

# Typing Exercises as Interactive Worked Examples for Deliberate Practice in CS Courses

Adam M. Gaweda, Collin F. Lynch, Nathan Seamon, Gabriel Silva de Oliveira, Alay Deliwa  
North Carolina State University



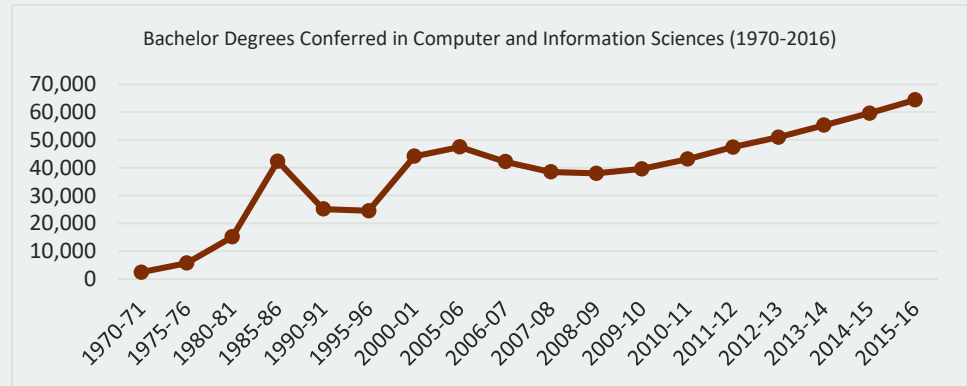
Creative Commons Attribution 4.0 International

# CS Popularity and Attrition

Computer Science enrollment is currently at an **all-time high** in the US (Bureau of Labor Statistics, 2017)

However, early CS courses commonly experience a **30-50% drop/fail** attrition rate

Novice CS students struggle with **syntax errors**, rather than problem-solving (Altadmri, Brown, 2015)



# Worked Examples and Templates

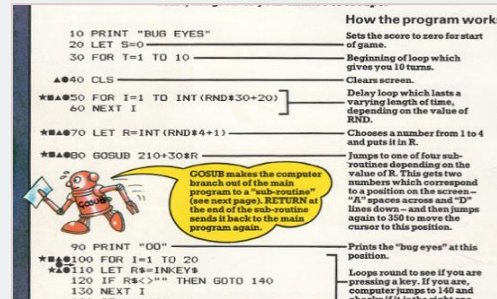
Computer Scientists store **recurring basic plan** mental models to apply a learned concept to a similar problem (Soloway, Ehrlich, 1984)

Debugging can take the majority of novices' time but can be mitigated with **canned code** (Brooks, 1977)

Multiple **worked examples** improve students' abstract mental models (Zhi et al, 2018)

# Novel CS Exercises

The emergence of novel CS exercises has created exercises like Parson's Puzzles, Output Prediction, and Bug Localization. Early computer magazines provided worked examples needed to be retyped for making computer games (Usbourne, 1982; Antic, 1982-90)



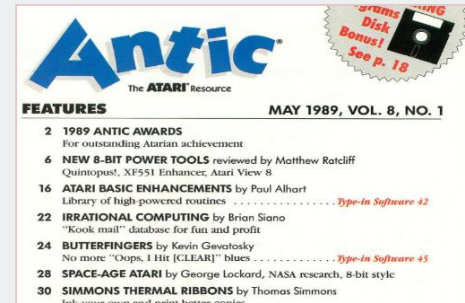
**How the program works**

```
10 PRINT "BUG EYES"
20 LET S=0
30 FOR T=1 TO 10
  ▲40 CLS
  *▲50 FOR I=1 TO INT (RND*30+20)
    60 NEXT I
  *▲70 LET R=INT (RND*4+1)
  *▲80 GOSUB 210+30R
  90 PRINT "OO"
  *▲100 FOR I=1 TO 20
    *▲110 LET R$=INKEY$
    120 IF R$<>" " THEN GOTO 140
    130 NEXT I
```

— Sets the score to zero for start of game.  
— Beginning of loop which gives you 10 turns.  
— Clears screen.  
— Delay loop which lasts a varying length of time, depending on the value of RND.  
— Chooses a number from 1 to 4 and puts it in R.  
— Jumps to one of four sub-routines depending on the value of R. This gets two numbers which correspond to a position on the screen— "R" spaces across and "D" lines down—and then jumps again to 350 to move the cursor to this position.  
— Prints the "bug eyes" at this position.  
— Loops round to see if you are pressing a key. If you are, computer jumps to 140 and checks if it is the right one.

**GOSUB makes the computer branch out of the main program to a "sub-routine" (see next page). RETURN at the end of the sub-routine sends it back to the main program again.**

Computer Space Games, 1982



**Antic**  
The ATARI Resource

grams Disk Bonus! See p. 18

**FEATURES** MAY 1989, VOL. 8, NO. 1

- 2 1989 ANTIC AWARDS  
For outstanding Atarian achievement
- 6 NEW 8-BIT POWER TOOLS reviewed by Matthew Ratcliff  
Quintapod, XFS1 Enhancer, Atari View 8
- 16 ATARI BASIC ENHANCEMENTS by Paul Althart  
Library of high-powered routines . . . . . *Type-In Software #2*
- 22 IRRATIONAL COMPUTING by Brian Siano  
"Kook mail" database for fun and profit
- 24 BUTTERFINGERS by Kevin Gevetosky  
No more "Oops, I Hit [CLEAR]" blues . . . . . *Type-In Software #5*
- 28 SPACE-AGE ATARI by George Lockard, NASA research, 8-bit style
- 30 SIMMONS THERMAL RIBBONS by Thomas Simmons  
Take your own and enjoy better copies.

Antic, 1985

# Typing Exercises

CS novices can benefit from retyping example code to build their own recurring basic plans without the risk of simply copying and pasting the code

Typing exercises give students canned code, which can be used as a template for future exercises

Retype the Code on the Left Here

```
1 public class NullDereference {
2     // Checks to see if name was a null value
3     public static void handlesNull(String name) {
4         // You can remove the if-else {}'s if they only
5         // execute 1 line of code.
6         if (name != null && name.length() > 0)
7             System.out.println("Hello " + name);
8         else
9             System.out.println("Name was a null value");
10    }
11
12    // This version only checks instantiated Strings
13    public static void crashesOnNull(String name) {
14        // Attempts to access the .length() method, that
15        // may not exist
16        if (name.length() > 0) {
17            System.out.println("Hello " + name)
18        } else {
19            System.out.println("Name was a null value");
20        }
21    }
22
23    public static void main(String[] args){
24        handlesNull("James Gosling"); // Runs Correctly
25        handlesNull(null); // Catches the null parameter
26        crashesOnNull("James Gosling"); // Runs Correctly
27        crashesOnNull(null); // null has no .length()
28        System.out.println("This line never runs due to the NPE");
29    }
30 }
```

# TYPOS

TYPOS is a low-level CS Exercise Platform

Students could **complete exercises** as often as they wanted and **download copies** after completion

Information and Images Available at [go.ncsu.edu/typos](https://go.ncsu.edu/typos)

The screenshot displays the TYPOS course interface. At the top, it says "Demonstration Course" and "CSC 111 - 001 (May '18 - Sep '18)". Below that, the word "Modules" is centered. The interface shows a list of modules, each in a box with a title and a list of exercises. The exercises are marked with star ratings and keyboard icons. The visible modules and exercises are:

- Getting Started** (minus icon):
  - ☆☆☆ Welcome to TYPOS! ⌨
  - ☆☆☆ Print Statements and Variables ⌨
- Variables and Data Types** (minus icon):
  - ☆☆☆ Parsing User Inputs ⌨
- Conditional Statements** (minus icon):
  - ☆☆☆ Booleans and Conditional Statements ⌨
  - ★★★ Comparison Misconceptions ⌨
- Looping Structures** (plus icon)

# Research Questions

**RQ1** - Do students refer to typing exercises while working on other tasks?

**RQ2** - How do different patterns of use correlate with student performance?

**RQ3** - Is there a correlation between completion of typing exercises and build failures on assignments?

# Study

TYPOS was deployed into **three offerings** of a second semester CS course which taught Java

Homework and Projects were graded through unit and code coverage tests on the Jenkins Continuous Integration system

Comments were graded by TAs

**3 - 4 optional typing exercises** were released each lecture mirroring the day's topic

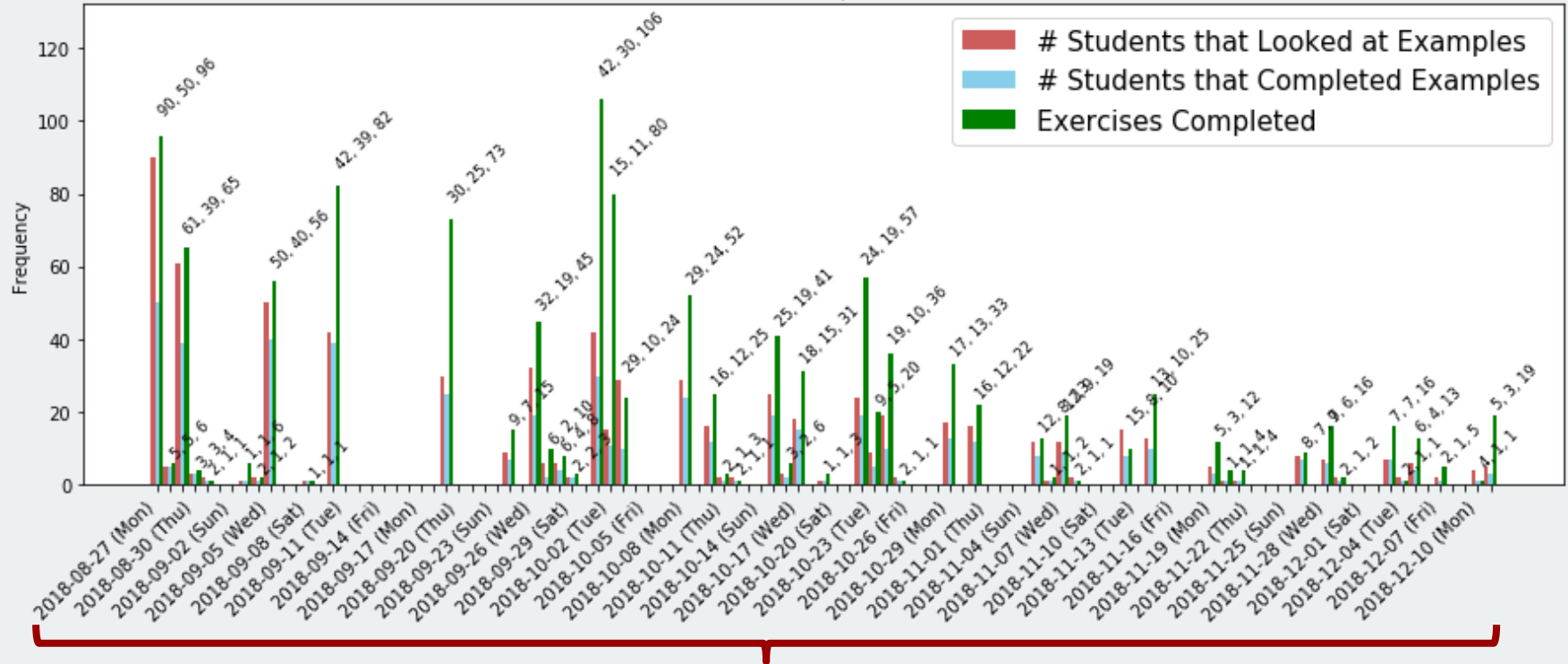
Total Number of Exercises Released per Offering: 66

In total, 337 students made 7,334 views, 10,097 attempts, and 3,024 completions



# Usage Analysis

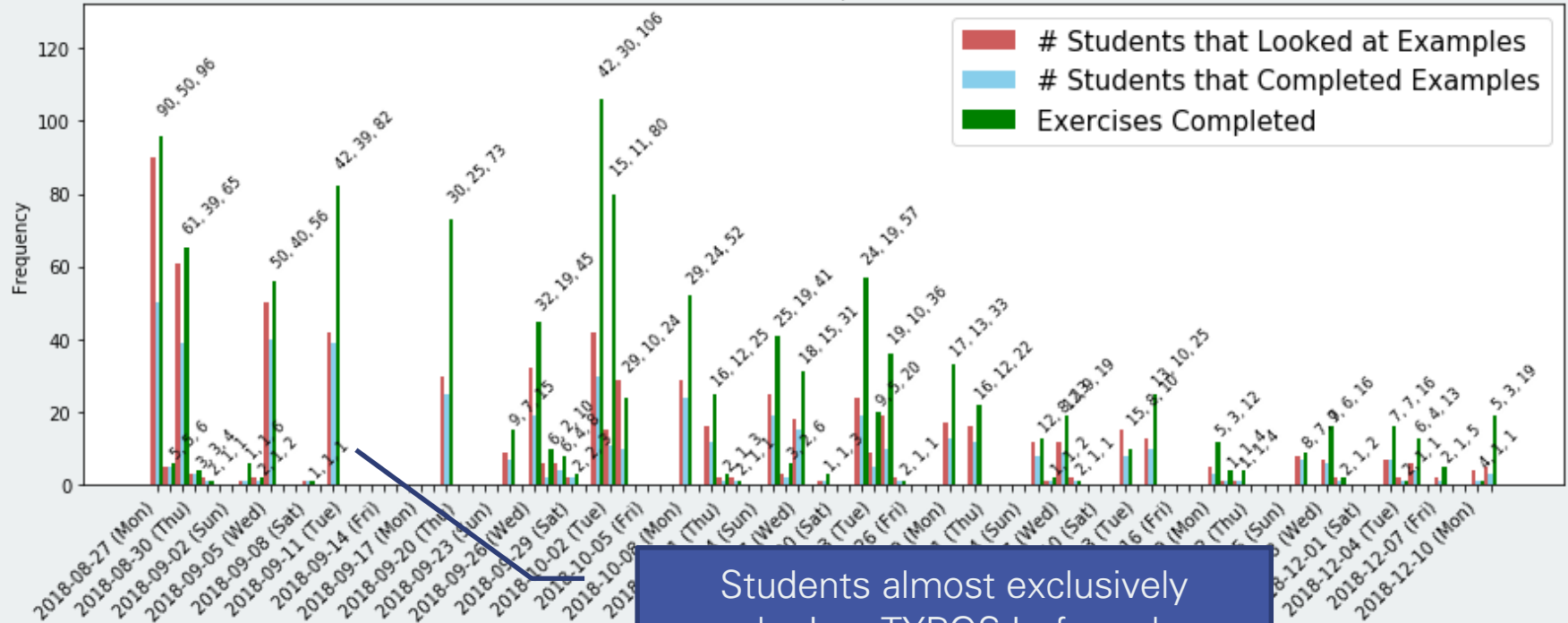
Student Activity for Fall 2018



Daily Activity for the Semester

# Usage Analysis

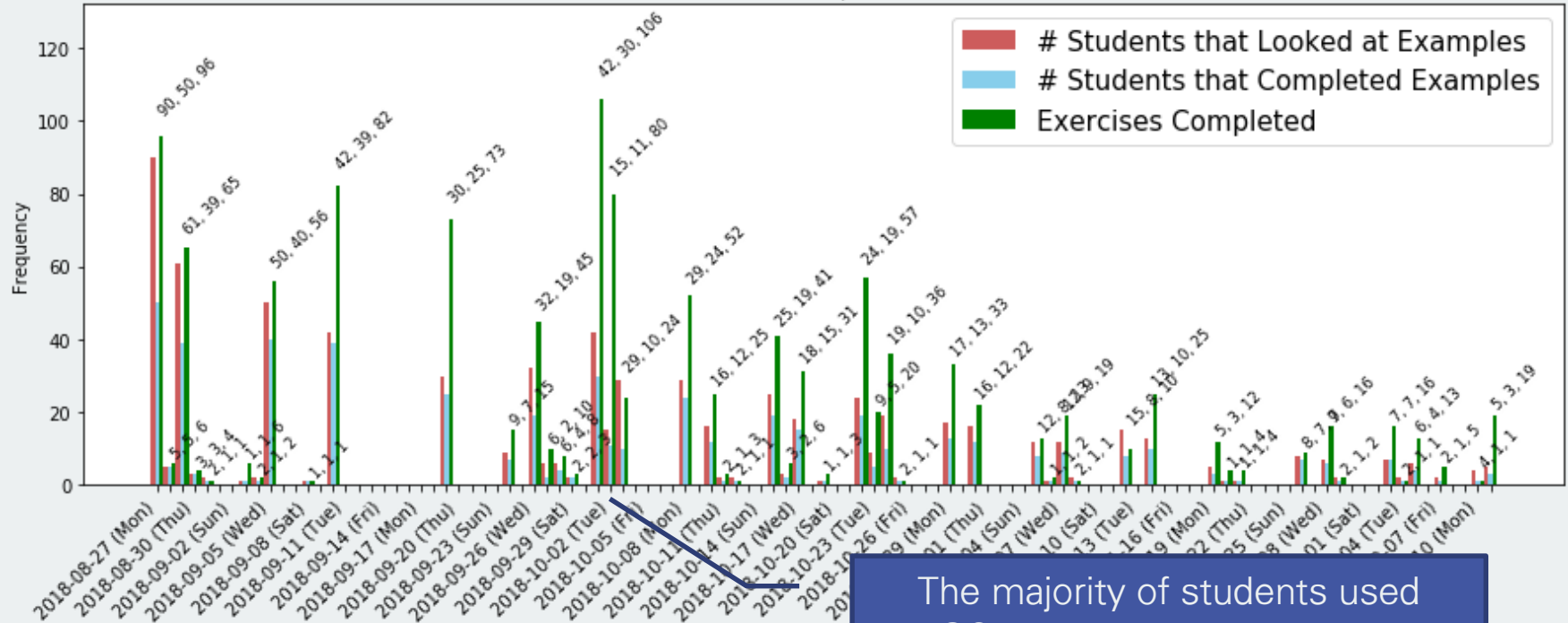
Student Activity for Fall 2018



Students almost exclusively worked on TYPOS before class lecture

# Usage Analysis

Student Activity for Fall 2018



The majority of students used TYPOS at the beginning but activity shifted to prior to exams

# Usage Analysis

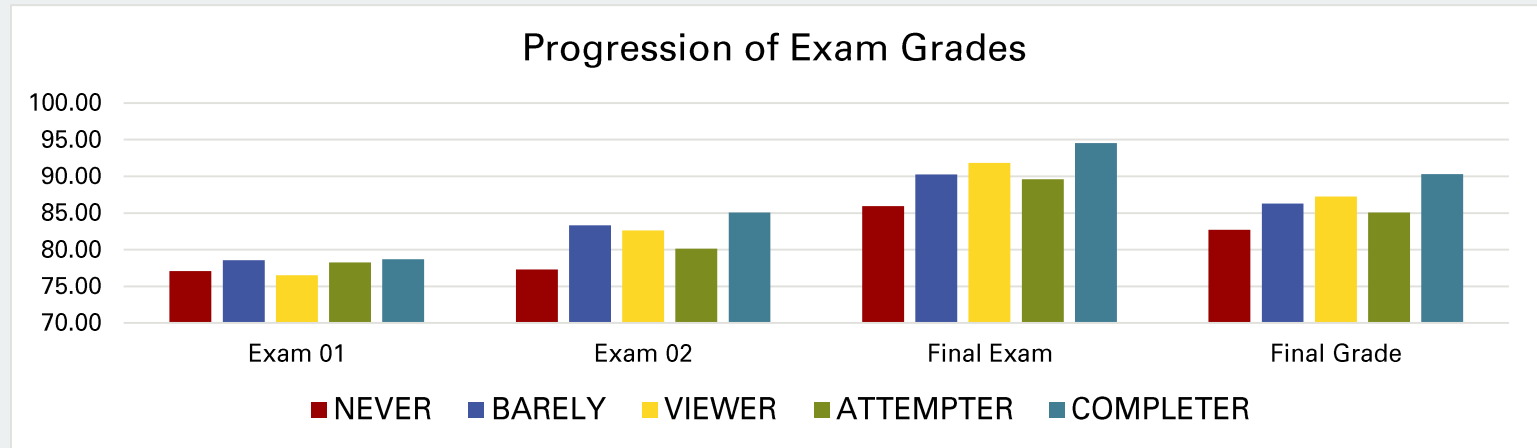
Active Users (77) comprised 75% of all activity on TYPOS

Category	Total	A	B	C	D	F	Avg Views	Avg Attempts	Avg Completes
Never	25	11	8	2	-	4	0	0	0
Barely	235	109	85	21	3	17	7.5	12.5	3.1
Viewers	17	9	4	3	-	1	58.1	13.7	3.4
Attempters	26	11	11	2	-	2	60.2	56.2	18.0
Completers	34	19	14	1	-	-	88.4	160.8	52.1

Active Users were classified as **viewers**, **attempters**, or **completers**

# Student Performance Analysis

Regular completers earned higher exam grades and final course grades when compared to other regular users



Student exam and course grades by category

# Student Performance Analysis

Initially low performing completers benefited from the additional low-level practice

Category	Total	Course Grades	Exam Learning Gains
Active Completers	34	0.065	0.161
Initially Low-Performing Students	16	0.038*	0.041*

P-values for all and lower-performing completers across course grade and learning gains (\*  $p < 0.05$ )

# Assignment Implementation Analysis

This course used the **Jenkins** Continuous Integration system to assess student submissions with three feedbacks



Red Ball

Build could not compile



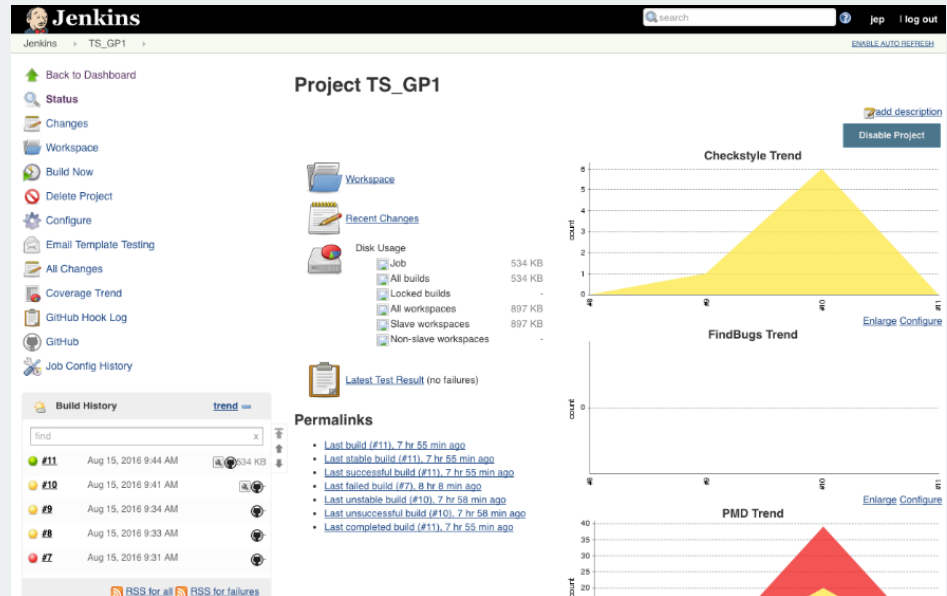
Yellow Ball

Build compiled, but failed tests



Green Ball

Build successfully completed all tests



# Assignment Implementation Analysis

We assessed all active users TYPOS behavior with Jenkins' red ball status

All TYPOS usage was **negatively correlated** with build failures

Category	# Viewed	# Attempted	# Completed
Active Students	-0.045**	-0.048**	-0.110**
Initially Low-Performing Students	-0.132*	-0.050**	-0.144**

The more exercises completed, the lower number of build failures

Pearson's r Comparing TYPOS Activity to Build Failures (\*  $p < 0.05$ , \*\*  $p < 0.01$ )



# Conclusions

**RQ1** - How do different patterns of use correlate with student performance?

**Regular completers earned higher course grades, especially in initially low-performing students**

**RQ2** - Do students refer to typing exercises while working on other tasks?

**Students reviewed TYPOS prior to exams and completed exercises prior to lecture**

**RQ3** - Is there a correlation between completion of typing exercises and build failures in other work?

**Reviewing and retyping worked examples reduced the number of build failures on Jenkins**

# Future Work



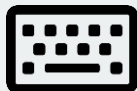
Lecture Video  
Introducing Topic



Topic Mastery

In a perfect world, everyone would master topics immediately.

# Future Work



Typing Exercise



Find the Bug



Parson's Puzzles



Lecture Video  
Introducing Topic



Self Explanation



Output Prediction



Fill in the Blank



Topic Mastery



Fix the Bug



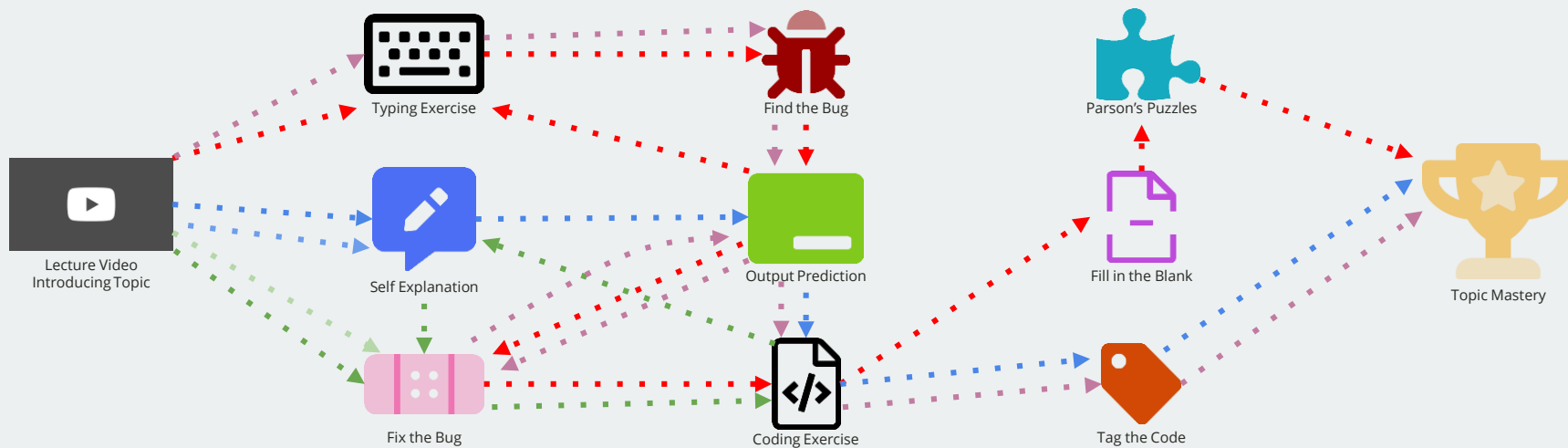
Coding Exercise



Tag the Code

However, in reality, there are multiple ways to practice

# Future Work



By mapping prior student **activity sequences**, we seek to provide students with their **next best step** to ensure they are mastering topics efficiently.