Developing Novice Programmers' Self-Regulation Skills with Code Replays

Benjamin Xie
benjixie@stanford.edu
@benjixie | he/they

Jared Ordona Lim*
jlim419@gatech.edu
@jaredordonalim | he/they

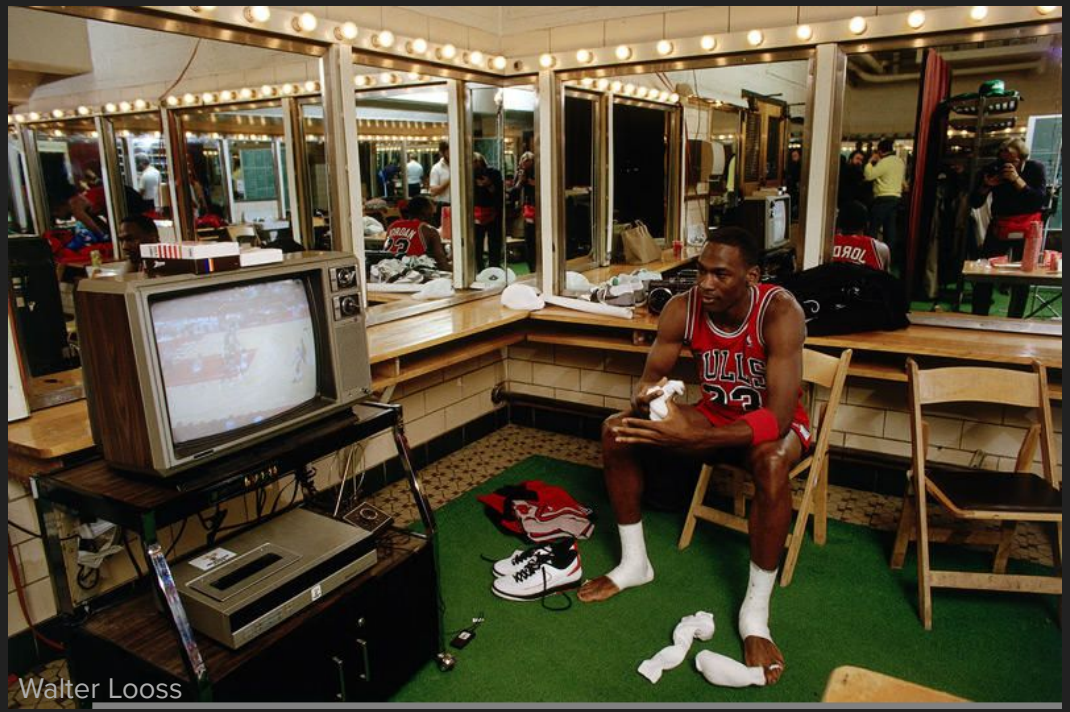
Paul K.D. Pham*
pkdpham@uw.edu
@pkdpham | he/him

Min Li
minli@uw.edu

Amy J. Ko
ajko@uw.edu
@amyjko | she/her

Information School UNIVERSITY of WASHINGTON
PAUL G. ALLEN SCHOOL OF COMPUTER SCIENCE & ENGINEERING
COLLEGE OF EDUCATION UNIVERSITY of WASHINGTON
Stanford University Human-Centered Artificial Intelligence
Stanford McCoy Family Center for Ethics in Society

Replays support learning in many contexts



How can replays support learning programming?

Supporting self-regulation of programming process

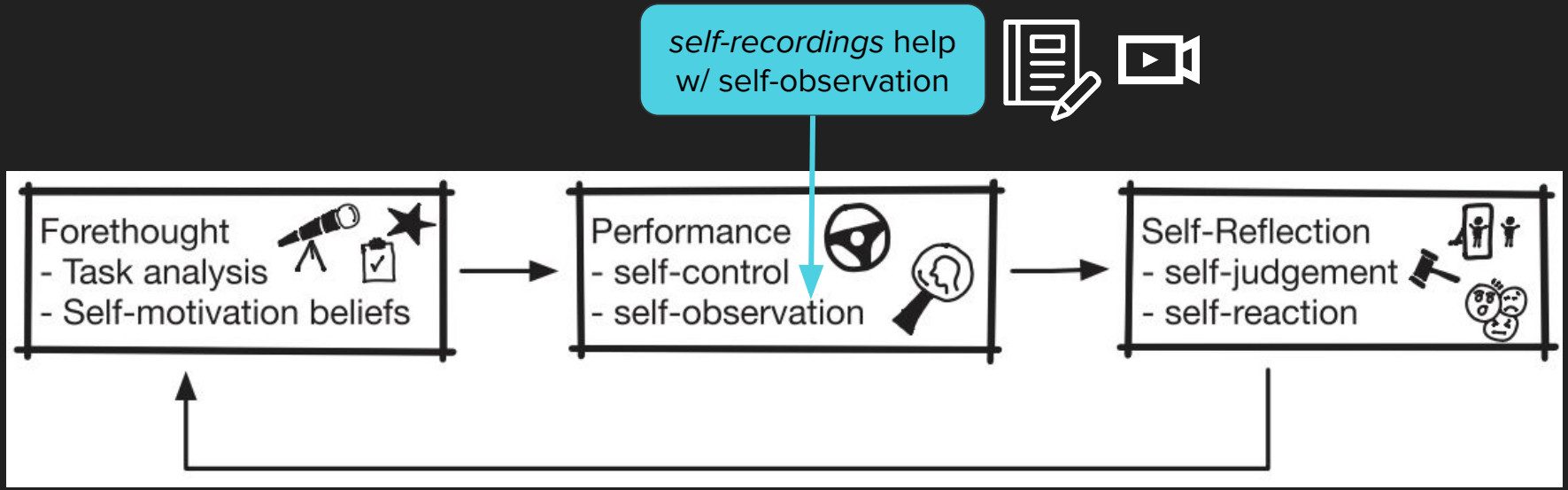


Fig. 5 from Loksa, Margulieux, Becker, Craig, Denny, Pettit, & Prather, J. (TOCE 2021)

Keystroke Logs (KSL)

- Timestamped logs of key presses, submissions used to recreate code production process
- Most prior work have teachers reviewing KSLs
- This study: Learners review KSLs to develop self-regulation skills

Novices reflecting on KSL self-recordings can help programming self-regulation

how? trade-offs? fit w/ pedagogy, tools? instructor use?

prompt

enough_candy()

Everyone loves candy, but you've only got so much to hand out for Halloween! Kids and teens come to your door asking for candy, so of course, you'll give it to them, but teenagers need more candy to get through high school.

Write a function called `enough_candy` that will return `True` if you had enough candy to last a night full of trick-or-treaters (kids and teens who come to your door asking for candy), or `False` otherwise.

Parameters:

- `total_candy` is an integer that represents the number of pieces of candy that you have
- `treaters` is a list of integers that contains only 0s and 1s that where each value represents someone who comes to your door; a `1` represents teens and a `0` represents kids.

This function has the following rules:

- Teens ask for 3 pieces of candy
- Kids ask for 1 piece of candy
- You have enough candy if you have 0 or more pieces of candy after helping everyone

EDIT

video player

```
def enough_candy(total_candy, treaters):
  for treater in treaters:
    if treater == 1:
        total_candy -= 3
    elif treater == 0:
        total_candy -= 1
  return total_candy
```

Code Run	Expected Output	Actual Output	Correct?
enough_candy(10, [1,1,0,1])	true	true	✓
enough_candy(3, [1,1])	false	false	✓
enough_candy(6, [1,0,0,0])	true	true	✓

edits pause

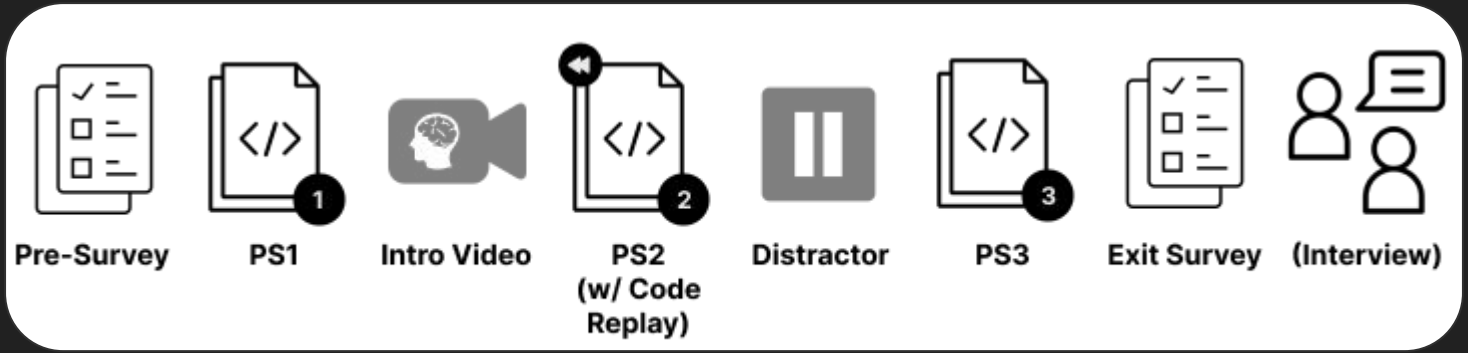
play prev/next edit

Evaluation Study w/ 21 Novice Programmers

Research Questions:

1. How do code replays affect self-regulation?
2. How do novice programmers use replays differently?

Study: 21 novices enrolled in CS1 classes using Code Replayer for 2 weeks



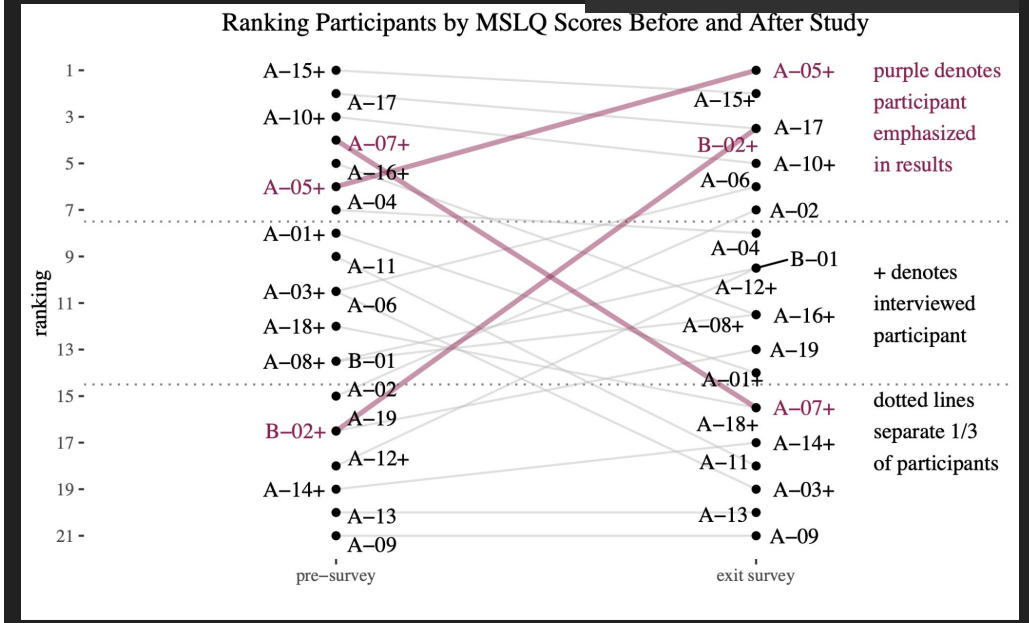
Read the paper!

Table 1: Descriptions and examples of programming and self-regulation behaviors we qualitatively coded for in 294 reflection prompts, split evenly between PS1 and PS3. Percentages reflect the proportion of reflection prompts (out of 147) that were coded for a given behavior for the problem set before (PS1) and after (PS3) participants watched their code replays.

Behaviors	Description	Example Response	PS1	PS3	Δ
PIN: Interpret Prompt	Interpreting the prompt, reconsidering actions in reference to the prompt, or decomposing the problem.	*I read the prompt and knew what it was asking of me, to subtract expenses from revenue...* A-09	52%	59%	+6%
PSA: Search for analogous problems	Demonstrating intent to reuse knowledge or code from related problems.	*I found that this problem is kind of similar to the previous reverse number question, so I go back and check some of my code there...* A-16	2.0%	3.4%	+1%
PSS: Search for solutions	Adapting solutions to related problems or by finding solutions in textbooks, online, or from classmates or teachers.	*I thought about how [my instructor] said in class that you can multiply integer n with strings...* A-18	10%	7%	-3%
PEP: Evaluation a potential solution	Demonstrating testing or evaluating outcomes intent to test a potential solution.	*I knew this should be a for loop that iterates between 1 and n.* A-11	4%	2%	-2%
PIM: Implement a solution	Translating a solution into code.	*I use str() to convert int to string, and then compare the result to target.* A-07	82%	65%	-17%
PEI: Evaluate implemented solution	Evaluating correctness and quality of an implementation.	*Once I tested my first run and realized I didn't get what I wanted, I looked at the actual output and decided where in my code I would need to make small adjustments...* A-08	20%	16%	-4%
SPL: Planning	Intended work goals or intended order of work.	*First I need to tell what's the original price of this car, then I tell its current price depending on my credit score. Finally I tell if I could buy*	33%	38%	+5%
		umbers, and f the 3 num- .. B-01	46%	47%	+1%
		tring by an ber times of	18%	22%	+4%
on Cognition	or feelings.	*s pretty ner-vous about running the code* A-05	14.3%	13.6%	-1%
SSE: Self-explanation/ rationale	Code explanation for increased understanding or to provide rationale to decisions or behaviors.	*I feel like a for loop is needed to get the different numbers to multiply and then also keep track because with factorials you build on the previous multiplication problem.* - A-10	44%	46%	+2%
(none)	no codes apply to this response	*Sorry, I chose to skip the question.* B-01	2.7%	7.4%	+5%

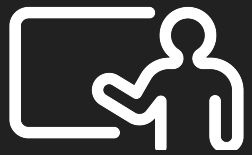
Qualitative coding of changes in self-regulation behaviors before and after using code replayers

Change in self-reported metacognitive skills before and after study



Interview Data: Deep Dive into 2 Participants

A: Younger adult learner
Learning CS for first time
Used replays for sports



B: Older learner
Former teacher
CS exp. from 10+ yrs ago

High level findings from interviews

Replays encouraged self-regulation through reflection and self-explanation

*"**I was like standing behind myself doing this code**... Whenever I'm talking about my ideas of what I'm learning, I feel like that helps me... And I feel like **if you're teaching or explaining what you're doing to other people, to myself, then I guess it shows I'm understanding.**"*

A: Younger adult learner
Learning CS for first time
Used replays for sports

B: Older learner
Former teacher
CS exp. from 10+ yrs ago

High level findings from interviews

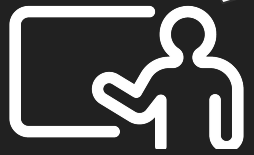
Replays encouraged self-regulation through reflection and self-explanation

Perspectives on replays of struggle were polarized

*"**Watching yourself can bring up bad memories**. I feel for somebody that is sensitive to... failure and stuff, it can be detrimental in their self-esteem and cause discouragement."*

A: Younger adult learner
Learning CS for first time
Used replays for sports

*"On the ones where they got harder, the most interesting for me **was watching where I changed course**, or where I made mistakes..."*



B: Older learner
Former teacher
CS exp. from 10+ yrs ago

High level findings from interviews

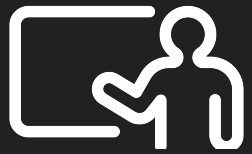
Replays encouraged self-regulation through reflection and self-explanation

Perspectives on replays of struggle were polarized

Scaffolding is necessary for replays to help students

*A: "**I could see [Code Replays] being a useful tool if there's some... guided process, or like something to help you understand**? ... as a teacher brain, I can be like 'that would be an amazing tool', because I could sit with a student, and I could be like, 'see, look what you were doing here'... It forces both the sort of meta-thinking music without thinking."*

A: Younger adult learner
Learning CS for first time
Used replays for sports



B: Older learner
Former teacher
CS exp. from 10+ yrs ago

Interpreting findings as opportunities

High level findings:

- Replays encouraged self-regulation through reflection and self-explanation
- Perspectives on replays of struggle were polarized
- Scaffolding is necessary for replays to help students

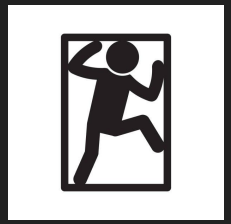
Opportunities for future research & practice

- Pedagogical Applications of Code Replays
- Alterations to interface design

Pedagogical Opportunities for Students, Teachers, and TAs

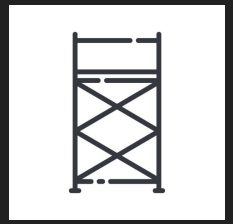
Practice

Replays can help students reflect and identify opportunities for improvement.



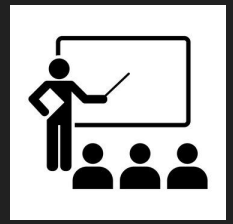
Getting unstuck

Replays could allow teachers and TAs to identify learners' mistakes.



Scaffolding

Teachers providing early guidance through solution replays.



Emphasis

Teachers can use replays to highlight important skills or common mistakes.

Design Opportunities: Alternatives to Interface Design

Emphasis on **inflection points** (e.g. changes in strategies, long gaps followed by bursts of activity).

More informative recordings of **pauses** (e.g. think-aloud, require comments)

Minimizing frustration during viewing (e.g. selective and pedagogically supportive uses of Code Replays as previously mentioned.)

Code Replays can serve as a complementary tool to augment and improve existing self-regulation practices.

Developing Novice Programmers' Self-Regulation Skills with Code Replays

Benjamin Xie
benjixie@stanford.edu
@benjixie | he/they

Jared Ordona Lim*
jlim419@gatech.edu
he/they

Paul K.D. Pham*
pkdpham@uw.edu
he/him

Min Li
minli@uw.edu

Amy J. Ko
ajko@uw.edu
@amyjko | she/her

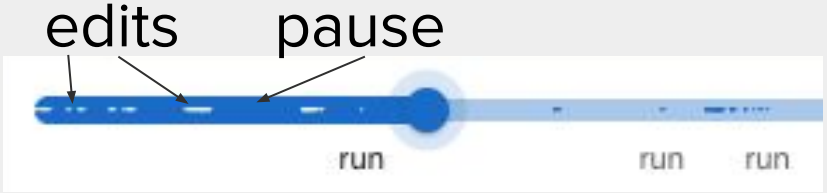
High level findings:

1. Replays encouraged self-regulation through reflection and self-explanation
2. Perspectives on replays of struggle were polarized
3. Scaffolding is necessary for replays to help students

Pedagogical opportunities: practice, help-seeking, scaffolding, emphasis

Design opportunities: more informative replays of pauses, selective use, representation

benji.phd/icer23



doi: 10.1145/3568813.3600127

enough_candy()

Everyone loves candy, but you've only got so much to hand out for Halloween! Kids and teens come to your door asking for candy, so of course, you'll give it to them, but teenagers need more candy to get through high school.

Write a function called `enough_candy` that will return `True` if you had enough candy to last a night full of trick-or-treaters (kids and teens who come to your door asking for candy), or `False` otherwise.

Parameters:

- `total_candy` is an integer that represents the number of pieces of candy that you have
- `treaters` is a list of integers that contains only 0s and 1s that where each value represents someone who comes to your door; a `1` represents teens and a `0` represents kids.

This function has the following rules:

- Teens ask for 3 pieces of candy
- Kids ask for 1 piece of candy
- You have enough candy if you have 0 or more pieces of candy after helping everyone

EDIT

```
1 def enough_candy(total_candy, treaters):
2   for treater in treaters:
3       if treater == 1:
4           total_candy -= 3
5   return total_candy
```

Code Run	Expected Output	Actual Output	Correct?
enough_candy(10, [1,1,0,1])	true	1	✓
enough_candy(3, [1,1])	false	-3	⚠
enough_candy(6, [1,0,0,0])	true	3	⚠

RESET RUN CODE

Done executing. See results on right.

- `treaters` is a list of integers that contains only 0s and 1s that where each value represents someone who comes to your door; a `1` represents teens and a `0` represents kids.

This function has the following rules:

- Teens ask for 3 pieces of candy
- Kids ask for 1 piece of candy
- You have enough candy if you have 0 or more pieces of candy after helping everyone

EDIT REPLAY

```
1 def enough_candy(total_candy, treaters):
2   for treater in treaters:
3     if treater == 1:
4         total_candy -= 3
5     elif treater == 0:
6         total_candy -= 1
7   return total_candy
```

Code Run	Expected Output	Actual Output	Correct?
enough_candy(10, [1,1,0,1])	true	true	✓
enough_candy(3, [1,1])	false	false	✓
enough_candy(6, [1,0,0,0])	true	true	✓

Done executing. Please complete reflection below

What did you notice while watching your replay?

Response to reflection prompt

I noticed a very long pause when working inside of the for-loop. I also saw that I didn't immediately think to use multiple conditions to check for the treater, which I didn't get until after my first run.

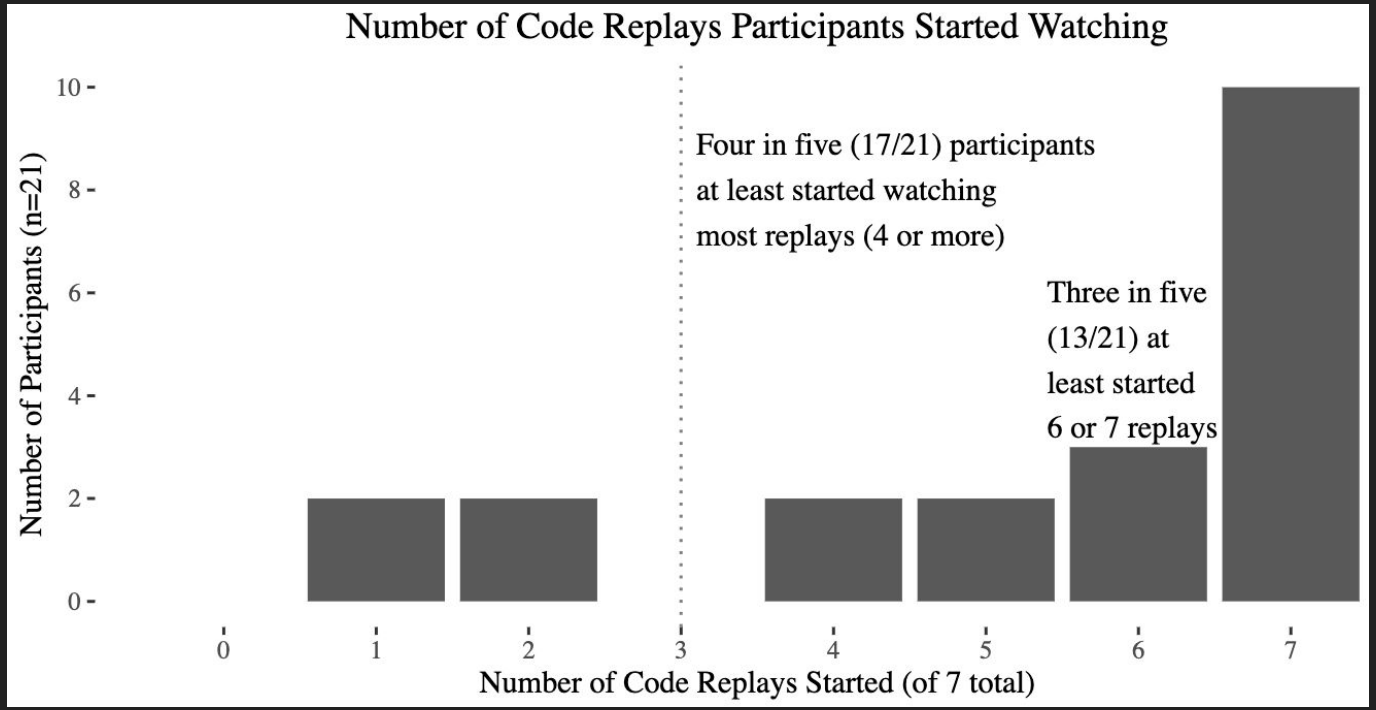
What steps would you take on future problems?

Response to reflection prompt

I need to go a lot slower and think before I code. I cannot assume I know exactly how to code something, and I should probably plan more.

SUBMIT REFLECTIONS

Most participants watched (almost) all replays



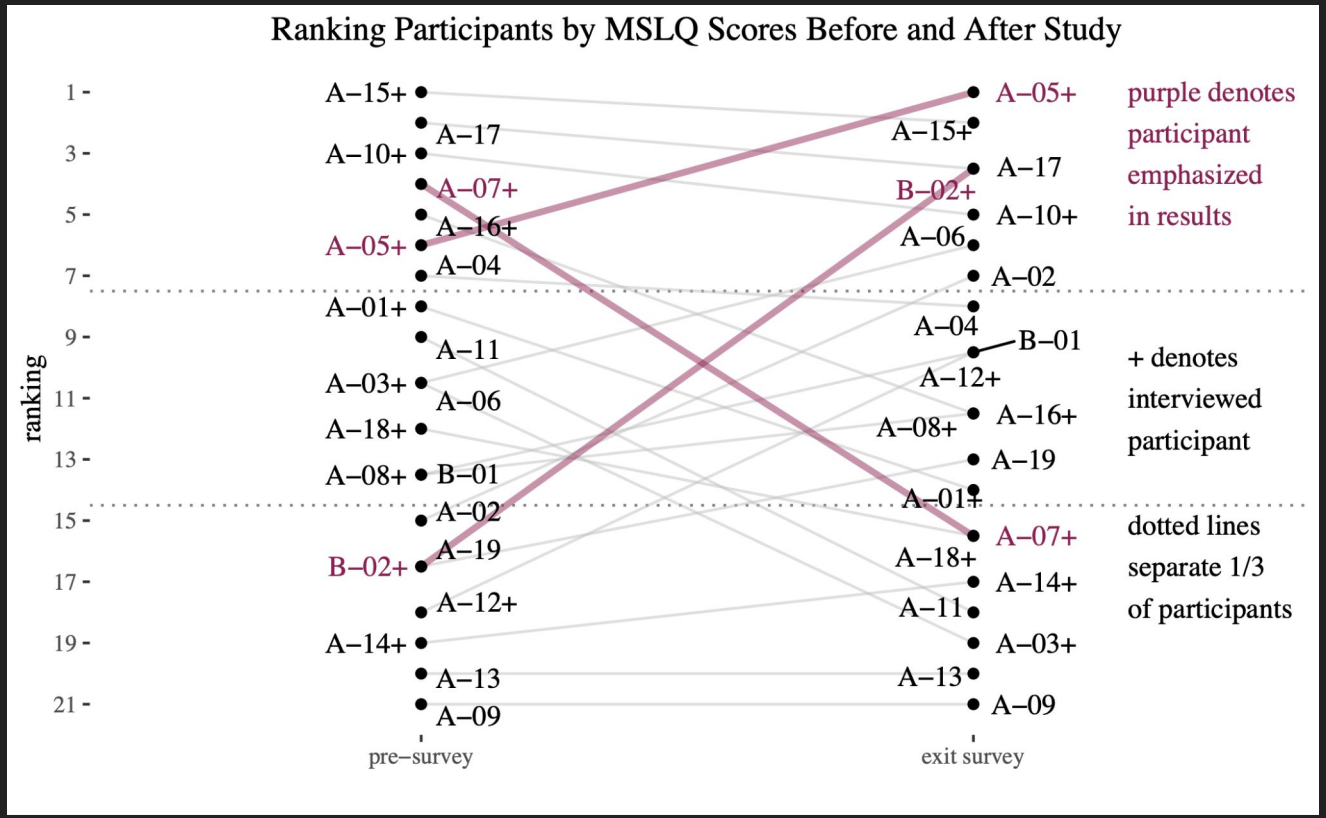

Ranking Participants by MSLQ Scores Before and After Study
ranking
1
3
5
7
9
11
13
15
17
19
21
pre−survey
exit survey
A−15+
A−17
A−10+
A−07+
A−16+
A−05+
A−04
A−01+
A−11
A−03+
A−06
A−18+
A−08+
B−01
A−02
B−02+
A−19
A−12+
A−14+
A−13
A−09
A−05+
A−15+
A−17
B−02+
A−10+
A−06
A−02
A−04
B−01
A−12+
A−08+
A−16+
A−19
A−01+
A−07+
A−18+
A−14+
A−11
A−03+
A−13
A−09
purple denotes participant emphasized in results
+ denotes interviewed participant
dotted lines separate 1/3 of participants

Table 1: Descriptions and examples of programming and self-regulation behaviors we qualitatively coded for in 294 reflection prompts, split evenly between PS1 and PS3. Percentages reflect the proportion of reflection prompts (out of 147) that were coded for a given behavior for the problem set before (PS1) and after (PS3) participants watched their code replays.

Behaviors	Description	Example Response	PS1	PS3	Δ
PIN: Interpret Prompt	Interpreting the prompt, reconsidering actions in reference to the prompt, or decomposing the problem.	*I read the prompt and knew what it was asking of me, to subtract expenses from revenue...* A-09	52%	59%	+6%
PSA: Search for analogous problems	Demonstrating intent to reuse knowledge or code from related problems.	*I found that this problem is kind of similar to the previous reverse number question, so I go back and check some of my code there...* A-16	2.0%	3.4%	+1%
PSS: Search for solutions	Adapting solutions to related problems or by finding solutions in textbooks, online, or from classmates or teachers.	*I thought about how [my instructor] said in class that you can multiply integer n with strings...* A-18	10%	7%	-3%
PEP: Evaluation a potential solution	Demonstrating testing or evaluating outcomes intent to test a potential solution.	*I knew this should be a for loop that iterates between 1 and n.* A-11	4%	2%	-2%
PIM: Implement a solution	Translating a solution into code.	*I use str() to convert int to string, and then compare the result to target.* A-07	82%	65%	-17%
PEI: Evaluate implemented solution	Evaluating correctness and quality of an implementation.	*Once I tested my first run and realized I didn't get what I wanted, I looked at the actual output and decided where in my code I would need to make small adjustments...* A-08	20%	16%	-4%
SPL: Planning	Intended work goals or intended order of work.	*First I need to tell what's the original price of this car, then I tell its current price depending on my credit score. Finally I tell if I could buy this car...* A-15	33%	38%	+5%
SPM: Process Monitoring	Work being started, work currently in progress, when a task is complete, or that identify actions as part of their process.	*I first split the value of n into 3 numbers, and then calculated whether the sum of the 3 numbers is equal to the value of target..* B-01	46%	47%	+1%
SCM: Comprehension Monitoring	Identifying known or unknown programming concepts or understanding of the problem prompt. Reflection about the understanding of code or problem prompts.	*I know that you can multiply a string by an integer and it will give me X number times of the string...* A-12	18%	22%	+4%
SRE: Reflection on Cognition	Reflecting on prior thoughts, behaviors, or feelings.	*With incredible frustration... I was pretty nervous about running the code* A-05	14.3%	13.6%	-1%
SSE: Self-explanation/ rationale	Code explanation for increased understanding or to provide rationale to decisions or behaviors.	*I feel like a for loop is needed to get the different numbers to multiply and then also keep track because with factorials you build on the previous multiplication problem.* - A-10	44%	46%	+2%
(none)	no codes apply to this response	*Sorry, I chose to skip the question.* B-01	2.7%	7.4%	+5%

Let’s try out an activity together!

In groups of three, assign each person one of the following roles:

A. ***The Presenter:*** take ~45 seconds to fold a paper airplane while explaining the steps they’re taking out loud.
B. ***The Observer:*** watch and listen to Person A fold their plane and take **mental** notes about their method.
C. ***The Notetaker:*** watch and listen to Person A give their pitch and take **written** notes (on paper or phone/computer) about their method.

Let’s replay!

- ***The Presenter*** and ***Observer*** reflect together - ***The Observer*** provides a recap of the steps the Presenter took when making the paper airplane.

- After 1-2 minutes, ***The Notetaker*** joins and provides a recap of the steps the Presenter took when making the paper airplane based on the notes they took.

Let’s reflect!

Observer: What was it like giving a recap from mental notes?

Notetaker: What was it like giving a recap from physical notes?

Presenter: Whose feedback provided a clearer summary of your airplane-folding technique?

Replays encouraged self-regulation through reflection & self-explanation

B: “I was like standing behind myself doing this code… Whenever I’m talking about my ideas of what I’m learning, I feel like that helps me… And I feel like if you’re teaching or explaining what you’re doing to other people, to myself, then I guess it shows I’m understanding.”

B: “Sometimes, you know, I record myself thinking, ‘what does my shot look like?’… That kind of reminds me [of] breaking down areas where I can improve. When I’m watching basketball, I ask myself, is there a way I can make this more efficient? … Kind of like Code Replay, I’m doing the same thing in a different way.”

Polarized perspectives on replays of struggle

B: “Watching yourself can bring up bad memories. I feel for somebody that is sensitive to… failure and stuff, it can be detrimental in their self-esteem and cause discouragement.”

A: “On the ones where they got harder, the most interesting for me was watching where I changed course, or where I made mistakes… But more often, just sort of like ‘oh I started to go at it this way and in the process of doing that, I realized a more elegant way of doing that.’”

Replays require some scaffolding

A: “I could see [Code Replays] being a useful tool if there’s some… guided process, or like something to help you understand? … as a teacher brain, I can be like ‘that would be an amazing tool’, because I could sit with a student, and I could be like, ‘see, look what you were doing here’… It forces both the sort of meta-thinking music without thinking.”

High level findings from interviews

- Replays encouraged self-regulation through reflection and self-explanation
- Perspectives on replays of struggle were polarized
- Scaffolding is necessary for replays to help students

Major Tensions with Code Replays

1. Benefits and Harm to Students
2. Lack of Visibility in Pauses
3. Design Issues and Branching Paths

Opportunities + Applications

- Pedagogical Applications of Code Replays
- Alterations to Interface Design

Opportunity 1: Pedagogical Applications of Code Replays

Who reviews Code Replays?

1-on-1 tutoring: ~~both individually and with an instructor.~~

Teaching demonstration in a lecture: instructors finding replays to highlight critical mistakes or skills.

Getting unstuck: instructors and students review replays together to identify areas of improvement.

Scaffolding individual practice with meaningful, guided reflection questions.

Opportunity 1: Pedagogical Applications of Code Replays

When should we use Code Replays?

- Introductory Programming courses where teachers and students can reflect on replays together
- Individual practice, so long as students have guided reflection questions about their solution approach

Opportunity 2: Interface Design

How can Code Replays support self-regulation skills?

Expand on inflection points (e.g. changes in solution, long gaps followed by bursts of activity).

Designing for pauses and capturing what is currently unrecorded information.

Minimizing frustration – replays are great reflective tools, but also make you relive past mistakes.

Design Opportunities: Alternatives to Interface Design

- Emphasis on **inflection points** (e.g. changes in strategies, long gaps followed by bursts of activity).
- More informative recordings of **pauses** (e.g. think-aloud, comments)
- **Minimizing frustration** (e.g. selective and supportive uses of Code Replays)