Effectiveness of Hackathons in Software Engineering Education

Risat Haque
Department of Electrical and Software Engineering
University of Calgary
Calgary, Canada
risat.haque@ucalgary.ca

Ali Salmani
Department of Electrical and Software Engineering
University of Calgary
Calgary, Canada
ali.salmani@ucalgary.ca

Niyousha Raessinejad
Department of Electrical and Software Engineering
University of Calgary
Calgary, Canada
niyousha.raeesinejad@ucalgary.ca

Mohammad Moshirpour
Department of Electrical and Software Engineering
University of Calgary
Calgary, Canada
mmoshirp@ucalgary.ca

***Abstract*—This Research Full Paper presents quantitative and qualitative data on the effectiveness of hackathons in Software Engineering Education. Hackathons have become a growing part of Software Engineering (SE) education in the past decade. Although the academic environment develops technical foundations, a hackathon can develop key competencies to hone lifelong learning. Improving interpersonal, entrepreneurial, and technical skills better prepare SE students for a career after graduation and reinforces engineering-relevant skills such as problem-solving, teamwork, and management. This study examines students' abilities to transfer relevant course skills into the hackathon environment, specifically those related to SE best practices such as software design, SOLID principles (Single-responsibility, Open-Closed, Liskov Substitution, Interface Segregation, and Dependency Inversion Principle), and Object-Oriented Programming (OOP). Furthermore, through faculty-led hackathon prep sessions and workshops, students are trained to follow an adapted design-thinking process known as the Hackathon Design Thinking Process (HDTP). This study directly addresses the following sub-questions: to what extent do participants feel that hackathons have a positive impact on their SE education, what are the participants' perceptions of SOLID principles and OOP skill development during the hackathon, and how effectively are participants able to apple SOLID principles and OOP practices during the hackathon? The data is aggregated from a second-year SE undergraduate and a first-year SE masters cohort from participant perception surveys, judges, and project submissions from two hackathons. We utilize Natural Language Processing (NLP) with Google's Sentiment Analysis, conduct Kendall's Tau-b Test using IBM's SPSS Statistics Tool, and compare Class Diagrams with student-submitted code. Results confirm that participants believe hackathons positively impact their education and tend to sacrifice planning time to implement solutions, often disobeying SE design principles and best practices due to the fast-paced nature of hackathons.**

***Index Terms*—Hackathons, software engineering, design thinking, object-oriented programming**

I. Introduction

A. Hackathons

A hackathon is a space that accelerates the growth of key competencies through rapid prototyping and is an immersive experience that many, both SE students, designers, innovators, and industry professionals tackle within a short period. This environment, both competitive and non-competitive, proposes an alternative method for developing technical skills and interpersonal skills.

Various formats of hackathons have been introduced in the past decade. In particular, the Think Global Hack Local (TGHL) community connects non-profit organizations with student developers [1]. Decker describes the TGHL structure as having been inspired by the University of British Columbia CPSC 319 course, "SE Project." Students dive into a term-long project (13 weeks) with a team of 6 individuals to design and develop a product for a real-world client. This not only tests students' technical skills, but also proposes a new method to test their interpersonal and industry-relevant skills such as planning, organization, research, teamwork, and adaptability.

Alternatively, Rosell [2] describes hackathons using 4 key attributes: focused intensity, novelty, collaboration, and incentivization. Given a shorter period, participants maintain strong focus with given limited technologies. Novelty is demonstrated by showcasing ground-breaking technology that challenges and energizes participants. Although other incentives such as awards, money, or recognition are assets, the strongest is the ability to work with new technology and work on prototypes beyond the hackathon. Before hackathons, participants may undergo important preparations. A multi-faceted Software Development kit (SDK) was provided for participants in which they are given brief training on various technologies [2]. Demonstrations of the SDK took place over mandatory prep sessions. This further engaged attendees and prompted the development of key competencies such as collaboration, planning, research, and teamwork skills.

B. Research Questions

This study aims to recognize the effectiveness of hackathons as a method to improve SE concepts in an academic setting.

Primarily through participant perception of the impact of hackathons on individual learning and through the application of SE best practices. The study will hone into the effectiveness of SOLID principles and the design thinking process in the hackathon environment. The effectiveness is evaluated through student perception and performance through surveys and project submissions from hackathons. The following sub-questions will be used to guide our study:

[RQ$_1$] To what extent do participants feel that hackathons can have a positive impact on their SE education?

[RQ$_2$] What are participants' perceptions on SOLID principles and OOP skill development during the hackathon?

[RQ$_3$] How effectively are participants able to apply SOLID principles and OOP practices during the hackathon?

C. Objective

The primary objective of this paper is to contribute to the effectiveness of hackathons in academia when determining whether students can apply SE best practices in a rapid-prototyping environment. By analyzing participant experience and the quality of projects, we hope to understand to which extent course materials are transferred into the hackathon. Thus, allowing instructors to amend their teaching styles to cater to hackathons.

The case study participants are 2nd year - SE undergraduate students and 1st year - SE masters students at the University of Calgary. For the 2nd year, the first requirement is having completed 2 fundamental programming courses. These courses will reinforce programming skills with memory allocation, pointer arithmetic, and basic data structures and algorithms. In the winter term, students will take a Software Development course with a hackathon at the end. This course is built with a hybrid Project-Based Learning (PBL) approach alongside contextualized learning and Just-In-Time (JIT) teaching [3]–[6]. The master's students enrolled in the Spring-Summer boot camp would have completed a fundamental programming course before participating in the hackathon. Bootcamp students have undergraduate degrees in non-SE disciplines, thus, their skill levels in SE are identical to those that have completed ENSF 409.

The data collected from the hackathons reflect project submissions and end-of-event surveys. After conducting quantitative and qualitative analysis, the main contributions are as follows:

- Perception of participants towards the impact of hackathons on SE education
- Perception of participants on applying SE best practices during a hackathon
- Ability of participants to utilize SE best practices during a hackathon.

The remaining structure is as follows. Section II establishes the relevant key competencies, SOLID principles, and the hackathon design thinking process. Section III introduces our Case Study and observed challenges. Section IV suggests the research question and methodology for data collection and analysis. Section V presents the Results and Discussion, and Section VI notes any Threats to Validity. Finally, the paper will summarize the conclusions and findings with recommendations for future research in section VII.

II. Background

A. Key Competencies

Software engineering (SE) has become an interdisciplinary field that is evolving rapidly. While similar to other engineering disciplines, the demanding nature of SE pressures educators to adapt their teachings to meet current industry standards. Evidently, at the Schulich School of Engineering (SSE), the SE program has evolved to meet those standards in a first-year programming course, ENGG 233, Computing for Engineers [7]. The use of project-based learning (PBL) and a flipped-classroom model encourages students to move beyond achieving technical mastery and exploring topics like project management and teamwork due to the self-paced environment. These key competencies are addressed through both procedural and analytical-based learning, inevitably enhancing student key competencies. The dictionary of basic terms and definitions in higher education define competencies as "specific and measurable patterns of behaviour and knowledge which generate or predict high performance in a position, context or responsibility" [8]. Evidently in ENGG 233, students utilize key competencies to better organize and adapt to the fast-paced nature of the term project - skills that are honed through the flipped-classroom model.

Although ENGG 233 supports the importance of key competency development, ENSF 409, a second-year software development course is the incubator to prepare SE students for industry-relevant practices. ENSF 409 prepares students to develop their understanding and application of fundamental Object-Oriented Programming (OOP) concepts while shedding importance on key competencies such as planning and decision making. These skills correlate to important design decisions in OOP, further supporting their progress and development to achieve technical mastery. Assessment of key competencies should be achieved through a demonstration, which in the case of ENSF 409, is a term project [9]. However, due to the lack of time and resources, current forms of demonstration prevent accurate assessment and development of key competencies such as innovation, leadership, adaptability, planning, research, creative thinking, and organization.

B. SOLID Principles

Software systems are regular victims of rigidity i.e., the tendency to be difficult to change, even in simple ways [10]. Since this problem nearly always originates from poor architecture, software developers must follow a clean and

simple design, which may be achieved by adhering to the allegedly timeless SOLID principles. Founded on the important concepts of high cohesion and low coupling, SOLID allows for software systems of any scale to become more modularized and simplified in terms of both development and maintenance [11].

The first principle, the Single Responsibility Principle (SRP), mandates that each class or module must have a single responsibility, hence only one reason to change [12]. Next is the Open Closed Principle (OCP), whose violation is the leading cause of rigidity due to its reliance on the concept of abstraction for modules to be open for extension but closed for modification. From these primary concepts are derived the remaining principles in the Object-Oriented Programming (OOP) realm: Liskov Substitution Principle (LSP), wherein subclasses should be substitutable for their based classes; Interface Segregation Principle (ISP), wherein clients have their own specific interfaces rather than one general purpose interface; and Dependency Inversion Principle (DIP), wherein every dependency must target an interface or abstract class rather than a concrete class [10]. When adhered together, these principles ensure that a software system's modules are isolated yet work together as a logical whole [12].

C. Hackathon Design Thinking Process

The HDTP is an adaptation of the design thinking process, which acts as a guide for participants in solving complex problems within a given timeframe [13]. Hackathons are not just about the software development expertise but require collaboration between developers, designers, and stakeholders. As a result, all work must be planned and done in a process consisting of stages such as empathize, define, ideate, test, prototype, and pitch in a linear and iterative model.

During the empathize, define and ideate phase, participants conduct preliminary research and engage with end-users through interviews. Afterward, participants define a problem that can be addressed with the given resources and time. During this phase, participants will divide up deliverables, utilize individual skillsets, and create a timeline to execute their solution.

During the test and prototype phase, participants will implement their technical solutions. At this stage, participants will utilize knowledge from coursework and apply SE best practices while collaborating with peers and mentors. The phase is designed to be iterative and will take up the majority of the development period.

Finally, participants present their solution to the given problem, often outlining their initial plan, execution and future implementations.

Hackathon formats are set to support collaboration between the development teams and the stakeholders. The first purpose: academic hackathons follow a design thinking methodology to guide teams in a safe-to-fail environment. With active mentors and support from hackathon organizers, participants obey the design thinking methodology to adapt and change their initial plan, thus, cultivating an experiential learning experience. [14] Second, to raise the hackathon prototypes' utility and quality. [15] notes that users abandon the prototypes because of the lack of focus on their needs as stakeholders, which is due to the developers' lack of domain knowledge. As a result, the developers use their personal experience to develop software that they think will appeal to users. With this in mind, the hackathon format is designed to understand the problem better and deeper to come up with a more appropriate solution.

III. Case Study

A. Academic Course: ENSF 409 and ENSF 592

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. This course covers essential aspects of OOP, software design and development, inheritance, and polymorphism in Java.

An important aspect of ENSF 409 is to highlight the importance of SE best practices by applying various principles. In addition, students code and create applications through small assignments and a final group project in the style of a hackathon. The current curriculum utilizes a hybrid approach to PBL, where students complete weekly lab assignments and a term project to see the implications of poor design more clearly. Lectures are consistent with students' current tasks through a JIT approach to promote active learning. The application of these ideas allows students to go beyond theoretical knowledge and find practicality in the coursework. Students are required to design programs based on client (lab assignment) requirements and evaluate their designs based on SE best practices.

Students are continuously encouraged to consider factors that could disrupt their software programs. The importance of adaptability is conveyed through teaching best design practices and documentation of dependencies and interfaces. The course instructors provide additional support on these concepts by engaging industry members and their experiences. This concept is known as contextualized learning, through which the concept is put into context for students to construct their understanding and interpret its significance outside of the classroom [16], [17].

Programming Fundamentals for Data Engineers (ENSF 592) is a spring course offered to 1st year SE masters students. The course is designed for students with minimal experience in programming and parallels the skill development in ENSF 409. Skill development includes reinforcing knowledge in Java and evaluating design decisions through UML[1] diagrams. Although SOLID principles are not a topic in ENSF 592, hackathon participants engage in best practices of SE through workshops after completing the course.

B. Hackathon Prep-Session Structure

Hackathon organizers designed workshops to accompany skill development not readily available in courses and better

[1]Unified-modeling Language

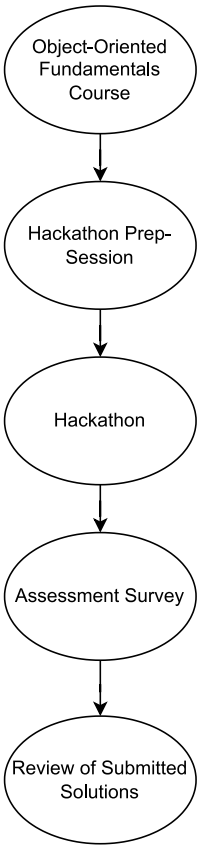

Fig. 1. Flowchart on the process followed by hackathon participants and research team.

prepare students to succeed in the rapid-prototyping environment. For example, boot camp students attended a 3-hour SOLID and HDTP workshop before the hackathon. At this time, instructors presented information in a lecture-style format; then, the instructor worked with participants through an example using class diagrams and live coding. For the Winter Hackathon, organizers led a series of workshops for participants, including a 6-hour session on the HDTP, a 5-hour session on Git/Github, and another 5-hour session on software collaboration. In each of these sessions, participants received information packages at the start of each event; then, with the support of active mentors and organizers, participants would solve problems in groups.

C. Hackathon Structure

The paper explicitly studies two hackathons organized at the University of Calgary for SE and Computer Engineering (CE) students. The process for each hackathon is shown in Fig. 1. Students will develop foundations in OOP in ENSF 409 or ENSF 592, attend the hackathon prep session, participate in the hackathon and complete a survey. The research them will then conduct a review of the submitted solutions. The first hackathon studied is the Winter hackathon, comprised of 160 second-year SE and third-year CE students. Participants tackled a pre-designed problem by hackathon organizers over 24 hours. The specific problem addresses skills students would have gained in ENSF 409; this includes students' ability to design class diagrams (a requirement for the project), utilize a MySQL database, follow OOP core principles in the design of their program, and develop an algorithm to solve the project requirements. The specific problem was to design an inventory management system to reuse and combine modular office furniture pieces. Although the problem was close-ended, hackathon organizers encouraged participants to fulfill client requirements creatively and develop additional features to enhance their program.

The Summer hackathon comprised 40 first-year SE masters students, which took place over 8 hours. The problem set tested students' ability to read JSON[2] text to mimic API usage in the Java environment, read from CSV[3] files with predefined data, develop a basic algorithm, and design a class diagram to plan their program. In both hackathons, the skills tested were similar, and participants were encouraged to go above and beyond to fulfill features they, as developers, would find necessary.

The Summer hackathon included a rubric (Table 1) which gave participants insight on necessary deliverables they should fulfill before submitting their project.

Technical Complexity was assessed by the participants ability to apply Java fundamentals, OOP skills and use of the API. Software Design was assessed by the participants ability to apply SE design practices in their programs. User Experience was assessed by the usability of the program alongside various fail cases for user input. Finally, the presentation was assessed by the teams ability to communicate their process and plan

IV. Methodology

A. Data Collection and Analysis

We collected the artifacts from consenting students' online submissions through 2 hackathons. This includes project submissions, comments made by judges, and a post-survey self-evaluation presented as a survey. Around half the judges are comprised of instructors from the Department of Software Engineering with backgrounds in teaching OOP and SOLID. The other half are industry professionals from local companies with a background in Software Engineering, Computer Engineering, or Computer Science. This paper has combined both datasets since the trends were identical. We will further analyze these similarities in the Results and Discussion section of the paper.

TABLE 1 - SUMMER HACKATHON RUBRIC

Summer Hackathon Judges Rubric	
Technical Complexity	/10
Software Design	/10
User Experience	/10
Presentation	/10

[2]JavaScript Object Notation: Human-readable text exchange format.
[3]Comma-separated Values for data exchange.

The post-hackathon survey consists of multiple-choice (MC), Likert-style questions (based on a 5 point scale), and open-ended questions. The Likert-style questions are designed for self-evaluation and individual perception of experience and ability to apply SE best practices.

To fulfill the requirements for **RQ_1**, we perform sentiment analysis on the open-ended question, *"Do you believe that hackathons can have a positive impact on your software engineering education?"* Participants in both hackathons justified their decision, and following that, the research team utilized Google's Natural Language Processing API (NLP) to generate sentiment scores. Sentiment analysis refers to the opinion mining of texts and is a sub-discipline of natural language processing [18]. The scores reflect on keywords and phrases participants used to answer the given question, and with supporting literature, we conclude that an average score less than -0.5 is considered "negative," and a score greater than 0.5 is considered "positive." We define the following terms in relation to our study:

- **"Negative"=** Hackathons can harm the students SE education.
- **"Positive"=** Hackathons can improve students SE education

NLP is utilized as a data analytics tool over Kendall's Tau-b test since **RQ_1** has a "yes" or "no" approach. As a result, sentiment analysis can separate results quantitatively and would be more beneficial than Kendall's Tau-b test.

The first application of the Likert-style question compares the level of confidence a participant feels before and after the hackathon. To satisfy the requirements of **RQ_2**, we provide a table of responses to a series of questions using the Likert scale. Survey participants answer each question on a scale of 1-5, where one is defined as **"strongly disagree,"** and five is defined as **"strongly agree."**

The first four questions relate specifically to the core principles of OOP:

Q1 (Abstract) My knowledge of Abstraction has increased
Q2 (Encap) My knowledge of Encapsulation has increased
Q3 (Inherit) My knowledge of Inheritance has increased
Q4 (Poly) My knowledge of Polymorphism has increased

Due to the ordinal nature of the data and the large number of tied ranks, we analyze the non-parametric correlations between variable using the Kendall's Tau-B test [19], [20]. Our Kendall's Tau-B hypothesis are:

$H_0 : t_b = 0$, both variables are independent from another and there exists no monatomic relationship.

$H_a : t_b \neq 0$, both variables are not independent and there exists a monatomic relationship.

For instance, the null hypothesis for the correlation of data points between *Q1 (Abstract)* and *Q8 (After)* is as a follow:

$H_0 =$ there exists no significant correlation that a student's knowledge of abstraction has increased with the level of confidence of programming after attending a hackathon.

According to literature, a score of $|0.7|$ indicates a strong relationship while a score close to 0 indicates that a monotonic relationship does not exists [21]. A monotonic relationship is defined when a) one variable increases and the other variable increases or, b) when one variable increases and the other variable decreases. The Kendall's Tau-b tests were performed using IBM SPSS Statistics software.

To satisfy the requirements of **RQ_3**, we analyze UML diagrams (specifically class diagrams) and code written by participants during the hackathon. These design diagrams reflect students' ability to showcase their design decisions in order to fulfill project requirements. The research team looked at the similarities and differences between code and class diagrams, determined the student's ability to apply SOLID principles, and analyzed the holistic behaviors of each class designed by teams. We determined that if either of the following two conditions is met, the group demonstrated one principle in SOLID:

- An example where the principle is demonstrated, or
- the group attempts to utilize the principle through code, even if it does not reflect in their class diagrams.

V. Results and Discussion

The total number of respondents to our study is 74, which includes participant responses from both hackathons, mentor/judges' responses from both hackathons, and consent from project submission from the summer hackathon [22]. After cleaning up our data set by removing null and incomplete responses, we utilize 25 responses from participant surveys, 6 mentors/judges responses, and submissions from 7 groups (22 participants). The demographics of participants include students that have completed ENSF 409 and ENSF 592.

A. *RQ_1 (Positive Impact)*

Table 2 presents the average sentiment score and magnitude of participant perception of the positive impact of hackathons on SE education. Since a 0.7 score is greater than 0.5 we conclude that participants believe hackathons can improve their SE education. In fact, none of the participants received a sentiment score below 0, indicating that, participants view that hackathons play an important part in developing key competencies and having a positive impact on their education. Students have suggested that *"... hackathons can play a crucial role in a student's software development skillset. It provides valuable in brainstorming and rapid development in a collaborative session"* and *"...allow for development of time*

TABLE 2 - AVERAGE SENTIMENT SCORE AND MAGNITUDE

Average Score	Average Magnitude
0.7*	1.0

*All sentiment scores were positive.

budgeting and feature choice decision making." Further analysis of these responses indicate that participant sees value in the HDTP, evidently seen through the following choices of words: *"rapid coding project, prototype, workflows, free environment, experiential learning, creative, engaging environment, decision making, real-world problems, and time-gated exercise."*

Fig. 2 reflects the Likert scale scores for participants' perception of confidence in programming before and after the hackathon. 45% of participants increased their score to a higher value while 45% kept their score the same. The scores suggest that participants believe that hackathons have impacted their programming abilities, including their knowledge of programming fundamentals in Java, syntax, documentation, application of SOLID principles, class diagrams, OOP, database manipulation, or utilization of I/0 (topics tested in both the Winter and Summer hackathon). We achieve a line of best fit with a slope of 0.5362. A number greater than 0 suggests that participants increased their score on average and that their level of confidence in programming had increased during the hackathon.

B. RQ_2 *(Design Perception)*

The results of Table 3 suggest that there exists a weak positive correlation between each OOP principle. The correlation rejects the null hypothesis for each case. We can conclude the following based on the results: A weak positive correlation exists between a participant's knowledge of inheritance and polymorphism after the hackathon. This result is statistically significant ($\tau_b = 0.656, \rho < 0.001$). This correlation suggests that participants can reinforce their knowledge of inheritance through the practice of polymorphism when defining their class relationships. However, this is not sufficient evidence on whether participants, in fact, utilized the inheritance property in the final

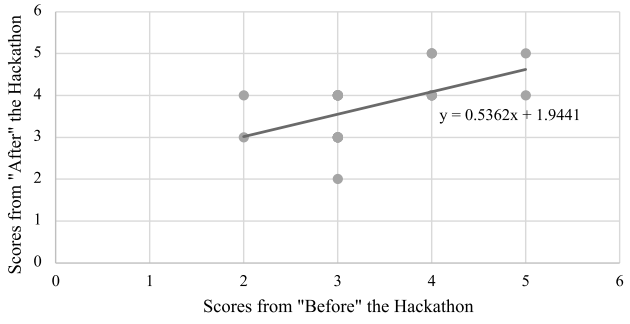

Fig. 2. Scatter plot of responses from question on confidence level of programming before and after the hackathon

design of the program. We will notice that participants fail to find value in the inheritance property in the findings of $\mathbf{RQ_3}$.

According to Fig 3, over 60% of survey respondents selected the importance of SOLID principles with a score greater than or equal to 4, suggesting high importance. Over 58% of survey respondents said that HDTP was highly important, and over 56% of survey respondents said Git was highly important. Student perception reflects that each of these design principles is important in the hackathon environment, and the practice of SOLID, HDTP, and Git can help them succeed. Alternatively, some participants commented that *"... UML Diagrams and memory management is not a priority..."* and *"nature of hackathons [does not] give enough time to spend [on] creating diagrams. A lot of class design/structure [are] created on the fly using best judgment."* The contrasting opinion suggests that not all hackathon participants view SOLID and HDTP of high importance and can *"cost valuable time"* during the implementation and prototype phase of the hackathon.

C. RQ_3 *(Design Application)*

To determine whether participants were successfully able to apply the best practices of SE, we first compared their class diagrams with the submitted code. 3 of the submitted projects have been discarded for this study due to the incomplete nature of the data. 2 of the groups failed to submit a class diagram while the final group included an incomplete diagram. Using 4 of 7 submitted projects in the Summer hackathon, we conclude the following:

None of the groups have diagrams that reflect the code completely. In most cases, classes are missing important private variables and methods. 2 out of 4 groups included classes in their class diagrams that are not used in their program, suggesting that participants deter away from their original plan.

Groups prefer associative relationships between classes. More than 75% of relationships in the class diagrams were unidirectional associative and groups avoided using aggregation and composition. In fact, none of the class diagrams had cardinalities, whereas the group code included boundaries for given class variables.

After analyzing each class diagram and code, using Table 4, we conclude the following: Groups can apply SRP in the context of the given problem. Participants designed their program by separating each class into one for reading data, one to use data for calculations, and one to showcase manipulated data via a Graphical User Interface (GUI). For instances, groups divided each of these classes further by utilizing child-parent relationships and encapsulated classes. Others resorted to interface classes to act as templates for different forms of data.

Half of the groups demonstrated OCP. We considered that a group could utilize OCP when reflecting on how their program would be impacted if additional classes were added. Half the group's designs could account for this change and existing classes and relationships would not be impacted.

TABLE 3 - KENDALL'S TAU-B RESULTS Q1-Q4

Q	Q1-Abstract	Q2-Encap	Q3-Inherit
Q2	$\tau_b = 0.432,\ \rho = 0.022$		
Q3	$\tau_b = 0.366,\ \rho = 0.045$	$\tau_b = 0.436,\ \rho = 0.021$	
Q4	$\tau_b = 0.399,\ \rho = 0.031$	$\tau_b = 0.480,\ \rho = 0.012$	$\tau_b = 0.656,\ \rho < 0.0001$

Only 1 group demonstrated LSP and ISP. We verified that interface classes were used properly, and that, parent classes could easily be replaced by the child without affecting the program. 1 group had broken down interfaces into specific functionality, further proving the use of ISP, OCP, and SRP.

None of the groups demonstrated DIP. We noticed that most groups did not include any abstraction in their designs.

Much of this analysis correlates with the interpretation of judges from the event. They suggested that *"Some students struggled with OOP [principles]"* and *"some groups [did not] demonstrate clear planning or design principles, but most groups showed at least some evidence of software engineering practices."* Judges also indicated that *"UML diagrams could be improved, students showed a good understanding of the various classes but did not demonstrate relationships or encapsulation..."* and *"about half of the groups had fairly comprehensive UML diagrams. Some ran out of time to produce them but had put some thought into their design thinking."*

In both analyses, planning was not sufficient among groups, and most projects did not demonstrate proper use of SOLID and other design principles. These best practices are often ignored, and as seen from both student comments and the analysis of UML diagrams, participants tend to rush past the important design stage and resort to coding. Participants, however, did demonstrate technical mastery in their software with proper documentation and error handling. As both hackathons included user input, groups accounted for failed cases and used personal observation to predict the judges' test scenarios.Judges observed that groups utilized the HDTP

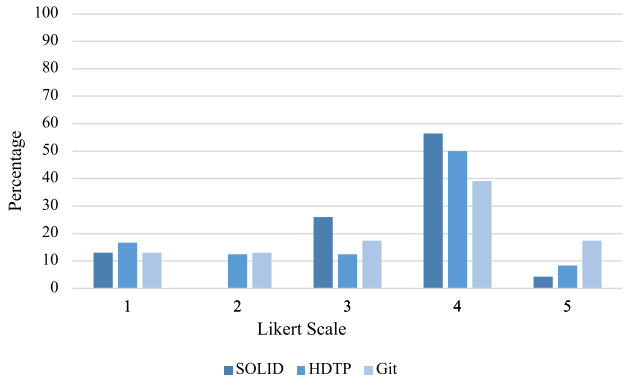

Fig. 3. Bar graph of the percentage of importance on SOLID, HDTP and Git

and participants underwent a small planning phase, but a rigorous prototyping and testing phase.

VI. Threats to Validity

Threats to external validity concern the generalizability of results. Our study includes data from 2 hackathons, including 25 survey respondents, six judges' responses, and seven project submissions. The population represents only 10% of the winter hackathon and 35% of the summer hackathon. However, these values are calculated through the number of initial registrants of the hackathons and not the number of individuals who have submitted a project. Taking this into account, the studied population represents 100% of the winter hackathon and 100% of the summer hackathon. Due to the high drop-off of participants, the surveys may not reflect the collective opinion of students in each cohort and could pose a threat to the study.

Threats to internal validity concern factors internal to the study. A small team of individuals studied the UML diagrams provided by groups. Bias and other unprecedented factors could influence the scoring of SOLID principle applicability in design. To mitigate other factors, the research team utilized "consensus" and openly discussed the design decisions pertaining to each group, thus coming to a generalization for each group.

Other threats include the participants' ability to provide accurate judgment of their skills. For example, questions regarding "*the importance of polymorphism*" and other OOP principles may not reflect the participant's growth in these areas from the hackathon. Although hackathon results are not interpreted in this study, the research studied team UML diagrams to confirm whether teams utilized OOP principles to design their programs. Survey respondents may misinterpret survey questions. For instance, **RQ$_1$** examined programming confidence levels. As the term "confidence" was not described, participants' interpretation could vary. The survey attempted to solve this by adding this question towards the end of the survey, explicitly following questions regarding SOLID and OOP principles.

TABLE 4 - PERCENTAGE OF PROJECTS THAT DEMONSTRATED SOLID PRINCIPLES

Principle	Percentage of Projects
SRP	100
OCP	50
LSP	25
IDP	25
DIP	0

VII. Conclusion

This paper reports the effectiveness of hackathons in SE education. We have conducted qualitative and quantitative analysis from 2 hackathons, including post-hackathon surveys, judges' feedback forms, and project submissions, including UML diagrams and code. Participants include students from a 2nd year SE undergraduate cohort and 1st year SE masters boot camp.

SE students' perception of the impact of hackathons was highly positive. Both cohorts believe that hackathons can influence their education, allowing them to develop skills not emphasized enough in courses. Although PBL courses, such as ENSF 409, allow students to engage in team management and planning, the nature of assessment is quite limited and not enough to gauge interest to apply those practices. SE students reinforced their technical mastery skills, growing the foundations taught in introductory programming courses while considering factors in software development such as testing and prototyping.

Study participants suggested that they developed a better understanding of polymorphism, inheritance, encapsulation, and abstraction. These findings suggest that SE students could reflect on course work, translate relevant skills, and build on foundations to hone their technical mastery. Upon further analysis, the research team discovered contradictory evidence to this. Most groups could not demonstrate 3 out of 5 of the SOLID principles and failed to incorporate abstraction in their designs. Although encapsulation and the use of the single-responsibility principle were effective, groups tended towards the *"easier"* approach when designing their programs. Judges noted that the planning phase (consisting of the define, empathize, and ideation phase) of the HDTP was rushed, preventing groups from reflecting on their designs, such as their class diagram and overall product.

For future studies, we encourage instructors to embed hackathons in their courses' design while considering the transferability of course material to the hackathon context and the development of key competencies for life-long learning. This includes emphasizing the importance of OOP principles such as abstraction in software design. In addition, future studies should develop a pedagogical framework to uphold students' favorable view of hackathons while training SE students to apply best practices of SE beyond the classroom. Studies may reflect on the diverse pool of participants and their impact on the culture of hackathons, the feasibility of open-ended prompts in academic hackathons, or the effect on mental health in the context of rapid prototyping. The findings of these studies may develop the necessary framework in SE education while conveying the importance of Hackathons to academia and may fill in the gaps seen in SE education.

References

[1] *A. Decker, K. Eiselt and K. Voll, "Understanding and improving the culture of hackathons: Think global hack local," 2015 IEEE Frontiers in Education Conference (FIE), El Paso, TX, 2015, pp. 1-8.*

[2] *B. Rosell, S. Kumar and J. Shepherd, "Unleashing innovation through internal hackathons," 2014 IEEE Innovations in Technology Conference, Warwick, RI, 2014, pp. 1-8.*

[3] *A. Martinez, "Use of JiTT in a graduate software testing course: An experience report," Proc. - Int. Conf. Softw. Eng., pp. 108–115, 2018.*

[4] *R. Juarez-Ramirez, S. Jimenez, and C. Huertas, "Developing software engineering competences in undergraduate students: A project-based learning approach in academy-industry collaboration," Proc. - 2016 4th Int. Conf. Softw. Eng. Res. Innov. CONISOFT 2016, pp. 87–96, 2016.*

[5] *S. C. D. Santos et al., "Applying PBL in Teaching Programming: An Experience Report," Proc. - Front. Educ. Conf. FIE, vol. 2018-Octob, pp. 2, 2019.*

[6] *D. Shoham, R. Paul, and M. Moshirpour, "Student Perceptions of Project-Based Learning in a Software Engineering Course," pp. 3, 2020.*

[7] *K. Dornian, R. Paul, S. Goli, I. Rontu, and M. Moshirpour, "Students' Perception Of a Term Project With Respect to Technical Concepts Understanding, Creative Thinking and Interest in Programming in a Large, Flipped Delivery Introductory Programming Course – A Case Study," Proceedings of the Canadian Engineering Education Association (CEEA), pp. 2-3, 2020.*

[8] *Vlascenau, L., Grünberg, L., Pârlea D. (2007). Quality Assurance and Accreditation: A Glossary of Basic Terms and Definitions (Revised and updated edition). Bucharest: UNESCO - CEPES.*

[9] *Holmes, G., Hooper, N. (2000). Core Competence and Education. Higher Education, 40(3), pp. 247-258.*

[10] *R. Martin, "Design principles and design patterns," pp. 2-3, 2000.*

[11] *D. R. Wright, "Towards a theory of software design: Timeless principles of software system design," pp. 2-4, 2007.*

[12] *R. C. Martin, Clean Code: A Handbook of Agile Software Craftsmanship20092Robert C. Martin. Clean Code: A Handbook of Agile Software Craftsmanship . Prentice-Hall, 2008. £27.99, ISBN: 9-780-13235-088-4, vol. 38, no. 6. 2009.*

[13] *M. Gutzmann, "Using hackathons to accelerate procurement's digital transformation,", ProcureHack at Digital Procurement World, Amsterdam, Netherlands, 2019.*

[14] *B. Mark, "Cause Hackathon as Experiential Learning" (2017). Faculty Publications and Scholarship. 11. http://source.sheridancollege.ca/fast$_a$ppl$_p$ubl/11.*

[15] *K. Gama, "Preliminary findings on software engineering practices in civic hackathons" (2017). IEEE/ACM 4th International Workshop on CrowdSourcing in Software Engineering (CSI-SE) IEEE, 14-20.*

[16] *R. Berns and P. Erickson, "Contextual Teaching and Learning: Preparing Students for the New Economy. The Highlight Zone: Research © Work No. 5.," Highlight Zo. Res., no. 5, pp. 1–8, 2001, doi: 10.1111/j.1471-0528.2012.03397.x.*

[17] *L. F. Theng and N. Mai, "Students' perceptions of a constructivist classroom: A collaborative learning approach," Proc. 2013 IEEE 63rd Annu. Conf. Int. Counc. Educ. Media, ICEM 2013, pp. 1–11, 2013, doi: 10.1109/CICEM.2013.6820183.*

[18] *B. Pang and L. Lee, "Opinion Mining and Sentiment Analysis", Foundations and Trends in Information Retrieval, vol. 2, no. 1-2, pp. 1-135, 2008.*

[19] *A. Field, "Discovering Statistics Using SPSS Statistics," Angew. Stat. mit SPSS, 2013, doi: 10.1007/978-3-8349-3571-7$_1$.*

[20] *D. R. Helsel and R. M. Hirsch, "Techniques of Water-Resources Investigations of the United States Geological Survey - Statistical Methods in Water Resources," 2002. Accessed: Aug. 07, 2020. [Online]. Available: http://www.jstor.org/stable/1269385?origin=crossref.*

[21] *H. Akoglu, "User's guide to correlation coefficients," Turkish Journal of Emergency Medicine, vol. 18, no. 3. Emergency Medicine Association of Turkey, pp. 91–93, Sep. 01, 2018, doi: 10.1016/j.tjem.2018.08.001.*

[22] *R. Haque, "Hackathon Data" (2021), Mendeley Data, V1, doi: 10.17632/zg45sccpd8.1.*