Guiding Students to Learn about Design Patterns with Process Oriented Guided Inquiry Learning (POGIL)[1]

PRIYA LOTLIKAR, Indian Institute of Technology- Bombay, India
CLIFTON KUSSMAUL, Green Mango Associates, LLC, USA

Process Oriented Guided Inquiry Learning (POGIL) is an instructional strategy that is based on research and widely used in STEM (science, technology, engineering, and mathematics). This paper describes how POGIL can be a powerful approach to help students learn about patterns and how to use them effectively. We summarize a previously published experiment, which found that when design patterns were taught using POGIL practices, students' learning, skills, and engagement increased. We also describe activities to help students develop an understanding of why and how patterns are useful, how they are structured, and how to use them effectively. Future work should include evaluating these activities and their impact on student learning.

Categories and Subject Descriptors: **K.3.1 [Computers and Education]:** Computer Uses in Education—Collaborative learning;

D.3.3 [Programming Languages]: Language Constructs and Features—Patterns

General Terms: Design, Human Factors

Additional Key Words and Phrases: Active Learning, Patterns, POGIL, Process Oriented Guided Inquiry Learning,

ACM Reference Format:

Lotlikar, P.; Kussmaul, C. 2020. Guiding Students to Learn about Patterns with Process Oriented Guided Inquiry Learning (POGIL). HILLSIDE Proc. of the Conf. on Pattern Lang. of Prog. (October 2020), 14 pages.

1. INTRODUCTION

Patterns describe effective solutions to recurring problems in ways that are specific enough to be useful and general enough to be reusable. Thus, patterns are particularly useful as a way to make the tacit knowledge of experts more explicit and available to others (including students) to help them develop effective solutions to non-trivial problems. However, the abstraction that makes patterns useful and reusable can also make them difficult to understand and apply, so we need effective ways to help people learn about patterns. *Process Oriented Guided Inquiry Learning (POGIL)* is an evidence-based approach to learning, and has been effective across STEM disciplines (e.g., Farrell, Moog, and Spencer, 1999; Straumanis and Simons, 2008; Hu, Kussmaul, Knaeble, Mayfield, and Yadav, 2016). This paper describes how POGIL can be a powerful approach to help students learn about patterns and how to use them effectively. First, it summarizes an experiment with POGIL-style activities to help teach patterns in a graduate computer science program. Second, it describes POGIL-style activities to help students learn about patterns. This complements earlier work (Kussmaul, 2016; Kussmaul, 2017) that explored ways to use patterns to capture POGIL practices.

This paper is organized as follows. The rest of Section 1 briefly describes patterns, POGIL, and why POGIL could help students to learn about patterns.. Section 2 describes an experiment in which POGIL activities were used to help students learn about specific design patterns. Section 3 describes work to develop, pilot, and revise more POGIL activities about patterns. These activities help students to understand: (a) the need for and value of patterns; (b) the general structure of patterns; and (c) specific design patterns and how to apply them. Section 4 describes conclusions and some future directions. Appendix A contains a short version of an activity from the experiment described in Section 2; C and D provide details from the experiment; and D and E show short POGIL-style activities described in Section 3.

[1]**This work is supported by the US National Science Foundation (NSF) Grant #1626765.**
Author's addresses: P. Lotlikar, Dept of IDP-Educational Technology, IIT Bombay, Powaii, Maharashtra 400071 India; email: lotlikarpriya@gmail.com; C. Kussmaul, 730 Prospect Ave, Bethlehem, PA 18018 USA; email: clif@kussmaul.org.

1.1 Patterns

Patterns are detailed descriptions of effective practices, typically structured as a reusable solution to a common problem. Patterns were first used to describe proven practices in architecture (Alexander, Ishikawa, Silverstein, et al, 1977), and have been adapted to other areas, such as software development (e.g., Fowler, 2002; Gamma, Helm, Johnson, and Vlissides, 1995) and education (e.g., Anthony, 1996; Bergin, 2000; Bergin, Kohls, Köppe, et al, 2015). *Software design patterns* focus on software design. A pattern can be described using various formats, but each typically contains similar elements. The pattern's *name* should be concise and evocative. The *context* describes situations in which a pattern might be relevant. The *problem* statement is supported by a description of *forces* (positive and negative) that influence the problem. The *solution* statement is supported by a description of *consequences*, and potential responses. Pattern descriptions often include further *discussion* and *examples*, and refer to other related patterns.

1.2 Teaching Patterns

Teaching students about patterns and how to use them presents some unusual challenges. Table 1 summarizes a pattern language of nine patterns to help teach design patterns (Köppe 2011a; Köppe 2011b). These patterns can help guide faculty to solve common problems in teaching patterns. Patterns represent knowledge and practices in ways that are unfamiliar to many students, as noted in HOLISTIC PATTERN UNDERSTANDING. Students need to study specific patterns relevant to their discipline or course, but also need to master broader concepts, such as the structure of typical patterns and various pattern forms. Patterns often assume that the reader is familiar or even expert in a problem domain; students might still be developing this experience, as noted in IMPLEMENTATION MATTERS. Thus, it is often necessary to scaffold experiences to help students understand the problem, solution, and benefits of specific patterns, as noted in EXPERIENCE OF PROBLEMS and EXPERIENCED ADVANTAGE. At the same time, students need broader perspective and skills, as noted in PRINCIPLE-SUPPORTING PATTERN USE and DISCOVER YOUR OWN PATTERNS.

Table 1: Summary of Pattern Language to Teach Design Patterns (adapted from Köppe, 2011a; 2011b)

Pattern Name	**Summary Patlet**
HOLISTIC PATTERN UNDERSTANDING	Patterns are conceptually different from other design techniques or methods. **Therefore**: Ensure that students understand all aspects of patterns, their lifecycle, and how their use relates to the overall context.
CONTEXT, PROBLEM, & CONSEQUENCES FIRST	Students often go straight to the solution and apply it, skipping other parts of the pattern. **Therefore**: Focus first on the context, problem (including the forces), and consequences. Ensure that the students understand the need for a good solution before applying it.
EXPERIENCE OF PROBLEMS	Students often apply patterns without understanding why the problem is a problem and the consequences if the problem is not addressed properly. **Therefore**: Let the students experience the problems first hand before they implement the pattern.
SIMPLICITY ABOVE PATTERNS	Students often want to show that they understand the patterns by implementing as many as possible, which can add unnecessary complexity without adding value. **Therefore**: Have students give a rationale for all design patterns they use. The application of the pattern should add value to the overall design.
DISCOVER YOUR OWN PATTERNS	Students see patterns as something that intelligent people have written, not as captured "best known practices" that experienced people use without thinking. **Therefore**: Show students how patterns emerge by letting them discover an existing and well-known pattern by themselves.
BEST FITTING PATTERN CHOICE	Students often choose patterns where problems don't match the students problem, context or forces are different, or the consequences are worse than the original problem. **Therefore**: Ensure that students analyse the design problem, context, forces, and consequences, and that all match with the pattern they choose.
EXPERIENCED ADVANTAGE	Students often don't see the advantages of correctly applied pattern solutions. **Therefore**: Give students something of value or satisfaction by letting them experience the advantages one gets after or during the correct application of a pattern.
IMPLEMENTATION MATTERS	Students have difficulties applying design patterns if they only read or hear about them. **Therefore**: Let the students implement the pattern solution.
PRINCIPLE-SUPPORTING PATTERN USAGE	Students often focus on pattern implementations in isolation, which results in a bad overall design. **Therefore**: Help students understand that basic design principles are more important than the patterns.

1.3 Process Oriented Guided Inquiry Learning (POGIL)

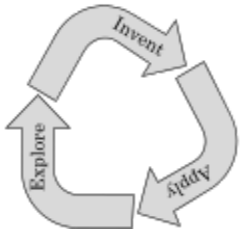


Figure 1: Explore-Invent-Apply (EIA) Learning Cycle

There is ample evidence that most people learn more effectively when they *interact* and discuss topics with other people, and when they *construct* their own understanding (Chi and Wylie, 2014; Piaget, 1964). *Process Oriented Guided Inquiry Learning (POGIL)* is an evidence-based instructional strategy that is collaborative and constructivist (Moog, Creegan, Hanson, et al, 2006; Moog and Spencer, 2008). In POGIL, student teams work on specifically designed activities that guide them to discover and understand core concepts (the *guided inquiry*). POGIL activities contain models (e.g. diagrams, graphs, tables, code excerpts) and questions about the models that guide students through *Explore-Invent-Apply (EIA) learning cycles* (Karplus and Thier, 1967). At the same time, students develop process skills, such as communication, teamwork, critical thinking, and problem solving (the *process oriented*).

POGIL uses teams of three or four students who work together and discuss to improve understanding. Typically, teams stay together for weeks or months, but each member has a different role each day. For example, the *manager* makes sure everyone focuses, participates, and understands the activity, the *recorder* takes notes for the team, and the *speaker* presents results to the rest of the class. The teacher's role shifts from disseminator ("sage on the stage") to facilitator of learning ("guide on the side"), who continually assesses when and how to guide teams as they work (Hanson, 2006). Thus, the teacher might use probing questions or short whole-class discussions to ensure that all teams reach the correct answers. The teacher monitors progress and team interactions, and supports teams that are moving too slowly (or quickly).

Too often, teachers focus on their own actions (e.g., "covering" content in lectures, readings, and assignments). In contrast, POGIL focuses on students and outcomes (learning, skills, and attitudes), and how teachers can support students. For example, POGIL learning objectives are always *active*, *specific*, *student-centered*, and *measurable* (similar to the LEARNING OUTCOMES in Bergin, Kohls, Köppe, et al 2015), and thus avoid terms like "learn", "understand" and "appreciate".

Thus, in POGIL, students interact and construct their own understanding of concepts, leading to better learning outcomes. POGIL often takes more time than a lecture "covering" the same content, although deeper understanding typically helps students to master related content, and teachers spend less time "reviewing" content they "taught" but students didn't really learn. Thus, POGIL is particularly appropriate for *threshold concepts* - difficult concepts that are key to long-term student success. Kussmaul (2016) provides more details on the history of POGIL, evidence of effectiveness, the structure of POGIL activities, and a brief description of a POGIL activity.

1.4 Patterns and POGIL

The abstraction that makes patterns widely useful can also make them difficult to understand and apply, so we need effective ways to help people learn about patterns, why they are useful, how they are structured, how to use them, and even how to create them.

The pattern language for teaching design patterns (Köppe 2011a; Köppe 2011a) focuses mostly on high-level problems and high-level strategies to address them. In contrast, a POGIL activity is a classroom implementation designed to achieve specific learning objectives. However, we see significant correlation between these perspectives. POGIL focuses on understanding key concepts, rather than rote memorization, as do HOLISTIC PATTERN UNDERSTANDING and PRINCIPLE SUPPORTING PATTERN USAGE. POGIL focuses on process and skills, as do CONTEXT, PROBLEM, & CONSEQUENCES FIRST, BEST FITTING PATTERN CHOICE, and PATTERN IMPLEMENTATION MATTERS. POGIL focuses on metacognition, as do SIMPLICITY ABOVE PATTERNS and EXPERIENCED ADVANTAGE. A POGIL activity guides a student team to explore a model, often following the original discovery process, as do EXPERIENCE OF PROBLEMS and DISCOVER YOUR OWN PATTERNS.

However, a POGIL perspective can also yield different views of pedagogical patterns. For example, HOLISTIC PATTERN UNDERSTANDING states that "Patterns are conceptually different from other design techniques or method …" and advises teachers to ensure that "students understand all aspects of patterns ...". However, this problem (conceptual differences with patterns) can also be viewed as a force that intensifies another problem (ensuring that students understand patterns). HOLISTIC PATTERN UNDERSTANDING recommends using other patterns, which can also be viewed in other ways. Thus, CONTEXT, PROBLEM, &

CONSEQUENCES advises teachers to ensure that "students understand the need for a good solution before applying the solution". This can also be viewed as a problem, solved by focusing on other elements before the actual solution, and using discussion and other active learning techniques. Similarly, EXPERIENCE OF PROBLEM and EXPERIENCED ADVANTAGE can also be viewed as problems: how to create such experiences? This POGIL perspective is important to help teachers develop a student-centered view of education.

This paper describes how a POGIL perspective can provide lower-level tactics and classroom activities to help students learn about patterns and how to use (and create) them.

2. LEARNING ABOUT PATTERNS USING POGIL

This section describes and elaborates on a previous study (Lotlikar and Wagh, 2016), which aimed to determine if POGIL is an effective way to teach design patterns. (Appendices A-C provide a sample activity and details not included in the earlier publication.) The study's research questions focused on (1) learning design patterns; (2) learning skills; and (3) engagement. Specifically:

RQ 1. (a) Does POGIL help students to better understand design patterns and do better on tests?
(b) Does POGIL help students to solve real life problems with appropriate design patterns?
(c) Does POGIL help students to perform well in programming tasks using design patterns?
RQ 2. Does POGIL improve student's critical thinking, process skills, and programming skills?
RQ 3. Does POGIL help engage students in class and lab, leading to effective learning?

2.1 Methodology

The study was performed in Goa University, India during summer 2016 for a fourth semester course with 60 students in the Graduate Computer Science Programme. The course included four weekly three-hour sessions, each to teach a design pattern, namely: STATE, CHAIN OF RESPONSIBILITY, OBSERVER and FACTORY METHOD (Gamma, Helm, Johnson, and Vlissides, 1995). In each session, the goal was for students to apply a design pattern to solve a problem. The students were divided into two groups: *Control*, guided through a problem by providing hints in between by the instructor, and *Experimental*, using a POGIL-style approach.

In the experimental (POGIL) group, students were divided into teams of six where each one had a role (e.g., manager, recorder, speaker, and technician). In the Exploration phase, students studied an example problem and a design solution (based on the pattern) using UML diagrams and code samples. In the Concept Invention phase, questions prompted students to explore various possibilities, review a pattern summary, and discuss as a team. In the Application phase, students answered application level questions and implemented the design pattern for a new example. The application phase was conducted in combination with Pair Programming (Williams, McCrickard, Layman, and Hussein, 2008). Afterwards, each team had to discuss and put forward their understanding using Think Pair Share (TPS) (Kothiyal, Majumdar, Murthy, and Iyer, 2013). Students also answered critical thinking questions posed by the instructor.

2.2 Classroom Activities

A set of four POGIL-style activities guided students to develop their own understanding of the design patterns concepts, and to develop teamwork, information processing, critical thinking, and problem solving. Table 2 lists these activities and their learning objectives. Before applying a pattern, a learner should understand the logical and working aspect of the pattern. Since basic OO programming skills were a prerequisite for conducting a POGIL session, codification was part of each module in the sessions.

Table 2: POGIL Activities for Design Patterns used in Study

Activity / Pattern	**Learning Objectives: Students should be able to:**
Introduction to STATE	Explain the concept and identify problems where STATE is applicable, describe how it could be applied, and evaluate possible consequences.
Introduction to CHAIN OF RESPONSIBILITY	Explain the concept and identify problems where CHAIN OF RESPONSIBILITY is applicable, describe how it could be applied, and evaluate possible consequences.
Introduction to OBSERVER	Explain the concept and identify problems where OBSERVER is applicable, describe how it could be applied, and evaluate possible consequences.
Introduction to FACTORY METHOD	Explain the concept and identify problems where FACTORY METHOD is applicable, describe how it could be applied, and evaluate possible consequences.

Each classroom session was monitored and observed by the facilitator and three trained observers. The activity was preceded with a pre-test with questions on the pattern, and followed by a post-test at the same complexity level. At the end, students' feedback was taken. Each activity was divided into three modules. Module 1 (Exploration) had a solved example with: (a) UML representation of the problem and solution; (b) the implemented code; and (c) questions for problem solving, concept understanding, and critical thinking. Module 2 (Concept Invention) had the concept summary for the students to reconnect and map with their concept understanding from Module 1. Module 3 (Application) had questions targeting analysis, refactoring, problem solving, and critical thinking. All three modules were conducted using active learning methods like peer discussion and pair programming. Module 3 was followed by group discussion and debate, where each group had to communicate and justify their understanding of the concept. The group discussion was also encouraged with critical thinking and thought provoking questions by the instructor. As an example, the CHAIN OF RESPONSIBILITY activity is in APPENDIX A. In module 1, the students must first understand the significance of elements in the pattern. This is enhanced with a handout with a solved example - a real-life scenario represented in UML with its code implementation. Students analyze the scenario and code to find the connections between the elements, to understand the underlying working of the pattern. Based on this understanding, students answer the recall and understand level questions. Module 2 briefly describes the pattern to help students build mental models of the concept based on their understanding from module 1. Module 3 returns to the problem posed in Module 1, and students must add more functionality and perspective, and then diagnose the situation.

The subsections below describe results and interpretation of pre-post testing, observation of student engagement, a feedback questionnaire, and rubric-based evaluation of programming assignments.

2.3 Pre-Post Test of Student Learning

In order to assess student learning, in-class Pre-Post testing was used to compare the Control and Experimental groups (for details, see Lotlikar and Wagh, 2016). Two hypotheses were formulated:

H0. POGIL will improve understanding of design pattern concepts, as measured by test scores.

H1. POGIL will help students in solving real life problems with appropriate design patterns.

Table 3 shows the results of Student t-Test performed on the pre-post data for each pattern. A Cohen's effect size of 0.5 (½ of a standard deviation) is considered "moderate", while 0.2 is considered "small" even when it is statistically significant. Thus, the OBSERVER activity seems less effective, and should be revised.

Table 3: Statistics for Pre-Post testing of four activities on specific design patterns.

	STATE	**CHAIN of RESP**	**OBSERVER**	**FACTORY**
T Stat	- 5.378	- 4.28	- 1.761	- 3.803
P(1 tail value)	0.0000012	0.00003	0.042	0.00018
P(2 tail value)	0.0000024	0.00008	0.084	0.0003
Cohen's Effect Size	0.621	0.503	0.237	0.453

2.4 Student Engagement and Feedback

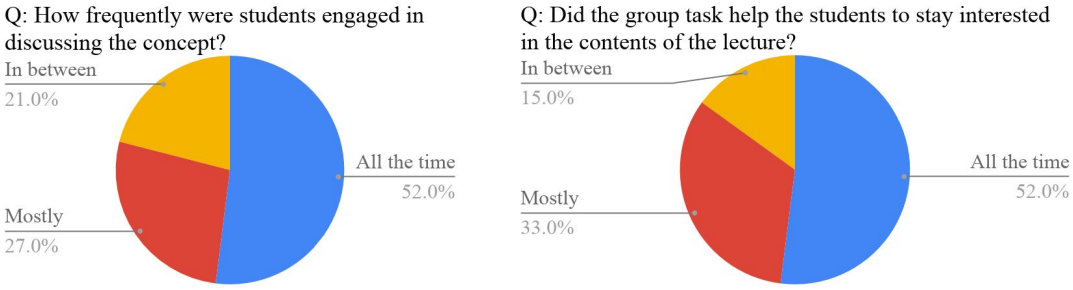
Student engagement was monitored in the Experimental group by three trained observers using an observation protocol followed with survey questions (Lotlikar and Wagh, 2016). As shown in Figure 2, 79% of the students reported spending all or most of their time discussing the concepts and 85% reported that the activity helped them to stay interested all or most of the time, thus showing strong student engagement. Students could also respond "Never", but none did.

After all four POGIL sessions, students answered a feedback questionnaire with 16 questions, organized into three groups each linked to a research question. Appendix B shows the three research questions, and their corresponding feedback questions. Chi-square test was performed on each group of questions. Two hypotheses were formulated (Lotlikar and Wagh, 2016):

H0. Research questions and Feedback questions are independent;

H1. Research questions and Feedback questions are not independent.

Figure 2: Student Engagement Survey Question 1 & 2 Analysis



As shown in Table 4, there is a significant relationship for Groups 3 (Engagement) and 1 (Concepts). showing that POGIL helped students to stay engaged in class leading to effective learning, and helped students to better understand Design Pattern concepts, solve real life problems, and improve programming skills. The relationship for Group 2 (Skills) is not significant, suggesting that the activities should be revised to better support critical thinking and problem solving skills.

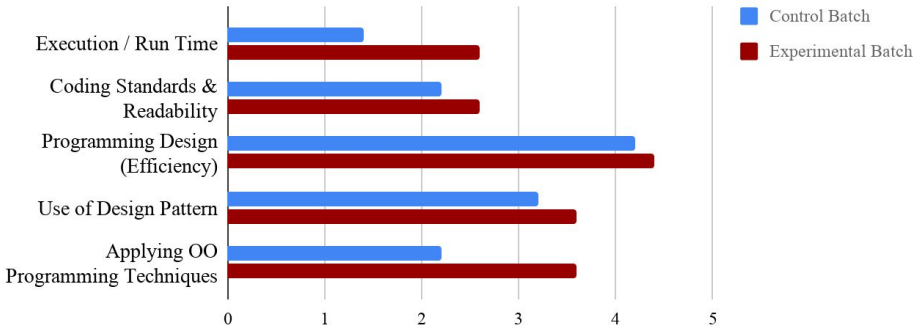
Table 4: Chi-Square Test Result for Research question & Feedback questions

	Group 1 (Concepts)	**Group 2 (Skills)**	**Group 3 (Engagement)**
Chi-Square	22.808	24.822	60.072
p-value	0.02939	0.20829	0.00039869

2.5 Student Programming

Student programming skills during the session were evaluated with rubrics (see Appendix C) with five items each worth 0-5 points. Figure 3 shows the average student scores (x-axis) for each rubric feature (y-axis). This suggests that POGIL helped students to get the solution with appropriate design patterns and also improved their programming skills (applying OO programming techniques) (Lotlikar and Wagh, 2016).

Figure 3: Rubric Evaluation of Control vs Experimental Group



Thus, this study showed that POGIL was associated with significantly greater student learning in three of the four activities; increased student perceptions of engagement, interest, and learning; and better scores on programming assignments using design patterns.

3. LEARNING ACTIVITIES

Inspired by and building on these results, we continue to seek more and better ways to help people learn about patterns and how to apply them effectively. As outlined in HOLISTIC PATTERN UNDERSTANDING, to work effectively with patterns, people should understand:

- Why patterns are an effective way to represent domain knowledge.
- The structure of a typical pattern, and common formats and variations.
- How to read and apply patterns.
- How to identify and write patterns. (Although not everyone who uses patterns will write them).

Thus, we are developing and starting to evaluate more POGIL-style activities to guide students to develop their own understanding of these concepts, and to practice related skills in communication, teamwork, critical thinking, and problem solving. Table 5 lists some of these activities and their learning objectives. Two activities are described below, and Appendices D and E have short versions of each. The full activities are available on request from the authors.

Table 5: New & Proposed POGIL Activities for Patterns

Activity	**Learning Objectives: After this activity, students should be able to:**
Knowledge Management (see section 3.1)	Describe 2 types of knowledge (*explicit, tacit*) and 4 ways to combine them (*articulate, combine, internalize, socialize*), and give or identify examples of each. Describe 2 KM strategies (*codification, personalization*), and give or identify examples of each.
Pattern Structure (see section 3.2)	Describe the key elements of a pattern (*name, problem, forces, solution, consequences, & discussion*), and explain why each is significant.
Reading & Using Patterns	Describe and apply effective strategies to read patterns, decide if they are relevant, and apply them. (in progress)
Finding & Writing Patterns	Describe and apply effective strategies to identify, document, and review patterns within an area of expertise. (not yet written)

3.1 Knowledge Management

Before using patterns to solve problems, it is important for people to understand the characteristics of problems that are or are not suitable for a pattern-based approach. Thus, this activity guides students to understand ways to create, manage, and share knowledge, in order to motivate the use of patterns. (A short version is included in Appendix D.) This activity could be used in courses on computer science, information technology, and other contexts to introduce key ideas in knowledge management. This activity is divided into sections, each of which presents a description or definition of terms, followed by questions to guide student learning.

Section A describes *explicit* and *tacit* knowledge (Nonaka, 1991). First, students decide which type is applicable in various circumstances. Next, they use these new terms to explain the phrase "we know more than we can tell" (attributed to Michael Polanyi) to show that they understand the terms and their relationship. Students then apply this new understanding to explain the value of making tacit knowledge more explicit; this foreshadows the next section, as well as the value of patterns. Section A then presents the four permutations to create knowledge (tacit → tacit, tacit → explicit, etc) (Nonaka, 1991). Students consider a set of examples (e.g., "I used the manual to make a reference sheet", "I do this so often that I don't need to look up the steps") and identify the appropriate permutation in each case. At the end of the section (or for homework), the instructor might have students pick a familiar topic and identify examples of explicit and tacit knowledge and the four permutations.

Section B presents two knowledge management strategies: *codification*, which focuses on storing content, and *personalization*, which focuses on contacting people with relevant experience (Hansen, Nohria, and Tierney, 1999). Students consider a set of examples and identify the appropriate strategy, and then use their new vocabulary (*tacit*, *explicit*, *codification*, *personalization*, etc) to explain when to use each strategy. Finally, students describe characteristics of problems in the middle, for which neither strategy is ideal; these are often the problems for which patterns are most useful.

Thus, this activity guides students to explore new information, develop their own understanding, and then apply that understanding in ways that support learning, foreshadow future topics, and motivate the need and benefits of patterns. Some of the later questions, such as relating new ideas to a familiar topic, could also be used as individual assignments in class or for homework.

3.2 Pattern Structure

Before using patterns effectively, it is helpful for people to understand the structure of a pattern and its common elements. Thus, this activity guides students to consider the presentation and purpose of each element. (A short version is included in Appendix E.) First, students read through a pattern (or a small set of patterns). Second, they notice the use and purpose of stylistic elements (e.g., headings, small caps, bold, special symbols). Next, they match labels (e.g., "context", "forces", "consequences") to descriptions of common elements in patterns. Next, they explore some implications: Which elements could be combined to form a patlet? Would the pattern make sense if the elements were not labeled? Finally, students might demonstrate their new understanding by labeling elements in a pattern, or arranging elements into the correct order (like a Parsons problem or jigsaw puzzle). Another section of this activity guides students to compare *pattern forms* (e.g., Alexandrian, Fowler, Gang of Four) to understand the similarities and differences.

This activity could readily be adapted to other pattern forms and contexts (e.g., education, software design, architecture) by using a different example and making minor changes to the questions. Again, the later questions could be used for individual assignments.

A future activity (in progress) in this sequence will help students learn when and how to read patterns, and how to identify and evaluate patterns they might use in a project. Another future activity (not yet started) will help students learn to identify and document patterns. These activities will likely build on prior work and patterns, such as SIMPLICITY ABOVE PATTERNS, BEST FITTING PATTERN CHOICE, PATTERN IMPLEMENTATION MATTERS , and DISCOVER YOUR OWN PATTERN (Köppe, 2011a; Köppe, 2011b).

4. CONCLUSIONS & FUTURE DIRECTIONS

In this paper, we have described how Process Oriented Guided Inquiry Learning (POGIL) can be a powerful approach to help students learn about patterns and how to use them effectively. We have summarized the results of an experiment that used POGIL-style activities in a graduate computer science program, and described new activities to help students develop an understanding of why and how patterns are useful, how they are structured, and how to use them effectively.

In addition to piloting and revising the POGIL activities described above, we plan to develop more POGIL activities focused on specific patterns, including design patterns, other software patterns, and patterns in other areas, such as architecture and education. We welcome collaborators who want to pilot and provide feedback on activities and develop new activities. We also plan to evaluate the current and future activities and measure the extent to which they enhance understanding of design patterns and other outcomes.

5. ACKNOWLEDGEMENTS

This material is based upon work supported by the US National Science Foundation (NSF) under Grants #1044679 (CS-POGIL) and #1626765 (IntroCS POGIL). Any opinions, findings and conclusions or recommendations expressed are those of the author(s) and do not necessarily reflect the views of the NSF. We also thank The POGIL Project (http://pogil.org), the PLoP reviewers, and our shepherds, Steve Warburton and Ademar Aguiar.

REFERENCES

C. Alexander, S. Ishikawa, and M. Silverstein. 1977. *A Pattern Language: Towns, Buildings, Construction*. Oxford Univ. Press.

D. L. G. Anthony. 1996. Patterns for classroom education. In J. M. Vlissides, J. O. Coplien, and N. L. Kerth, eds., *Pattern Languages of Program Design 2*. Addison-Wesley Longman, pp. 391–406.

J. Bergin. 2000. Fourteen Pedagogical Patterns. In *Proc. of the European Conf. on Pattern Languages of Programs* (EuroPLoP).

J. Bergin, C. Kohls, C. Köppe, Y. Mor, M. Portier, T. Schummer, S. Warburton. 2015. Assessment-driven course design - Foundational patterns. In *Proc. of the European Conf. on Pattern Languages of Programs (EuroPLoP)*, 31:1–31:13.

M. T. H. Chi and R. Wylie. 2014. The ICAP framework: Linking cognitive engagement to active learning outcomes. *Educational Psychologist*, 49, 4, 219-243.

J. J. Farrell, R. S. Moog, and J. N. Spencer. 1999. A Guided-Inquiry General Chemistry Course. *Journal of Chemical Education*, 76, 4, 570-574.

M. Fowler. 2002. *Patterns of Enterprise Application Architecture*. Addison-Wesley.

E. Gamma, R. Helm, R. Johnson, J. Vlissides. 1995. *Design Patterns: Elements of Reusable Object-Oriented Software*. Addison-Wesley.

M. T. Hansen, N. Nohria, and T. Tierney. 1999. What's your strategy for managing knowledge? *Harvard Business Review*, 77, 2, 106-116.

D. M. Hanson. 2006. *Instructor's Guide to Process-Oriented Guided-Inquiry Learning*. Pacific Crest.

H. Hu, C. Kussmaul, B. Knaeble, C. Mayfield, and A. Yadav. 2016. Results from a survey of faculty adoption of Process Oriented Guided Inquiry Learning (POGIL) in Computer Science. In *Proceedings of the Conference on Innovation & Technology in Computer Science Education (ITiCSE)*.

K. Karplus, and H. D. Thier. 1967. *A New Look at Elementary School Science*. Rand McNally & Co.

C. Köppe. 2011a. A pattern language for teaching design patterns (part 1). In *Proceedings of the European Conference on Pattern Languages of Programs (EuroPLoP)*, 21 pages.

C. Köppe. 2011b. A pattern language for teaching design patterns (part 2). In *Proceedings of the Conference on Pattern Languages of Programs (PLoP)*, 16 pages.

C. Kussmaul. 2016. Patterns in classroom activities for Process Oriented Guided Inquiry Learning (POGIL). In *Proceedings of the Conference on Patterns Languages of Programs* (PLoP).

C. Kussmaul. 2017. Patterns in classroom facilitation for Process Oriented Guided Inquiry Learning (POGIL). In *Proceedings of the Nordic Conference on Patterns Languages of Programs* (VikingPLoP).

C. Kussmaul and T. Pirmann. 2012. Guided inquiry learning for computer science. In *Proceedings of the Computer Science Teachers Association (CSTA) Conference*.

P. Lotlikar and R. Wagh. 2016. Using POGIL to teach and learn design patterns: A constructionist based incremental, collaborative approach. In *Proceedings of the IEEE Conference on Technology for Education (T4E)*.

R. S. Moog, F. J. Creegan, D. M. Hanson, J. N. Spencer, and A. R. Straumanis. 2006. Process-oriented guided inquiry learning: POGIL and the POGIL Project. *Metropolitan Universities Journal*. 17, 41-51.

R. S. Moog and J. N. Spencer, Eds. 2008. *Process-Oriented Guided Inquiry Learning (POGIL)*. American Chemical Society.

I. Nonaka. 1991. The knowledge-creating company. *Harvard Business Review*, 69, 6, 96-104.

J. Piaget. 1964. Cognitive development in children: Piaget development and learning. Journal of Research in Science Teaching 2, 176-186.

SourceMaking (not dated) Chain of Responsibility. Accessed May 28, 2019 from: https://sourcemaking.com/design_patterns/chain_of_responsibility

A. Straumanis, E. A. Simon. 2008. A multi-institutional assessment of the use of POGIL in Organic Chemistry. In Process Oriented Guided Inquiry Learning (POGIL). American Chemical Society, 226-239.

L. Williams, D. S. McCrickard, L. Layman, and K. Hussein. 2008. Eleven guidelines for implementing pair programming in the classroom. In *Proceedings of Agile 2008 Conference*.

Received May 2019; revised July 2019; accepted October 2019, withdrawn due to travel conflicts for conference date. Received July 2020, revised August 2020, accepted September 2020. Revised February 2021.

APPENDIX A.

This appendix contains a short version of a POGIL-style classroom activity on CHAIN OF RESPONSIBILITY.

Module 1: Exploration

Note: Each team is given a UML class diagram with a `Sender` class, an abstract `Receiver` class, and two concrete `Receiver` subclasses. Each team is also given Java source code for an interface (`Chain`), three implementing classes (`Negative`, `Zero`, `Positive`), a data class (`Number`), and a driver class (`TestChain`). Note that the Java source code implements the pattern but uses different names.

1. Consider the provided Java implementation of a model which includes interface, inheritance and set of operations by respective handlers. In the table below, show what result you expect for each input: (2 mins)

Test input:	0.1	22	45	-56
Expected output:				

2. Describe what the above program does.

3. What happens when you input a decimal number? Why? (2 mins)

4. What are the classes Negative, Zero and Positive used for? What request does the client have? (3 min)

5. How are these three classes linked together? (3 mins) 6. How is inheritance used? (2 mins)

7. Using the UML diagram, figure out the following roles in the program and what each does. (2 mins)
   a. Client b. Request Handler (receiver) c. Concrete Receivers (how many are there and who)

8. Draw the UML diagram for the above problem. Show the successor of each concrete handler. (2 mins)

9. How is the client kept independent of the other class implementation?
Are the request handlers and client coupled? Justify your answer. (3 mins)

10. How is the Request passed to the Handlers? (1 min) 11. Will it be handled by all Handlers? (1 min)

12. Does the client know which Handler has/is handled/handling its request? (1 min)

13. Explain what happens if the Request cannot be handled. (1 min)

14. What kind of scenarios of receivers do you observe in the program? Explicit or implicit receiver?
Justify your answer with a proper statement. (3 mins)

15. What is the Motive of this Model? (2 min)

16. Can you solve this problem without using this model? Justify your answer. (3 mins)

17. Can you think of another Design Pattern taught before that achieves the motive of this Model? (3 mins)

18. Can you think of another problem that can be solved with this Model?
Show how with respect to the UML class diagram and roles specification and description.

Module 2: Invention

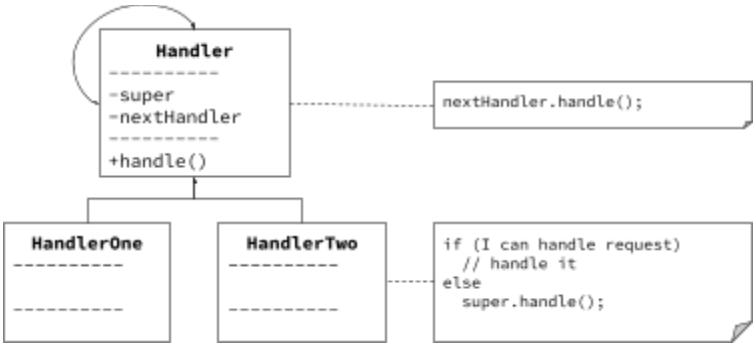
Note: This module is adapted from descriptions of CHAIN OF RESPONSIBILITY on SourceMaking.com (n.d.) and other websites.

Intent

- Avoid coupling the sender of a request to its receiver by giving more than one object a chance to handle the request. Chain the receiving objects and pass the request along the chain until an object handles it.
- Launch-and-leave requests with a single processing pipeline that contains many possible handlers.
- An object-oriented linked list with recursive traversal.

Problem

There is a potentially variable number of "handler" or "processing element" or "node" objects, and a stream of requests that must be handled. Need to efficiently process the requests without hard-wiring handler relationships and precedence, or request-to-handler mappings.



The pattern chains the receiving objects together, and then passes any request messages from object to object until it reaches an object capable of handling the message. The number and type of handler objects isn't known a priori, they can be configured dynamically. The chaining mechanism uses recursive composition to allow an unlimited number of handlers to be linked.

CHAIN OF RESPONSIBILITY simplifies object interconnections. Instead of senders and receivers maintaining references to all candidate receivers, each sender keeps a single reference to the head of the chain, and each receiver keeps a single reference to its immediate successor in the chain.

Make sure there exists a "safety net" to "catch" any requests which go unhandled.

Do not use CHAIN OF RESPONSIBILITY when each request is only handled by one handler, or when the client object knows which service object should handle the request.

Points to Remember:

1. The base class maintains a "next" pointer.
2. Each derived class implements its contribution for handling the request.
3. If the request needs to be "passed on", then the derived class "calls back" to the base class, which delegates to the "next" pointer.
4. The client (or some third party) creates and links the chain (which may include a link from the last node to the root node).
5. The client "launches and leaves" each request with the root of the chain.
6. Recursive delegation produces the illusion of magic.

Module 3: Application

In Module 1, you were given example code which implements this Design Pattern.

1. Try to implement this Design Pattern to solve the problem of Exception Handling.
(You can use the internet to understand exception handling and its processing.)

Based on the understanding and implementation answer the following questions:

2. Draw UML class diagram for this scenario.

3. Use the above UML diagram to figure out the following roles in the code and what each does. (2 mins)

    i. Client ii. Request Handler (receiver) iii. Concrete Receivers (how many are there and who)

4. Explain the flow of the program.

5. How inheritance will come into existence?

6. Can you think of any other Design Pattern taught till date that achieves the motive of this Model? (3 mins)

7. What request does the client have?

8. How the client is decoupled from the request handlers?

APPENDIX B.

This appendix shows the activity feedback questions, grouped under the appropriate research question.

#	Activity Feedback Question	RQ 1 Concept	RQ 2 Skill	RQ 3 Engage
1	An activity learning mode like POGIL is highly enjoyable as compared to traditional lectures.			Y
2	The questions in the activity guided my thinking and helped me understand key ideas.	Y	Y	
3	POGIL activity sheets were designed appropriately to match the learning objectives as mentioned in the syllabus.	Y		
4	Solving POGIL sheets helped in building self-confidence towards independent learning.		Y	
5	Solving POGIL sheets in groups brought forward the advantages of peer or cooperative learning.		Y	Y
6	POGIL activities helped in retaining and recollecting the concepts for a longer duration as compared to traditional lecture .		Y	
7	POGIL activities helps construct knowledge in the classroom and hence helped in better understanding of the concepts.	Y		
8	POGIL activities have a potential to improve student capacity to handle application based questions in the exams and help in scoring more marks.	Y	Y	
9	POGIL will be useful in teaching other concepts in Software engineering.		Y	Y
10	POGIL does not lead to learning, it is just a waste of time. It is boring.	Y		Y
11	The POGIL worksheets were very lengthy and difficult to understand.			Y
12	I used extra resources from internet/textbook. They helped me understand the concept deeper.	Y		
13	I would spend less time for this Design Pattern topic to study at home.			Y
14	I could not concentrate for the entire session.			Y
15	I would not recommend the use of POGIL for teaching design patterns.	Y		Y

APPENDIX C

This appendix shows the rubric used to evaluate student computer programming skills.

	Unsatisfactory (marks = 0)	Satisfactory (marks = 2)	Good (marks = 3)	Excellent (marks = 5)
Runtime / execution	• Does not execute due to errors. • User prompts are misleading or nonexistent. • No testing has been completed.	• Executes without errors. • User prompts contain little information, poor design. • Some testing has been completed.	• Executes without errors. • User prompts are understandable, minimum use of symbols or spacing in output. • Thorough testing has been completed.	• Executes without errors. • Excellent user prompts, good use of symbols and spacing in output. • Thorough and organized testing has been completed and output from test cases is included.
Coding standards and readability	• No name, date, or assignment title included. • Poor use of white space (indentation, blank lines). • Disorganized and messy. • Poor use of variables (many global variables, ambiguous naming).	• Includes name, date, and assignment title. • White space makes program fairly easy to read. • Organized work. • Good use of variables (few global variables, unambiguous naming).	• Includes name, date, and assignment title. • Good use of white space. • Organized work. • Good use of variables (no global variables, unambiguous naming).	• Includes name, date, and assignment title. • Excellent use of white space. • Creatively organized work. • Excellent use of variables (no global variables, unambiguous naming).
Program design (efficiency)	• A difficult and inefficient solution.	• A logical solution that is easy to follow but is not the most efficient.	• Solution is efficient and easy to follow (i.e. no confusing tricks).	• Solution is efficient, easy to understand and maintain.
Use of design pattern object modeling concepts to solve problem	• The student cannot apply object modeling concepts to write software applications with multiple classes.	• The student is able to apply object modeling concepts to write software applications with multiple classes.	• The student is able to sufficiently apply object modeling concepts to write software applications with multiple classes.	• The student is able to extensively apply object modeling concepts to write software applications with multiple classes.
Applying OO techniques to software packages	• The student cannot apply OO techniques to write software applications with multiple classes.	• The student is able to apply OO techniques in some key elements of software applications with multiple classes.	• The student is able to sufficiently apply OO techniques in some key elements of software applications with multiple classes.	• The student is able to extensively apply OO techniques in some key elements of software applications with multiple classes.

APPENDIX D.

This appendix contains a short version of a POGIL-style activity that develops some key ideas in knowledge management, as a motivation to learn about and use patterns. Sample answers are shown in italics. The questions follow learning cycles; the explore-invent-apply phase for each question is shown in parentheses, although this is not shown in a typical student activity.

Activity: Knowledge Management

Section A. Types of Knowledge

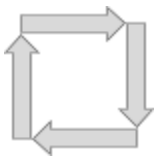
> Ikujiro Nonaka (1991) describes two types of knowledge:
> - knowledge that is formal, systematic, and easy to communicate
> - knowledge that is personal, hard to formalize, and hard to communicate
>
> He called the first **explicit** and the second **tacit**.

1.(*Explore/Invent*) Use the text above to decide which type (explicit, tacit, or both) is learned from:

a.	a cookbook or textbook	*explicit*
b.	talking to a master chef	*tacit / both*
c.	reading the rules to a sport (in a book or online)	*explicit*
d.	playing a sport	*tacit / both*
e.	watching a sport (in person or on video)	*tacit / both*
f.	a lecture or video	*explicit*
g.	an in-class activity or homework assignment	*both*

2. (*Invent*) 🔑 Michael Polanyi is quoted as saying "We know more than we can tell".
Explain what he meant, using the terms **explicit** and **tacit**.

Everyone has some tacit knowledge that they can't make explicit.

	to: → **tacit**		to: → **explicit**
from: **tacit** →	**socialize**, e.g.: master teaches apprentice		**articulate**, e.g.:
from: **explicit** →	**internalize**, e.g.:		**combine**, e.g.: summary of other reports

3. (*Explore*) Nonaka also describes 4 ways to **create or transform** knowledge, shown above.

a.	Combining knowledge **uses** which type?	*explicit*
b.	Combining knowledge **creates** which type?	*explicit*
c.	When an apprentice learns from a master, which way is used?	*socialize, T→T*
d.	When a chef creates a new recipe, which way is used?	*articulate, T→E*

4. (*Invent*) For each example below, decide which of the 4 ways is used.

a.	"I went through the manual and made a reference sheet."	*E→E, combine*
b.	"I keep making the same mistakes, so I made a list with what causes them."	*T→E, articiulate*
c.	"I do this task so often that I don't need to look up the steps any more."	*E→T, internalize*

5. (*Apply*) Explain why companies, educational institutions, and other organizations need good ways to convert knowledge from tacit to explicit.

Converting knowledge from tacit to explicit makes the knowledge easier to share.

Section B. Managing Knowledge

Companies and other organizations need ways to share knowledge across the organization; this is called **knowledge management**. Morton Hansen, Nitin Nohria, and Thomas Tierney (1999) studied how companies approached knowledge management. They found two quite different strategies:

- Focus on **content** (such as data, reports, checklists, templates, & spreadsheets) that could be used elsewhere in the organization. This strategy is **codification**.
- Focus on finding **people** with knowledge, expertise, and experiences that could be used elsewhere in the organization. This strategy is **personalization**.

1. (*Explore*) Use the information above to answer these questions. Which strategy:

a.	focuses more on documents?	*codification*
b.	focuses more on conversations?	*personalization*
c.	could find brochures used by other sales offices?	*codification*
d.	could help a team that has trouble making decisions?	*personalization*

2. (*Invent*) Which strategy (codification, personalization, both, neither) is used in:

a.	a shared drive of white papers and other marketing materials?	*codification*
b.	a staff directory with name, department, email, & phone?	*neither*
c.	website documentation (e.g. user guides, APIs, FAQs)?	*codification*
d.	a mailing list or discussion forum?	*both*

3. (*Invent*) Which strategy (codification, personalization, both, neither) would:

a.	be best for explicit knowledge?	*codification*
b.	be best for tacit knowledge?	*personalization*
c.	use the most storage space?	*codification (many files)*
d.	be easiest to design and organize?	*personalization (less data)*
e.	solve problems most quickly?	*codification*

4. (*Invent*) 🔑 In complete sentences, describe when to use which strategy.

Use codification for explicit, when problems are similar, and it's easy to adapt previous solutions.
Use personalization for tacit, when problems are different, and it's hard to adapt previous solutions.

APPENDIX E.

This appendix contains a short version of a POGIL-style activity to help students understand key parts of a pattern and how the parts are related. Each student would be given a copy of an example pattern in a familiar domain. The instructor would tell students to quickly read through the pattern, and then to work as a team to agree on the answer to one question at a time. As in Appendix D, sample answers are shown in italics, and the learning cycle phase is shown in parentheses.

Activity: Elements of a Pattern

1. (*Explore*) Refer to the example pattern(s) provided to answer these questions:

a.	How many sections are shown?	*6 (including name)*
b.	What font or style visually marks a pattern's name?	*small caps*
c.	What does the second bold sentence do?	*describe solution*
d.	What does the first bold sentence do?	*describe problem*

2. (*Invent*) Most written patterns contain a similar set of parts, often with labels like these:
Consequences Context Discussion Examples Forces Name Problem Solution
However, not every format or pattern has every element or labels every element.
Refer to the example pattern to choose the best label for each description below.

	Description	**Label**
a.	A concise statement of the problem fixed by the pattern.	*problem*
b.	A concise statement of how the pattern fixes the problem.	*solution*
c.	Situation(s) in which the pattern might be useful.	*context*
d.	A concise but evocative label for the pattern.	*name*
e.	Other related patterns.	*discussion*
f.	Other factors that may affect the problem.	*forces*
g.	Other factors that may be affected by the pattern.	*consequences*
h.	More detail on how the pattern can be used.	*examples*

3. (*Apply*) Which two elements could you combine to get a two or three sentence summary? (This is called a *patlet*.)

Combine the problem & solution to make a summary.

4. (*Apply*) If the labels (Content, Problem, Solution, etc) were removed, would the pattern make sense? Explain your answer.

Yes, probably - this seems like a natural flow.