Coding Strip: A Pedagogical Tool for Teaching and Learning Programming Concepts through Comics

Sangho Suh
University of Waterloo
Waterloo, Canada
sangho.suh@uwaterloo.ca

Martinet Lee
University of Waterloo
Waterloo, Canada
martinetlee@gmail.com

Gracie Xia
University of Waterloo
Waterloo, Canada
graciexia@gmail.com

Edith law
University of Waterloo
Waterloo, Canada
edith.law@uwaterloo.ca

***Abstract*—The abstract nature of programming makes learning to code a daunting undertaking for many novice learners. In this work, we advocate the use of comics—a medium capable of presenting abstract ideas in a concrete, familiar way—for introducing programming concepts. Particularly, we propose a design process and related tools to help students and teachers create coding strips, a form of comic strips that are associated with a piece of code. We conducted two design workshops with students and high school computer science teachers to evaluate our design process and tools. We find that our design process and tools are effective at supporting the design of coding strips and that both students and teachers are excited about using *coding strip* as a tool for learning and teaching programming concepts.**

***Index Terms*—comics; coding strip; visual language; computing education; concreteness fading; computational thinking**

I. Introduction

Initiatives to improve people's technical literacy for the 21st century has become a top priority in many countries around the world. Programming classes are now a mandatory part of many K-12 curricula and increasingly available online [1]. Despite growing interest, programming remains difficult to master because it involves concepts and procedures that are abstract [2], [3]. Programming requires learning a large number of unfamiliar syntax and arbitrary conventions. As Giraffa et al. [4] put it: "in learning programming, [students] need to imagine and comprehend many abstract terms that do not have equivalents in real life: how does a variable, a data type, or a memory address relate to a real-life object?" Finally, programming requires learners to be able to trace a sequence of execution steps, a task that novices find abstract [5] because procedures are often presented as an abstraction (e.g., loop [6]), with the sequence of steps hidden.

Prior work tried to introduce abstract concepts in programming by using concrete representations, such as manipulatives [8], visualizations [9], or metaphors [10], and elucidate abstract process with tools, such as tracing strategies [5] and visualization tools [11]. In this work, we explore how we can leverage the visual language of comics [12], [13] to make

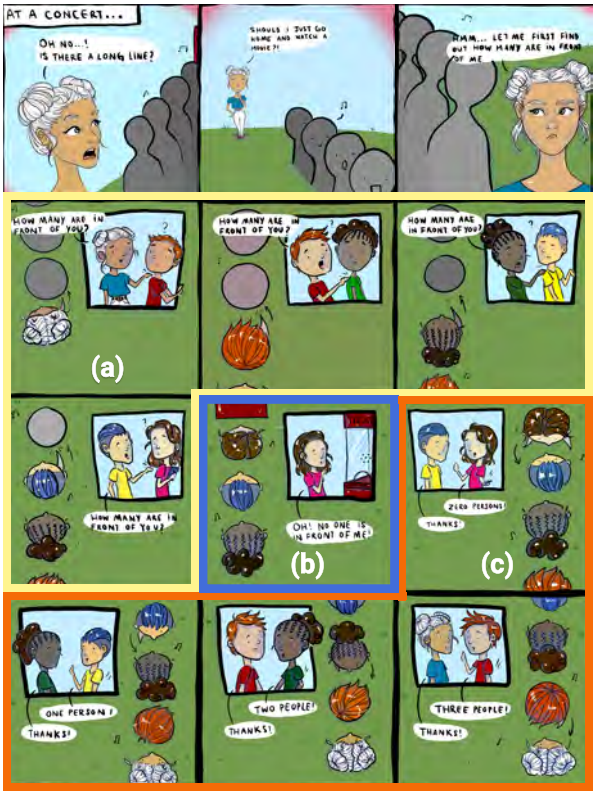


Fig. 1: Coding strip example on *recursion* using the *lens* [7] design pattern to illustrate the execution steps in the recursion code: (a) function call (forward recursion), (b) base case, and (c) return (backward recursion)

978-1-5386-5541-2/18/$31.00

©2020 IEEE

programming concepts and procedures more concrete. The sequential nature of the medium and its ability to express the complicated process and abstract concepts through visual storytelling [7], [14] provide reasons to believe that it is capable of making programming concepts and procedure more concrete. In fact, it has already been used successfully to introduce the scientific process and abstract concepts in many disciplines, such as science and math [15], [16], [17].

Several comic books—such as *Hello Ruby*, *Lauren Ipsum*, and *Secret Coders*—have been used to introduce computing in classrooms; however, these comics are typically formatted as storybooks without any correspondence to code, making transfer difficult. In this work, we are particularly interested in *coding strip*, which we define as a form of comics where the story is accompanied by a piece of code, to enable learners to see how a line of code can map to a meaningful action in real-life presented in the medium of comics. But ***how do we design it?*** And ***in what ways can it be used to support teaching and learning of programming concepts?*** To answer these questions, we formulated the design process and tools to support the creation of coding strips. Then we conducted two design workshops, one with students and the other with teachers, to evaluate our design process and tools, as well as to understand how students and teachers perceive and see it being used to support learning and teaching of programming.

II. Background

A. Making Programming More Concrete

Many solutions have been proposed to make coding more concrete and approachable. CS unplugged is a kinesthetic learning method in which students learn computing concepts through hands-on activities without the use of computers. Scratch [8], a block-based programming language, minimizes syntax errors and the need to memorize conventions by replacing text with familiar LEGO-like constructs. Other research explores tangible languages in which physical objects function as programming constructs to make programming more hands-on [18]. Interactive visualizations, e.g., Python Tutor, enable learners to step through each line of code and watch the program state change, showing the sequence of the process and what each line does to program states [11]. Visual learning environments, e.g., Alice [19], incorporate storytelling into coding to help link abstract concepts to real-life referents. Our work builds on these works by contributing new knowledge about ways to incorporate comics into computing education.

B. Linking Concrete with Abstract

While these visual, block-based, and tangible programming approaches help to lower the barrier to entry into programming, learners need to eventually transition to text-based programming [20]. Several transition methods have been proposed [21], [22], [23]; while their methods may vary in specifics, researchers agree that explicitly linking the concrete with abstract programming benefits transfer [22], [24]. Based on this idea, we combined the idea of concreteness fading and comics to establish a tighter connection between the concrete (comic) and abstract (code), as shown in Fig. 1. Concreteness fading [25] is a well-established learning progression technique, involving two or more stages, where a concept is illustrated with decreasing levels of concreteness, while its core idea(s) remain intact across stages [24], [13]. Although widely adopted in math education, interest in applying concreteness fading to computing education has emerged only recently. Arawjo et al. [26] implemented a concreteness fading strategy in which programming constructs (e.g., equality check) initially appear as familiar physical manipulative (e.g., reflective glass) then gradually fade into code (e.g., the expression, "=="). Trory et al. [27] explored the potential for augmented reality to support concreteness fading in introducing computing concepts, such as internet routing.

C. Using Visual Language of Comics

Broadly defined as sequential art [28] and juxtaposed images and words [12], comics have recently gained in popularity as a tool for teachers as they began to notice the power of this medium and its educational benefits, including the ability to ground abstract contents in the real world and foster interest, creativity, and discussion [29], [30], [31], [32]. Its popularity is evidenced by many studies and online resources that report many ways it has been incorporated into teaching practices [33], [34], [35]. For instance, teachers use them to introduce a topic and motivate students; students create comics to illustrate concepts they have learned and use them to teach their peers, which encourages students to reflect on the learned concepts and ground them in familiar situations [36], [37].

Many tools and techniques have been proposed to lower the barrier to creating comics. Commercial tools, such as Pixton [38], are widely used by universities and schools to support students engaging in comic creation activities in the classrooms and teachers designing learning materials for subjects, such as math, science, business, and history [32], [39], [15], [36], [40].

Recently, research communities have also begun to study the visual language of comics actively, exposing its design dimensions [28], [14] and advantage of this medium over other media such as video [41] and infographic [17]. The visualization research communities are seeing *data comics*—a form of comics that communicates insights in data with the visual language of comics—emerge as a genre of its own [42]. Our work is inspired by these works that contribute new understandings of how the visual language of comics can be used in their domain [14], as well as tools [43] and design analysis [7] to facilitate its use.

D. Supporting Design Process with Ideation Cards

Ideation cards are design tools that proved useful at supporting the design process [44], [45]. As formulating initial ideas and exploring the design space is a fundamental process of the design process [46], many design studies have used ideation cards to support their design process [47], [48], [49], [50]. For instance, Mora et al. [51] developed *Tiles*, a set of ideation cards to help non-experts engage in creating ideas

for IoT projects, along with a design board to scaffold the design process. They tested the effectiveness of their tools, ideation cards and design board, by running workshops and asking the participants to assess the perceived usefulness of the tool in supporting their design process. We also developed ideation cards and design board as our design supporting tools and tested their effectiveness in the same manner, which is a standard methodology other studies proposing design process and ideation cards also used [44], [47], [48].

III. Coding Strip: Example

Fig. 1 shows a coding strip example on *recursion* created by one of the authors. The labels (i.e., (a), (b), (c)) and boxes surrounding the panels and lines of code specify the mapping between the panels and the lines of code, with the example consisting of three main parts: (a) forward recursion, (b) base case, and (c) backward recursion. In Fig. 1, the story begins with a girl arriving at a concert. She debates returning home because she suspects the waiting line may be too long. She decides to find out how many are in front of her, so she asks a man in front of her a question: "how many are in front of you?" The man in front of her asks the same question to a girl in front of him: "how many are in front of you?" This continues until it finally reaches a girl (b) who does not have anyone in front of her. She turns back and tells the boy behind her that there is no one ("zero people") in front of her. Then he turns back and tells the girl behind him: "one person," a number derived by adding one to the number he was given by a girl in front of him. The girl behind him does the same. This continues until it reaches the girl who initiated the question. Although not shown here, the story ends with the girl saying, "That's not too bad! I will stay then!"

Initial feedback towards this coding strip from piloting it with 28 university undergraduate students with no prior knowledge in programming provided confidence to the authors that introducing *recursion* with comics first and then its closely mapped code can help even novice learners follow the sequence of recursion code shown in Fig. 1 as well as understand what each line of code does. This preliminary finding, in part, motivated this study, which is to support the creation of coding strips.

TABLE I: Concepts used in our study. Bolded are the concepts that workshop participants chose for their coding strips.

Constructs	Data Structures	Algorithms	Problem Solving Techniques
Variable	Array	Selection Sort	Greedy
Boolean	Linked List	Insertion Sort	**Divide & Conquer**
Condition	Queue	Merge Sort	**Recursion**
Counted Loop	Stack	Bubble Sort	
Conditional Loop	Tree	Linear Search	
Function	Graph	Binary Search	
	Dictionary		

IV. Coding Strip: Design Process & Tools

This section describes how we developed the design process and tools for guiding learners or teachers to create comic strips that illustrate programming concepts.

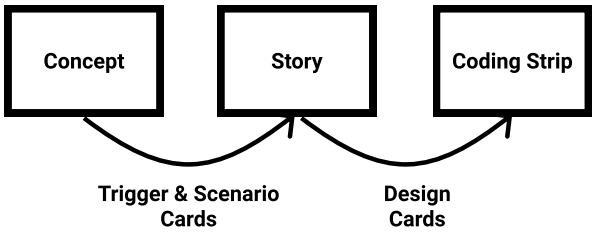


Fig. 2: Three high-level stages in the design process, with trigger & scenario cards supporting transition from concept to story and design cards from story to coding strip.

A. Design Process

To formulate the design process, we first identified the main concepts taught in introductory computer science courses in high school and university (Table I) as outlined in curriculum guidelines [52], [53]. Then, the authors followed the common practice [14], designing in parallel at least two comic strips for concepts in each category in Table I and until all the concepts were used. Through this experience, we identified three high-level stages in our own design process, as shown in Fig. 2—**concept formulation**, where we distill the core ideas behind a given concept, **story development**, where stories are invented to illustrate these core ideas, and finally **comic illustration**, where the stories are sketched out in the form of a comic strip.

From this, we created a design board (Fig. 3), an empty canvas with 5 sections corresponding to the steps of the

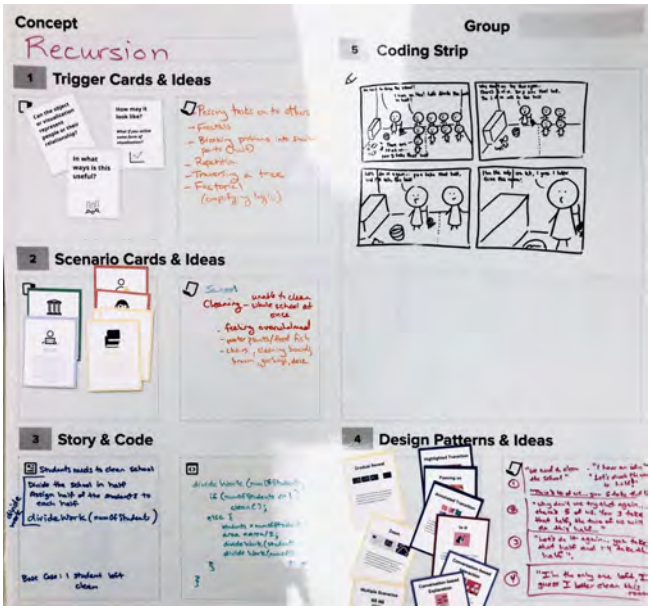


Fig. 3: Design board with coding strip on *recursion* made by a teacher group at our workshop

ideation process: (1) trigger: reflect on the important properties of a programming concept, (2) scenario: generate many stories to illustrate the concept, (3) story & code: select one of the stories and write the corresponding code, (4) design: select an appropriate set of design patterns for the comic strip, (5) draw: produce the actual coding strip.

B. Design Tools

To help designers brainstorm effectively, we created a set of *ideation cards* and introduced them into certain stages of the design process.

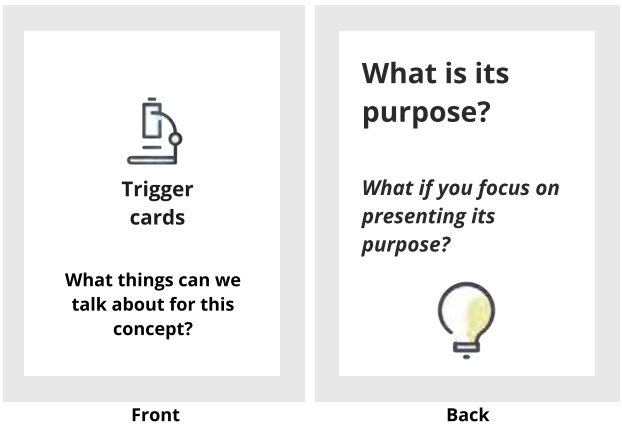
We introduced 16 **trigger cards** (Fig. 4a), which pose questions to designers to help them remember the properties of a concept and brainstorm ways to explain it. For example, for any given concept, one can reflect on "what it is useful for," "in which context it is used," its "particular strength or weakness," or "how it compares with a similar concept" (e.g., linear vs. binary search). Trigger questions were generated by the authors writing down potential explanations for each programming concept in Table I, clustering similar explanations, and then formulating questions that would trigger such explanations. For instance, for the concept of *variable*, the explanations include: "it is used for storing data" and "it is like a box that you can put something into"; for *conditional*, the explanation is "it is used for decision making." These explanations were clustered under the theme of "purpose," giving us the trigger question "What is its purpose?"

For brainstorming stories/scenarios, we created 19 **scenario cards** (Fig. 4b) that ask designers to think about the four essential elements of story: setting, character, action, and conflict. *Setting* cards provide ideas for when and where the story takes place; *Character* cards for the protagonist or antagonist in the story; *Action* cards for actions undertaken by characters or objects; and *Conflict* cards for conflicts that drive the plot of the story.

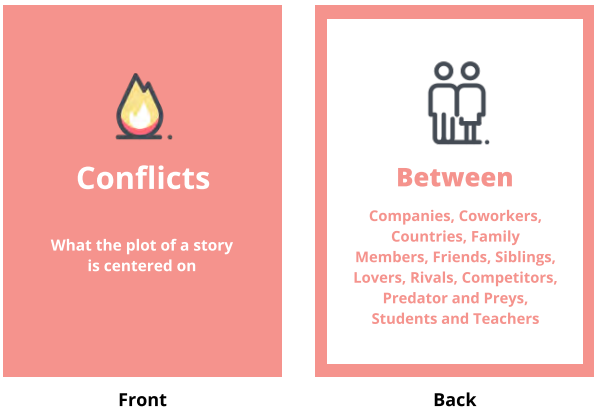
Finally, for creating the comic strip itself, we introduced 30 comic **design cards** (Fig. 4c), based on design patterns seen in data comics [7] and comic strips, under five categories: sequence, selection, repetition, explanation, and presentation. *Sequence* patterns help highlight the transition from one step to another in the context of program execution. *Selection* patterns provide ways to introduce conditional and different outcomes. *Repetition* patterns help illustrate conditional and counted repetitions (i.e., while and for loop, respectively). *Explanation* patterns showcase direct and indirect ways to explain a concept. *Presentation* patterns (e.g., Fig. 4c) cover various presentation techniques that can make reading more engaging and the delivery of intended message more effective.

V. Design Workshops

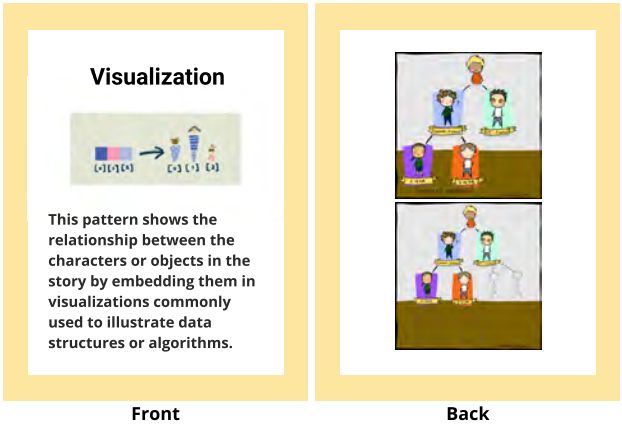
We conducted two three-hour workshop studies: one (W1) with a group of undergraduate and graduate students, and the other (W2) with high school computer science teachers, to ensure a balance in perspective and feedback.



(a) Trigger card



(b) Scenario card



(c) Design card

Fig. 4: Examples of ideation cards

A. Participants

For W1, we recruited 13 university students (5M, 8F; age: 17-29, mean 21), with basic knowledge in programming concepts and from a variety of programs (4 Environment, 3 Science, 1 Computer Science, 1 Engineering, 3 Mathematics, 1 Arts). Most mentioned that they have no experience (8) or very little experience (3) teaching computer science (e.g., programming), and similarly in terms of teaching experience

in other areas. Participants were split in terms of their level of confidence in drawing (4 Not confident; 5 Somewhat confident; 4 Confident), and confidence in designing comics for teaching coding concepts (5 Not confident; 7 Somewhat confident; 1 Confident). Participants were recruited via posters and SONA (a research participant recruitment platform), and provided $60 for participating in the workshop.

For W2, we had 6 participants (2M, 4F). Among these participants, only three teachers (age: all 45) answered the pre-study survey questions. The three teachers specified that they had 3-5 years, 5+ years, 1-3 years of teaching experience, respectively. As for teaching experience in computer science (e.g., programming), one teacher had no experience, while two teachers had 5+ years and 3-5 years of experience. The three teachers rated their own teaching ability as 4 out of 5. All three participants were not confident about their ability to design comics for teaching coding concepts. W2 was conducted at a computer science educator conference hosted by the university; as such, participants were not paid.

B. Procedure

Both workshops consisted of two sessions (90 minutes each), with a 15-min break in between. After the consent forms and pre-study questionnaire (10 min), we introduced the idea of *coding strip* using an example, shown in Fig. 5 (10 min).

Then we engaged in two warm-up activities (10 min each), which involved (1) several iterations of sketching simple stick figures and (2) translating story to code. After the warm-up, each group of two to four participants selected a programming construct from a list (Table I), and proceeded to design a coding strip about the concept using the design board and ideation cards. In the second design session (50 min), groups were free to choose any programming concepts, including problem solving ones. The workshop ended with a discussion (35 min) and post-study survey comprised of short answer, multiple-choice, and five-point Likert scale questions (5 min).

For both workshops, we had two artists assist participants with sketching. They did not engage in the design activity but intervened when asked to help. We video-recorded both workshops with the consent of participants for accurate transcription of their responses and analysis of their interaction. The authors also made observations throughout the workshops.

VI. Results

This section presents the analysis of the effectiveness of our design process and tools, and reports how students and teachers perceive and see coding strips being used to support learning and teaching. To ensure anonymity, we refer to our student participants from W1 as S1...S13, and teacher participants from W2 as T1...T6.

A. Effectiveness of Ideation Cards

Ideation cards were central to facilitating groups in generating coding strips. Thus, to understand their effectiveness, we examined (1) how many ideas each card was able to help generate, (2) how they were used, and (3) what ideas were generated from them.

```
int i = 0;
String[ ] lilypad = {"green", "green", "green", "rock", "red"};
while (lilypad[i] != "red") {
    frog.jump();
    if (lilypad[i] == "rock") {
       frog.confused();
    }
    i++;
}
```

Fig. 5: Example used to introduce the idea of *coding strip* during the workshops. Participants were asked to guess the underlying concepts/code from the comic strip before seeing its code, to show how comics can be used to introduce concepts and start a discussion.

1) Trigger Cards: They helped generate more than one idea for each card (# of cards: M=4.2, SD=1.7; # of ideas: M=6.7, SD=2.9). Across both workshops, the most frequently used trigger cards were "How would you explain this to a five-year-old?" and "When should you use this?" The "Can you act it out?" card was never used possibly because this question asks a person to embody a concept, and such an approach is used mainly when teaching data structure and algorithm concepts (e.g., sorting). The "How would you explain this to a five-year-old?" card was used by groups to generate stories containing objects and context familiar to children. For instance, a group working on the concept *boolean* used candy in their story, while another group working on *conditional loop* came up with an idea of asking a child to "check series of toys and report whether the color is red."

2) Scenario Cards: Each scenario card also generated more than one idea on average (# of cards: M=3.6, SD=1.7; # of ideas: M=5, SD=1.7). The groups usually had five story ideas to choose from, suggesting that scenario cards were quite effective at supporting ideation. Our analysis also reveals that the groups had different preferences on how to use scenario cards. While participants were encouraged to generate as many story ideas as possible during the ideation stages, only half of the

Fig. 6: Design workshops with groups of students (left) and high school computer science teachers (middle & right).

groups followed this advice while the other half used scenarios cards (i.e., *Setting*, *Character*, *Action*, *Conflict*) to add details to the story. For instance, one group generated "playing sport" as an idea for Context, "soccer ball" for Object, "pass, shoot, defend" for Action, "between teammates & competitors" and "unable to decide what action to take" for Conflict. The group used these to form a story about when to pass, shoot, and defend in the soccer field, in order to illustrate the concept *conditional*. The most frequently used scenario cards were Unable To and Institution. With the Unable To card, a student group working on *recursion* came up with the conflict: "unable to graduate due to a shortage of completed terms." One teacher group working on *conditional* used the Unable To card to generate the conflict idea "unable to remember," then used the Work card to come up with the action "deliver," and subsequently produced scenario: "Delivery man is unable to remember the house number he is delivering food to, so he just stands outside in the rain with his soggy pizza."

3) Design Cards: Several design cards (M=5.2, SD=2.4) were used each time to support design. For design ideas, since it required the participants to sketch, which they had varying levels of confidence on, we saw different results. Some put down sketch ideas, as shown in Fig. 7; some wrote script (e.g., dialogue) for each panel, as shown in Fig. 3; some did not sketch and went straight into drawing the actual coding strip. The most commonly used design cards were What-If (*Selection*), Multiple Scenarios (*Selection*), and Assign (*Sequence*). Fig. 7 shows the comic produced by one group working on the concept *boolean*; they used the Comparison, Multiple Scenarios and Before/After cards to generate a story comparing two possible scenarios for a fictional character Bob Lean (a play on word *boolean*): "Bob Lean is at home. It is 2:02 am, and he's playing games instead of sleeping. Next day, he can't go to class because he's sleep-deprived. Bob Lean is at home. It's 9:30 pm. Bob is well asleep. The next day he wakes up and goes to class/study."

B. Generated Stories & Comics

Participants, in general, were able to generate interesting coding strips using ideation cards. For example, one student group working on the concept *conditional* used scenario cards—such as Food, Unable To, Planets—to generate different stories and corresponding code, such as "is this pizza? yes/no", "if sleep == false: cannot study, else: study" and

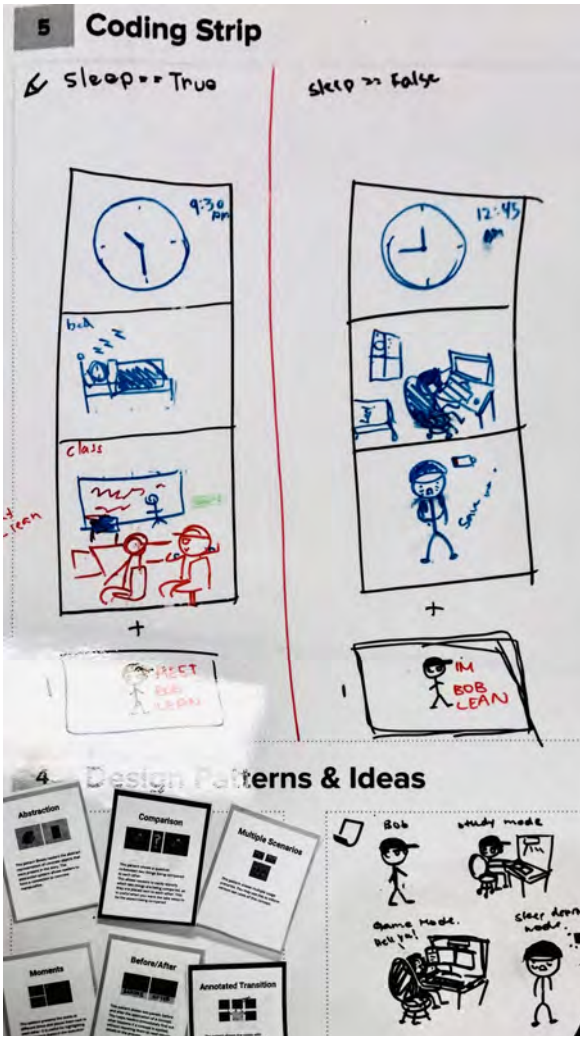


Fig. 7: Design ideas generated by design patterns and the resulting coding strip on *boolean* by a student group

"if pluto != planet specs; return false." As another example, one teacher group working on *recursion* used 3 trigger cards: "Can the object or visualization represent people or their relationship?", "How may it look like?", and "In what ways is this useful?" From these triggers, they generated ideas about the properties of *recursion*, such as "passing tasks on to others," "breaking problems into smaller points (half)," "repetition," and "traversing a tree." Then, they selected 7 scenario cards, and used the Unable To card to generate the "unable to clean" conflict, the Institution card to select "school" as the setting of their story, and produced the scenario "unable to clean the whole school at once." After writing down the story and its code, they selected 9 design cards and without sketching them out, wrote a script for each panel and asked one of the artists to draw the coding strip for them. The final product is shown in Fig. 3.

C. Panel-to-Execution Mapping

Analyzing the 18 coding strips generated from the two workshops, we identified 1-to-1, 1-to-many, and many-to-1

patterns in terms of how panels were mapped to execution steps. In 1-to-1 mapping, the most frequently used pattern, each panel describes an action or values (state) at each execution step (e.g., Fig. 1, 5). Examples of comics that used 1-to-many mapping include flowcharts that represent multiple execution steps within a loop, or a set of panels that illustrate only the beginning, middle, and end of a loop. Finally, we identified the use of many-to-1 mapping in a story about *counted loop* where a child colors every object in the household. The group allocated three panels for a single step, e.g., each panel showing more areas of the sofa being gradually covered in crayon marks. In general, 1-to-1 mapping may benefit novice learners' procedural understanding by spelling out the action step by step; 1-to-many mapping may help learners better grasp high-level concepts by creating meaningful, comprehensible abstractions for execution steps; finally, many-to-1 mapping may improve conceptual understanding by providing more details about a concept's specific properties by leveraging multiple panels to explain a single execution step.

D. Ease of Design Process

Contrary to our initial skepticism, participants did not require much help in producing the comics. Every student group in W1 designed their coding strip without help from the artists; only once did one group request an artist to help them sketch a minor part of their design. On the contrary, most teacher groups asked the artists to help sketch their coding strips. However, their request for help seems to stem from the desire for a more polished final product. In general, participants found the design process and tools useful (M=4.1, SD=0.8 for W1, and M=4.2, SD=0.8 for W2), the instructions relatively easy to follow (M=3.8, SD=0.5 for W1 and M=3.6, SD=0.9 for W2), and the process of designing comics very engaging (M=4.2, SD=1 for W1, M=4.3, SD=1 for W2).

Their confidence in their own ability to produce coding strips also improved. Eleven students indicated that they feel more confident after this design session, with 2 participants, who were confident and somewhat confident prior to the session, remaining the same as before. At W2, 5 teachers said that they feel more confident after the session, with 1 participant who was not confident feeling the same as before.

E. Perceived Utility for Teaching and Learning

In the post-study survey, we asked—"How useful do you think comics is for teaching/learning compared to before the design session?"—to assess whether our design process changed their view of coding strips. Eight student participants said that coding strip seems more useful after the design session, with 5 students, who rated the usefulness highly (M=4, SD=1) before the session, feeling the same as before. At W2, 4 teachers stated that it seems more useful, with 2 participants perceiving usefulness the same as before. A similar trend was observed for the perceived utility of *coding strip* for learning.

When probed about the reasons for their desires to learn with *coding strip*, participants said that it is "fun," "engaging," "easier to understand as it relates to everyday life," and "offers visual representation." One student who said "no" explained that he thinks "it makes concepts harder to understand." A teacher participant who said "not sure" explained that she is not familiar yet with the design process; she may have misunderstood our question as she was enthusiastic about *coding strip*: she asked us to come to her school and run the same workshop with her students. Two other teachers also expressed a desire to access and use our design board and ideation cards in their classrooms. As for teaching with *coding strip*, teachers (5 "yes"; 1 "not sure") liked the idea, while students (6 "yes"; 7 "not sure") were divided.

1) Classroom Environment and Teaching Time: Almost all participants saw *coding strip* as having a positive impact on the classroom atmosphere. Many believed that coding strips would lead to a "fun, light-hearted environment" for learning. Several participants also noted the potential of the design activity to "create a more open and active atmosphere," (S11) "prompt others to discuss more about ... the concept" (S8) and "encourage engagement among students" (S10). T1 indicated that she favors this over traditional instructional method, as this design activity is "better than just listening to the teacher" and students "get ideas from their peers and get to talk about [programming] concepts."

Many participants thought that coding strips would help reduce learning and teaching time since the concepts are grounded in real life and thus easier to understand. However, participants who incorrectly interpreted teaching time as *preparation* time expressed concern over how *coding strip* may increase teaching load. Students thought that manual creation of the comics would be time-consuming; as S8 said, "It needs significant time from teachers' end to create one, but it will make it easy for students to learn."

Two student participants stressed that "proper" comics and "relatable" content are necessary prerequisites to reducing the time it takes to learn. During the workshop, one teacher group working on the concept *function* actually focused extensively on ensuring that the story is relevant to high school students. In the end, they generated the story about a student searching for wifi access: "Student wants to snapchat, but there is no wifi. While there is no wifi, move to other locations until a signal is obtained, then snap!"

2) Specific Learning Benefits: Participants mentioned that coding strip helps students "[see] a visual representation" (S11, T2) and "connect CS to everyday life" (T6), which in turn contribute positively to comprehension (T2). One teacher participant suggested that coding strips would help "introduce computational thinking." Another teacher thought that the design activity would encourage students to develop "higher-level thinking about [how to] design [program before embarking on coding]." Many participants at W1 mentioned the positive effect on retention, through the act of "associating concepts with pictures and characters" (S12) and "breaking [broad and difficult concepts] down into small parts" (S5).

Many participants praised the power of comics to engage, motivate, and interest students in learning CS. T4 suggested

that "time invested in [the design activity] will have a lasting impact on students' attitude toward CS education." Several participants (S9, S13, T2) attributed this to the medium itself, saying "the fact they are comics ... make them seem more fun and would entice the students to read them." Other participants saw *coding strip* as "a new way to present the information" (S6) and "encourage students to pay attention" (S12).

F. Use Case Scenarios

Overall, designing coding strips seems to be an engaging activity, perceived to be useful in both learning and teaching contexts as well. One teacher participant said she wants to use the design activity to get students to think about the design of their program prior to coding, explaining that this will help them "reduce mistakes" and make them realize that "the program involves thinking and understanding the problem before touching a keyboard" (T3). This participant also suggested formatting the design activity for "vertical learning," a learning model that forces students to think on their feet while engaging in creative work and collaborating with other students in a group [54]. Both student and teacher participants suggested using coding strips to introduce concepts as they would help better understand and remember difficult concepts. As such, several participants wanted coding strips to be available as a complementary resource in textbooks, "[for explaining] abstract data structures and algorithms [that] are hard to visualize through code" (S1). Several ideas for assessing students' learning with *coding strip* were also discussed. S2 mentioned that at times it would be more instructive to "reverse the process—showing [coding] strips and [have students] writing the code," which echoed other participants' comments on the usefulness of engaging students in guessing the coding strips' underlying algorithm and code. S1 and S13 also suggested introducing a story with a missing piece at the end so that "a student has to come up with an end" (S1) to "ensure that the student has [correct procedural understanding]" (S13).

VII. Discussion

A. Extensions and Limitations

Several suggestions emerged from the workshops. One recommendation is to include a "peer feedback" section on the design board to allow groups to interact with each other by leaving feedback on each other's work. Another suggestion is to develop "a unified comic language" which would allow people to see "similar design everywhere." Two participants (S1, T1) voiced the desire to have a system with pre-defined templates that enable users to create coding strips without having to manually draw them. One participant (S12) explained that in order to "help maximize learning," the design activity should help students "[focus more] on creating the ideas for the coding strip [than] on drawing." One participant (T5) suggested labeling ideation cards with level of difficulty.

Participants also noted several limitations in *coding strip*. One participant was skeptical of the ability of *coding strip* to portray hard concepts: As S5 said, "I believe that this should only be used for beginner computer science concepts, as it would be more difficult to express more advanced and abstract concepts through this medium." This is probably because he did not get to see, or design, coding strips for advanced concepts, such as *merge sort*, like the authors did while creating coding strips for every concept in Table I. Some participants (S3, S5) expressed uncertainty about engaging novice learners in the design activity, as the activity might rely too much on the novice learners' "capability to make the correct representation of a concept." Two student participants noted (S3, S4) that coding strips might be more effective for students with a certain learning style, e.g., those who "learn better with visual representations." While this concern is valid, the variety of use case scenarios, such as using the design activity to teach collaborative skills, suggests that there are ways for *coding strip* to benefit various types of students. One student participant also mentioned that in order for students to participate in the design activity, they must have an adequate level of understanding of the concepts. This is aligned with another teacher's feedback that our design process would work well for her 12th grade students, but not for the 11th grade students as they have not learned all the concepts yet.

B. Extensibility of Ideation Cards and Design Process

Golembewski et al. and Chung et al. noted that another use of ideation cards is defining the design space [44], [55]. While this was not the focus of our study, our design cards can certainly help inform the design space for programming concepts. Additionally, our design process in which we use trigger and scenario cards to generate stories for a concept may be useful for any domain that wants to use stories as analogies to explain any of its abstract, complicated concepts. Several teachers at the workshop noted the broad applicability of trigger and scenario cards, saying that they would like to use them when teaching other subjects, such as math and science, for brainstorming ways to explain complicated concepts.

VIII. Conclusion and Future Work

In this work, we explored how to design *coding strip*, comic strip for teaching and learning programming concepts. Findings from our two design workshops show that both students and teachers are enthusiastic about the prospect of applying *coding strip* to the current pedagogical practices, and our design process and ideation cards are able to support the design process, even without the support of artists. In addition to the design process and supporting tools—which are available online [56], our work contributes new understandings about what design considerations (e.g., panel-to-execution mapping) exist and how the visual language of comics can be used to support the teaching and learning in programming. Building on this work, we are testing various use cases suggested during the workshop in an introductory computer science course, and are planning to report our results once our study is over. As another future work, we are planning to develop an authoring tool based on the design process and ideation cards developed during this study to further facilitate the creation, sharing, and use of *coding strip*.

IX. Acknowledgment

This research was funded by Learning Innovation and Technology Enhancement (LITE) Grant at the University of Waterloo. We would also like to thank Edward Lank, workshop participants, and reviewers for their feedback and suggestions for improvements.

References

[1] A. Yadav, H. Hong, and C. Stephenson, "Computational thinking for all: pedagogical approaches to embedding 21st century problem solving in k-12 classrooms," *TechTrends*, vol. 60, no. 6, pp. 565–568, 2016.

[2] A. Yadin, "Reducing the dropout rate in an introductory programming course," *ACM inroads*, vol. 2, no. 4, pp. 71–76, 2011.

[3] R. P. Medeiros, G. L. Ramalho, and T. P. Falcão, "A systematic literature review on teaching and learning introductory programming in higher education," *IEEE Transactions on Education*, vol. 62, no. 2, pp. 77–90, 2018.

[4] L. M. Giraffa, M. C. Moraes, and L. Uden, "Teaching object-oriented programming in first-year undergraduate courses supported by virtual classrooms," in *The 2nd International Workshop on Learning Technology for Education in Cloud*. Springer, 2014, pp. 15–26.

[5] B. Xie, G. L. Nelson, and A. J. Ko, "An explicit strategy to scaffold novice program tracing," in *Proceedings of the 49th ACM Technical Symposium on Computer Science Education*, ser. SIGCSE '18. New York, NY, USA: ACM, 2018, pp. 344–349. [Online]. Available: http://doi.acm.org/10.1145/3159450.3159527

[6] S. Derus and A. Z. M. Ali, "Difficulties in learning programming: Views of students," in *1st International Conference on Current Issues in Education (ICCIE 2012)*, 2012, pp. 74–79.

[7] B. Bach, Z. Wang, M. Farinella, D. Murray-Rust, and N. Henry Riche, "Design patterns for data comics," in *Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems*, ser. CHI '18. New York, NY, USA: ACM, 2018, pp. 38:1–38:12. [Online]. Available: http://doi.acm.org/10.1145/3173574.3173612

[8] M. Resnick, J. Maloney, A. Monroy-Hernández, N. Rusk, E. Eastmond, K. Brennan, A. Millner, E. Rosenbaum, J. S. Silver, B. Silverman *et al.*, "Scratch: Programming for all." *Commun. Acm*, vol. 52, no. 11, pp. 60–67, 2009.

[9] E. Fouh, M. Akbar, and C. A. Shaffer, "The role of visualization in computer science education," *Computers in the Schools*, vol. 29, no. 1-2, pp. 95–117, 2012.

[10] J. P. Sanford, A. Tietz, S. Farooq, S. Guyer, and R. B. Shapiro, "Metaphors we teach by," in *Proceedings of the 45th ACM technical symposium on Computer science education*, 2014, pp. 585–590.

[11] P. J. Guo, "Online python tutor: embeddable web-based program visualization for cs education," in *Proceeding of the 44th ACM technical symposium on Computer science education*. ACM, 2013, pp. 579–584.

[12] S. McCloud, "Understanding comics: The invisible art," *Northampton, Mass*, 1993.

[13] S. Suh, "Using concreteness fading to model & design learning process," in *Proceedings of the 2019 ACM Conference on International Computing Education Research*, 2019, pp. 353–354.

[14] B. Bach, N. Kerracher, K. W. Hall, S. Carpendale, J. Kennedy, and N. Henry Riche, "Telling stories about dynamic networks with graph comics," in *Proceedings of the 2016 CHI Conference on Human Factors in Computing Systems*, ser. CHI '16. New York, NY, USA: ACM, 2016, pp. 3670–3682. [Online]. Available: http://doi.acm.org/10.1145/2858036.2858387

[15] M. Tatalovic, "Science comics as tools for science education and communication: a brief, exploratory study," *Journal of Science Communication*, vol. 8, no. 4, p. A02, 2009.

[16] M. J. Green and K. R. Myers, "Graphic medicine: use of comics in medical education and patient care," *Bmj*, vol. 340, p. c863, 2010.

[17] Z. Wang, S. Wang, M. Farinella, D. Murray-Rust, N. Henry Riche, and B. Bach, "Comparing effectiveness and engagement of data comics and infographics," in *Proceedings of the 2019 CHI Conference on Human Factors in Computing Systems*. ACM, 2019, p. 253.

[18] T. S. McNerney, "From turtles to tangible programming bricks: explorations in physical language design," *Personal and Ubiquitous Computing*, vol. 8, no. 5, pp. 326–337, 2004.

[19] S. Cooper, W. Dann, and R. Pausch, "Alice: a 3-d tool for introductory programming concepts," *Journal of Computing Sciences in Colleges*, vol. 15, no. 5, pp. 107–116, 2000.

[20] L. Moors, A. Luxton-Reilly, and P. Denny, "Transitioning from block-based to text-based programming languages," in *2018 International Conference on Learning and Teaching in Computing and Engineering (LaTICE)*, 2018, pp. 57–64.

[21] M. Homer and J. Noble, "Combining tiled and textual views of code," in *2014 Second IEEE Working Conference on Software Visualization*. IEEE, 2014, pp. 1–10.

[22] M. Kölling, N. C. Brown, and A. Altadmri, "Frame-based editing: Easing the transition from blocks to text-based programming," in *Proceedings of the Workshop in Primary and Secondary Computing Education*. ACM, 2015, pp. 29–38.

[23] D. Franklin, C. Hill, H. A. Dwyer, A. K. Hansen, A. Iveland, and D. B. Harlow, "Initialization in scratch: Seeking knowledge transfer," in *Proceedings of the 47th ACM Technical Symposium on Computing Science Education*, ser. SIGCSE '16. New York, NY, USA: ACM, 2016, pp. 217–222. [Online]. Available: http://doi.acm.org/10.1145/2839509.2844569

[24] E. R. Fyfe and M. J. Nathan, "Making "concreteness fading" more concrete as a theory of instruction for promoting transfer," *Educational Review*, vol. 71, no. 4, pp. 403–422, 2019.

[25] S. Suh, M. Lee, and E. Law, "How do we design for concreteness fading? survey, general framework, and design dimensions," in *Proceedings of the 19th ACM Conference on Interaction Design and Children*, 2020.

[26] I. Arawjo, C.-Y. Wang, A. C. Myers, E. Andersen, and F. Guimbretière, "Teaching programming with gamified semantics," in *Proceedings of the 2017 CHI Conference on Human Factors in Computing Systems*. ACM, 2017, pp. 4911–4923.

[27] A. Trory, K. Howland, and J. Good, "Designing for concreteness fading in primary computing," in *Proceedings of the 17th ACM Conference on Interaction Design and Children*. ACM, 2018, pp. 278–288.

[28] N. Cohn, *The Visual Language of Comics: Introduction to the Structure and Cognition of Sequential Images.* A&C Black, 2013.

[29] C. Bolton-Gary, "Connecting through comics: Expanding opportunities for teaching and learning." *Online Submission*, 2012.

[30] B. Zimmerman, "Creating comics fosters reading, writing and creativity," *Education Digest*, vol. 74, no. 4, pp. 55–57, 2008.

[31] J. C. Short, B. Randolph-Seng, and A. F. McKenny, "Graphic presentation: An empirical examination of the graphic novel approach to communicate business concepts," *Business Communication Quarterly*, vol. 76, no. 3, pp. 273–303, 2013.

[32] Educatorstechnology, "Teachers guide to the use of comic strips in class: Some helpful tips and resources," 2018, last accessed 17 January 2020. [Online]. Available: https://www.educatorstechnology.com/2018/01/teachers-guide-to-use-of-comic-strips.html

[33] M. Manno, "Comics in the classroom: Teaching content with comics," 2014, last accessed 17 January 2020. [Online]. Available: https://teach.com/blog/teaching-content-with-comics/

[34] R. with pictures, "Reading with pictures," 2020, last accessed 17 January 2020. [Online]. Available: https://www.educatorstechnology.com/2018/01/teachers-guide-to-use-of-comic-strips.html

[35] C. M. Tribull, "Sequential science: A guide to communication through comics," *Annals of the Entomological Society of America*, vol. 110, no. 5, pp. 457–466, 2017.

[36] L. F. Pelton and T. Pelton, "The learner as teacher: Using student authored comics to "teach" mathematics concepts," in *EdMedia+ Innovate Learning*. Association for the Advancement of Computing in Education (AACE), 2009, pp. 1591–1599.

[37] L. F. Pelton, T. Pelton, and K. Moore, "Learning by communicating concepts through comics," in *Society for Information Technology & Teacher Education International Conference*. Association for the Advancement of Computing in Education (AACE), 2007, pp. 1974–1981.

[38] "Pixton," last accessed 5 January 2020. [Online]. Available: https://www.pixton.com/

[39] J. Sellars, "Comics in the classroom," 2017, last accessed 5 December 2019. [Online]. Available: https://www.gse.harvard.edu/news/uk/17/12/comics-classroom

[40] V. Yulian, "Developing teaching materials using comic media to enhance students' mathematical communication," in *IOP Conference Series: Materials Science and Engineering*, vol. 335, no. 1. IOP Publishing, 2018, p. 012110.

[41] G. Yang, "Why comics belong in the classroom," 2016. [Online]. Available: https://www.youtube.com/watch?v=Oz4JqAJbxj0&feature=youtu.be
[42] B. Bach, N. H. Riche, S. Carpendale, and H. Pfister, "The emerging genre of data comics," *IEEE computer graphics and applications*, vol. 37, no. 3, pp. 6–13, 2017.
[43] Z. Zhao, R. Marr, and N. Elmqvist, "Data comics: Sequential art for data-driven storytelling," *tech. report*, 2015.
[44] M. Golembewski and M. Selby, "Ideation decks: a card-based design ideation tool," in *Proceedings of the 8th ACM Conference on Designing Interactive Systems*, 2010, pp. 89–92.
[45] IDEO, *IDEO Method Cards: 51 ways to inspire design*. William Stout, 2003.
[46] R. A. Fowles, "Design methods in uk schools of architecture," *Design Studies*, vol. 1, no. 1, pp. 15–16, 1979.
[47] E. Luger, L. Urquhart, T. Rodden, and M. Golembewski, "Playing the legal card: Using ideation cards to raise data protection issues within the design process," in *Proceedings of the 33rd Annual ACM conference on human factors in computing systems*, 2015, pp. 457–466.
[48] K. Compton, E. Melcer, and M. Mateas, "Generominos: Ideation cards for interactive generativity," in *Thirteenth Artificial Intelligence and Interactive Digital Entertainment Conference*, 2017.
[49] J. S. Bauer and J. A. Kientz, "Designlibs: A scenario-based design method for ideation," in *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, 2013, pp. 1955–1958.
[50] Y. Deng, A. N. Antle, and C. Neustaedter, "Tango cards: a card-based design tool for informing the design of tangible learning games," in *Proceedings of the 2014 conference on Designing interactive systems*, 2014, pp. 695–704.
[51] S. Mora, F. Gianni, and M. Divitini, "Tiles: a card-based ideation toolkit for the internet of things," in *Proceedings of the 2017 conference on designing interactive systems*, 2017, pp. 587–598.
[52] M. of Education, *The Ontario Curriculum Grades 10 to 12 Computer Studies*, 2008. [Online]. Available: http://www.edu.gov.on.ca/eng/curriculum/secondary/computer10to12_2008.pdf
[53] A. J. T. Force, "Computer science curricula 2013: Curriculum guidelines for undergraduate degree programs in computer science," Technical report, Association for Computing Machinery (ACM) IEEE Computer Society, Tech. Rep., 2013.
[54] S. Duncan, C. Burkholder, and M. Hennigin, "Vertical learning with classroom walls," 2019, last accessed 16 December 2019. [Online]. Available: http://www.communityplaythings.com/resources/articles/2019/vertical-learning-with-classroom-walls
[55] D. Chung and R.-H. Liang, "Understanding the usefulness of ideation tools with the grounding lenses," in *Proceedings of the Third International Symposium of Chinese CHI*, 2015, pp. 13–22.
[56] S. Suh, "Coding strip," last accessed 12 February 2020. [Online]. Available: https://codingstrip.github.io/