivist classroom: A collaborative learning approach," 10 2013, pp. 1–11.

[26] L. Cohen, L. Manion, and K. Morrison, "Research methods in education," *http://lst-iiep.iiep-unesco.org/cgi-bin/wwwi32.exe/[in=epidoc1.in]/?t2000=011160/(100)*, 01 2000.

[27] A. Field, *Discovering Statistics Using IBM SPSS Statistics*, 4th ed. Sage Publications Ltd., 2013.

[28] D. R. Helsel and R. M. Hirsch, *Statistical Methods in Water Resources*. U.S. Geological Survey, 2002.

[29] H. Akoglu, "User's guide to correlation coefficients," *Turkish Journal of Emergency Medicine*, vol. 18, 08 2018.

[30] H. Partridge and G. Hallam, "Educating the millennial generation for evidence based information practice," *Library Hi Tech*, vol. 24, 07 2006.

[31] T. D. Cook and D. T. Campbell, *Quasi-Experimentation: Design and Analysis Issues for Field Settings*. Houghton Mifflin, 1979.

[32] D. T. Campbell and J. C. Stanley, *Quasi-Experimentation: Design and Analysis Issues for Field Settings*. Houghton Mifflin, 1963.