UCR

A Many Small Programs (MSP) Approach in a CS1 Course

Joe Michael Allen

Advisor: Frank Vahid
Dept. of Computer Science and Engineering
University of California, Riverside

UNIVERSITY OF CALIFORNIA, RIVERSIDE

Problem

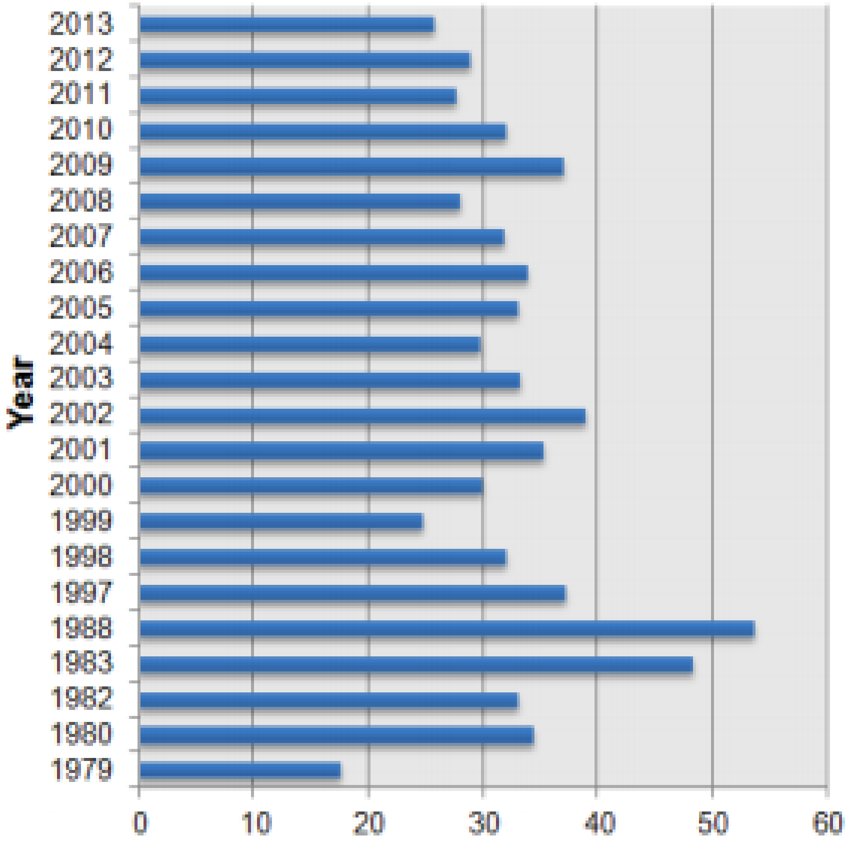
- CS1 issues
  - High student stress
  - Student dissatisfaction
  - High drop rates
  - Low retention
  - Low grades
  - Academic dishonesty

Our Solution

- Improve students' experience
  - Improve student satisfaction & happiness
  - Without worsening performance

- Focus on weekly programming assignments
  - Large part of the students' experience
  - Key source of issues – student struggle/fear

~ 30% non-passing rate over a 30 year period



~28%

~55%

~18%

Watson, C. and Li, F. "Failure Rates in Introductory Programming Revisited, " iTiCSE, 2014
http://dro.dur.ac.uk/19223/1/19223.pdf%3FDDD10%2Bd74ks0%2Bdcs0lw

Outline

- **Background & Related work**
- Our experience with an MSP approach
- MSP approach across universities & programming languages
- MSP approach tools & analysis
- Contributions

Background & Related work

- Improving CS1
  - Pair programming / Peer instruction
  - Programming language variety
  - Various tutoring models
  - Student self agency in assignments
  - Plagiarism detection

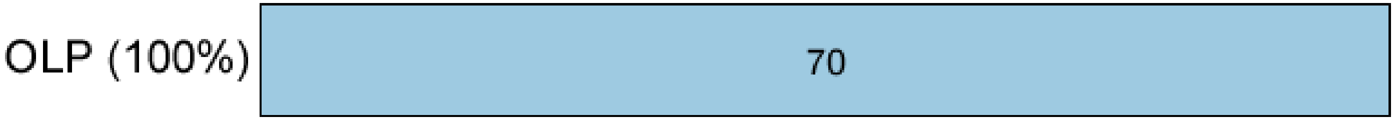
- Focus on programming assignments
  - Programming applications
  - Game design and Gamification

Outline

- Background & Related work
- **Our experience with an MSP approach**
- MSP approach across universities & programming languages
- MSP approach tools & analysis
- Contributions

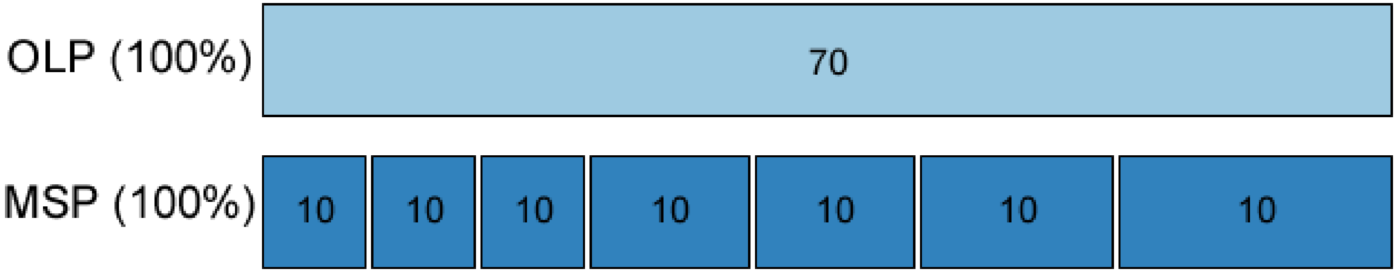
MSP teaching approach introduction

- Traditional: One Large Program (OLP) Approach
  - One larger programming assignment a week
  - Solution 50-200 lines
  - Long spec
  - Multiple topics



MSP teaching approach introduction

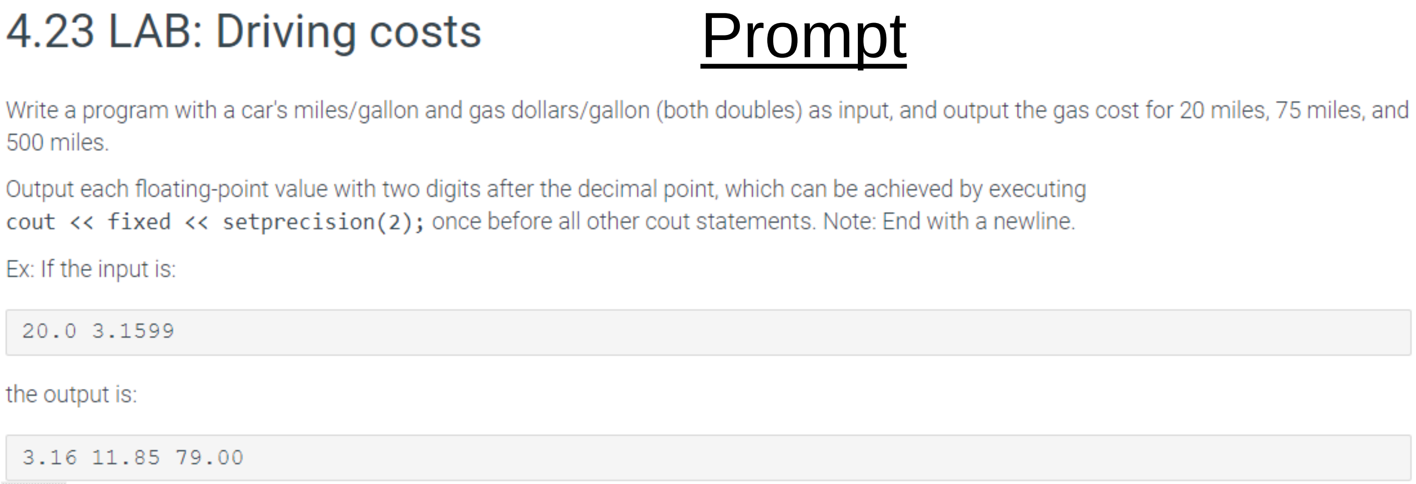
- Many Small Programs (MSP) approach: 5-7 small lab activities
  - Solution 10-50 lines each
  - Short & concise prompt



- Benefits
  - Less intimidating
  - Pivot if stuck
  - Build confidence, more practice
  - Partial credit

- Enabled by modern auto-graders
  - Easy to create / Instant feedback
  - zyLabs (zyBooks): ~30 min create lab

MSP lab activity - sample 1



Prompt

4.23 LAB: Driving costs

Write a program with a car's miles/gallon and gas dollars/gallon (both doubles) as input, and output the gas cost for 20 miles, 75 miles, and 500 miles.

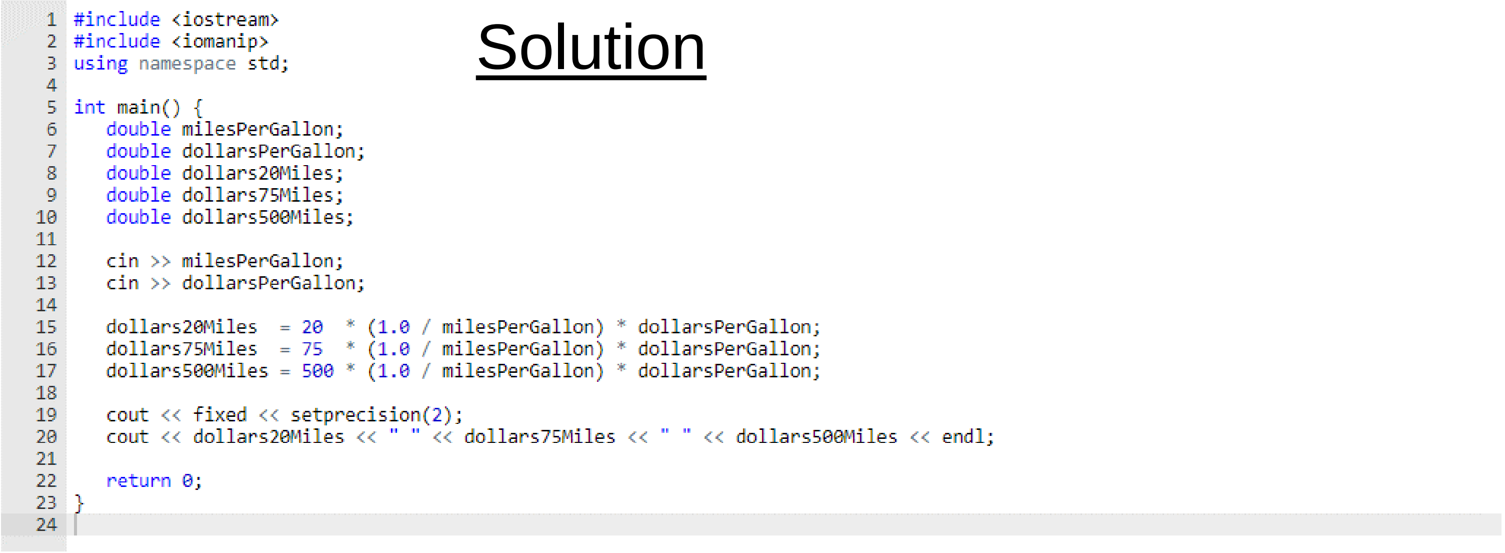
Output each floating-point value with two digits after the decimal point, which can be achieved by executing `cout << fixed << setprecision(2);` once before all other cout statements. Note: End with a newline.

Ex: If the input is:

```
20.0 3.1599
```

the output is:

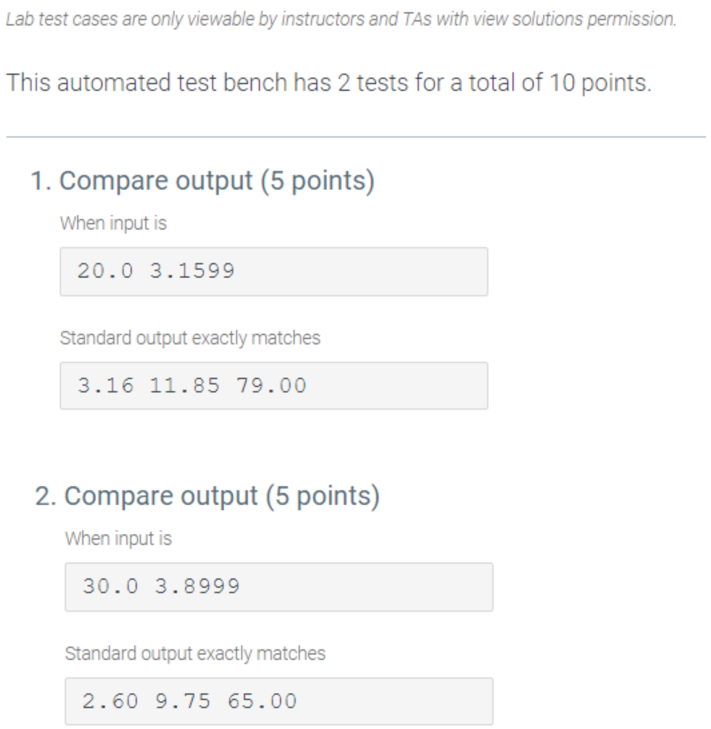
```
3.16 11.85 79.00
```



Solution

```cpp
#include <iostream>
#include <iomanip>
using namespace std;

int main() {
   double milesPerGallon;
   double dollarsPerGallon;
   double dollars20Miles;
   double dollars75Miles;
   double dollars500Miles;

   cin >> milesPerGallon;
   cin >> dollarsPerGallon;

   dollars20Miles  = 20  * (1.0 / milesPerGallon) * dollarsPerGallon;
   dollars75Miles  = 75  * (1.0 / milesPerGallon) * dollarsPerGallon;
   dollars500Miles = 500 * (1.0 / milesPerGallon) * dollarsPerGallon;

   cout << fixed << setprecision(2);
   cout << dollars20Miles << " " << dollars75Miles << " " << dollars500Miles << endl;

   return 0;
}

```

Test cases



Lab test cases are only viewable by instructors and TAs with view solutions permission.

This automated test bench has 2 tests for a total of 10 points.

1. Compare output (5 points)

When input is

```
20.0 3.1599
```

Standard output exactly matches

```
3.16 11.85 79.00
```

2. Compare output (5 points)

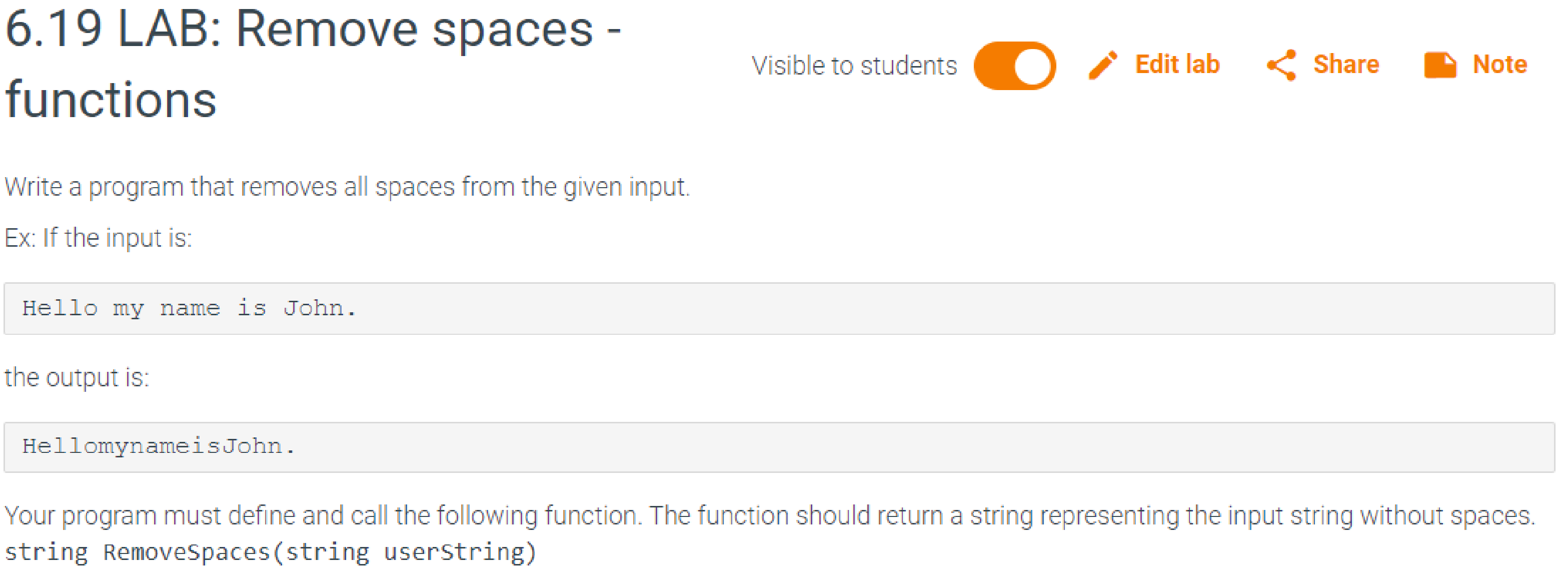
When input is

```
30.0 3.8999
```

Standard output exactly matches

```
2.60 9.75 65.00
```

MSP lab activity - sample 2



6.19 LAB: Remove spaces - functions

Write a program that removes all spaces from the given input.

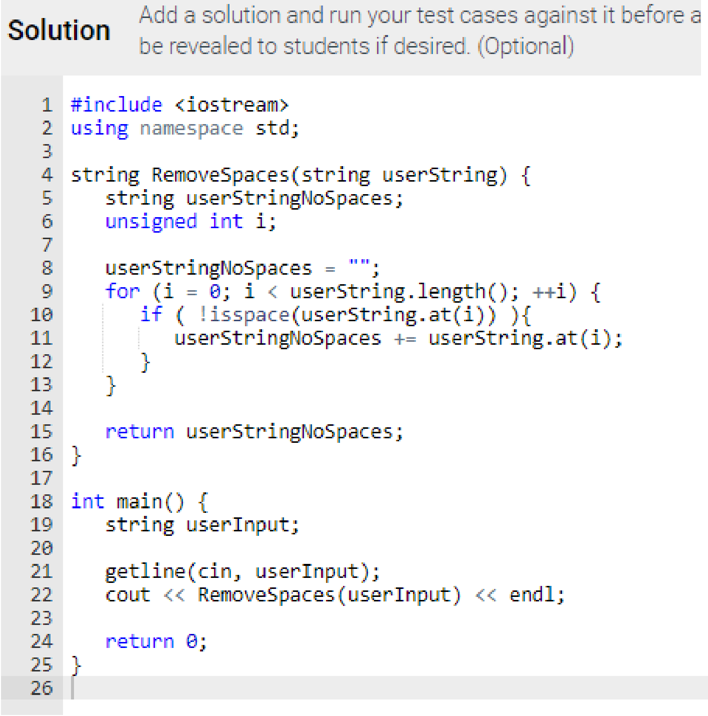
Ex: If the input is:

```
Hello my name is John.
```

the output is:

```
HellomynameisJohn.
```

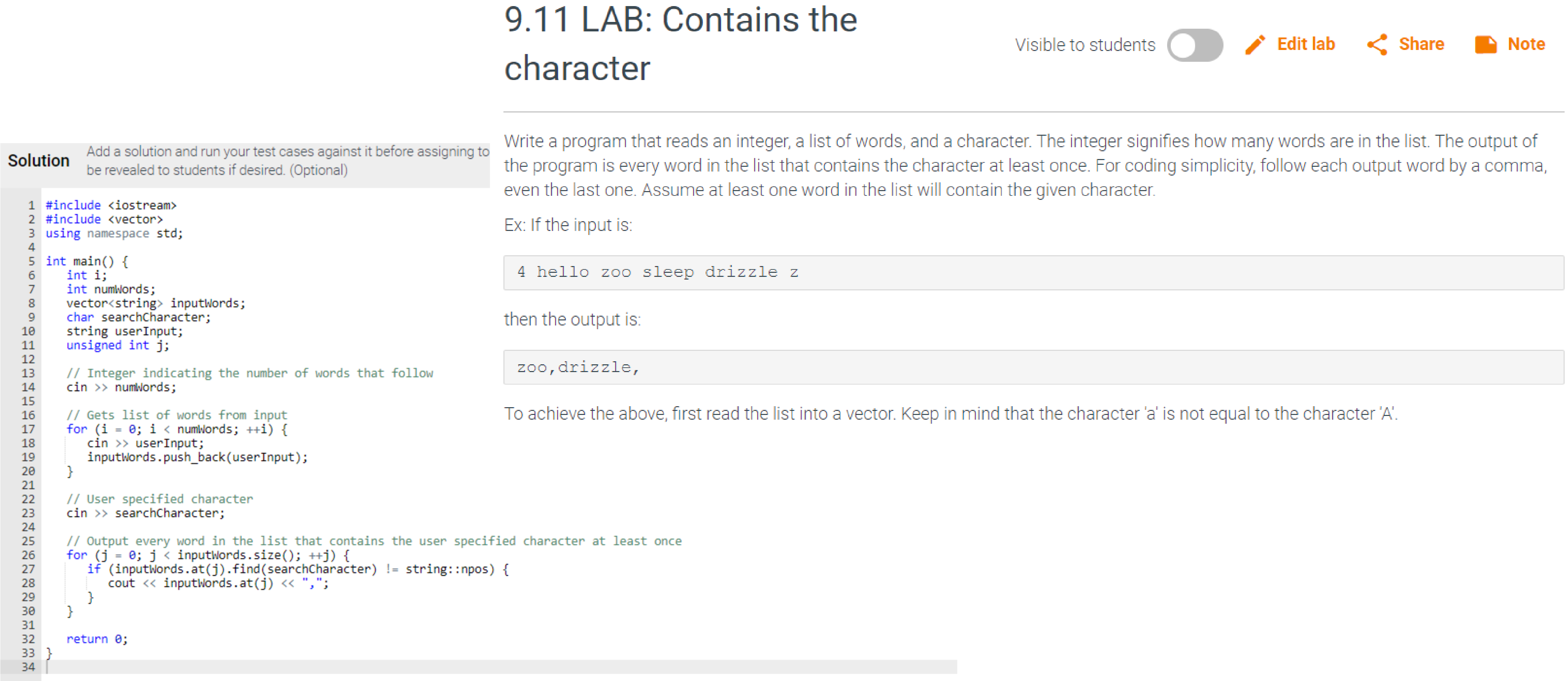
Your program must define and call the following function. The function should return a string representing the input string without spaces.
`string RemoveSpaces(string userString)`



Solution Add a solution and run your test cases against it before a
be revealed to students if desired. (Optional)

```cpp
#include <iostream>
using namespace std;

string RemoveSpaces(string userString) {
   string userStringNoSpaces;
   unsigned int i;

   userStringNoSpaces = "";
   for (i = 0; i < userString.length(); ++i) {
      if ( !isspace(userString.at(i)) ){
         userStringNoSpaces += userString.at(i);
      }
   }

   return userStringNoSpaces;
}

int main() {
   string userInput;

   getline(cin, userInput);
   cout << RemoveSpaces(userInput) << endl;

   return 0;
}

```

MSP lab activity - sample 3



9.11 LAB: Contains the character

Write a program that reads an integer, a list of words, and a character. The integer signifies how many words are in the list. The output of the program is every word in the list that contains the character at least once. For coding simplicity, follow each output word by a comma, even the last one. Assume at least one word in the list will contain the given character.

Ex: If the input is:

```
4 hello zoo sleep drizzle z
```

then the output is:

```
zoo,drizzle,
```

To achieve the above, first read the list into a vector. Keep in mind that the character 'a' is not equal to the character 'A'.

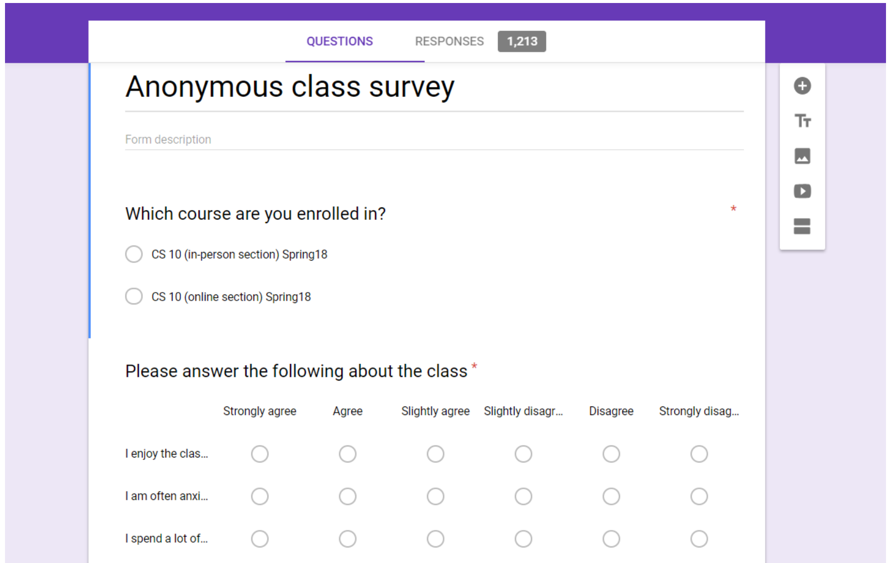
Solution Add a solution and run your test cases against it before assigning to be revealed to students if desired. (Optional)

```cpp
#include <iostream>
#include <vector>
using namespace std;

int main() {
   int i;
   int numWords;
   vector<string> inputWords;
   char searchCharacter;
   string userInput;
   unsigned int j;

   // Integer indicating the number of words that follow
   cin >> numWords;

   // Gets list of words from input
   for (i = 0; i < numWords; ++i) {
      cin >> userInput;
      inputWords.push_back(userInput);
   }

   // User specified character
   cin >> searchCharacter;

   // Output every word in the list that contains the user specified character at least once
   for (j = 0; j < inputWords.size(); ++j) {
      if (inputWords.at(j).find(searchCharacter) != string::npos) {
         cout << inputWords.at(j) << ",";
      }
   }

   return 0;
}
```

CS1 student satisfaction & grade performance

- RQ’s:
  - Does an MSP approach impact student satisfaction?
  - Does an MSP approach impact student grade performance?
- Methods
  - Student “stress” survey
    - Given week 8 of the quarter
    - Ask students about their experience
    - 18 questions: Strongly agree (6) to Strongly disagree (0)
  - Student grade performance
    - Participation, Challenge, and Lab Activities, Midterm, Final, Total grade

J.M. Allen, F. Vahid, K. Downey, A. Edgcomb. *Weekly Programs in a CS1 Class: Experiences with Auto-graded Many-small Programs (MSP).* ASEE Annual Conference, 2018. ***(Best paper nominee)***



Anonymous class survey

Form description

Which course are you enrolled in? *

○ CS 10 (in-person section) Spring18

○ CS 10 (online section) Spring18

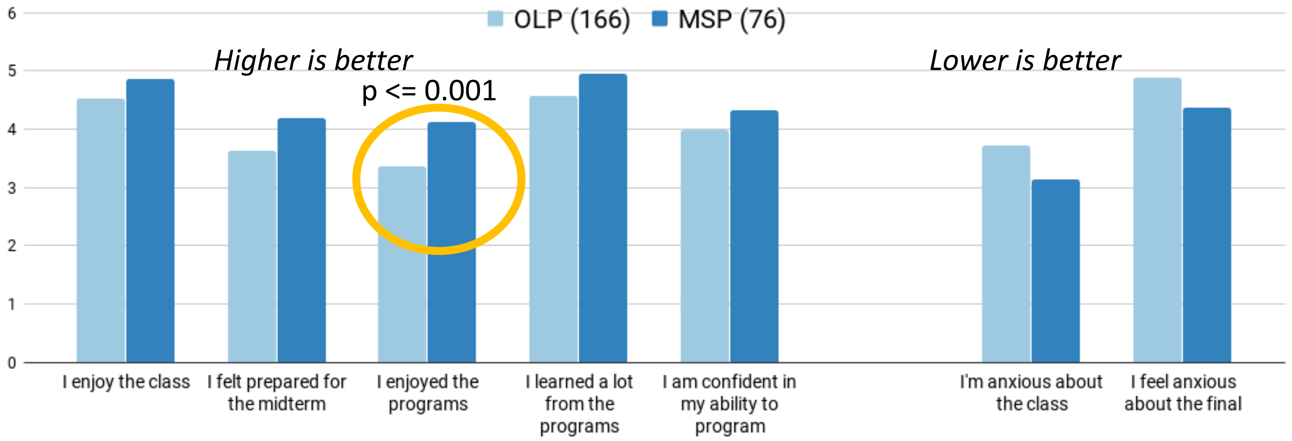
Please answer the following about the class *

	Strongly agree	Agree	Slightly agree	Slightly disagr...	Disagree	Strongly disag...
I enjoy the clas...	○	○	○	○	○	○
I am often anxi...	○	○	○	○	○	○
I spend a lot of...	○	○	○	○	○	○

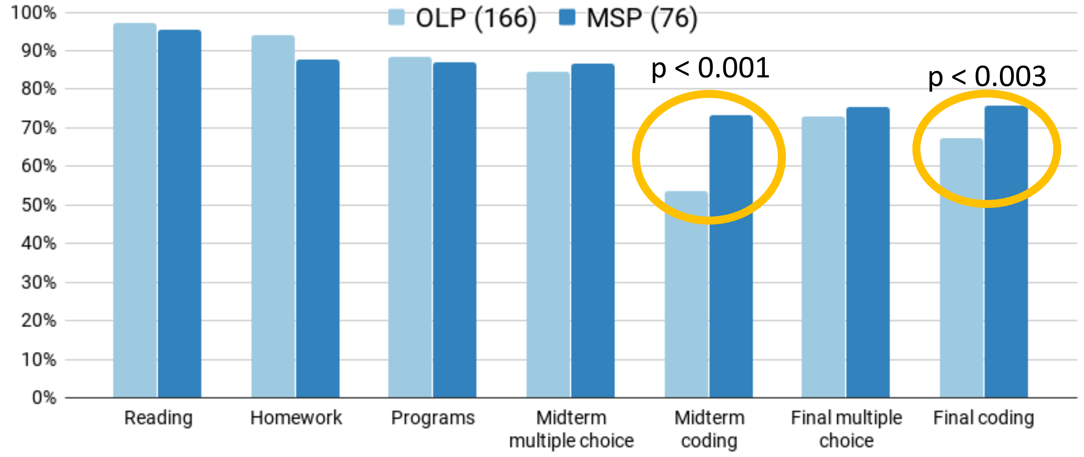
Findings

- MSP group had more favorable responses and enjoyed the class more
- MSP group student grade performance did not worsen (higher coding scores on exams)
- Results: UCR CS1 use an MSP approach, ~200 universities use MSPs, and zyBooks mimicked and now maintains MSPs

Student satisfaction – stress survey results



Student grade performance results

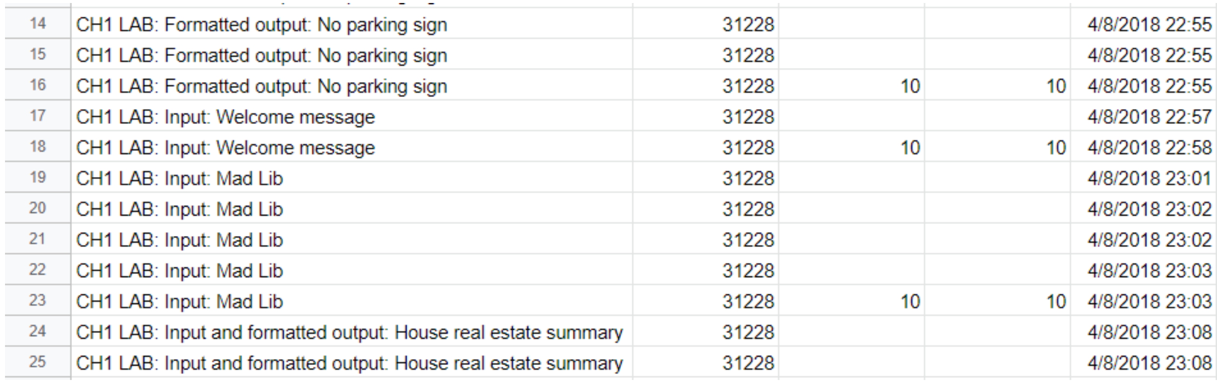


MSP usage analysis - UCR

- RQ’s:
  - How do students interact with MSPs?

J.M. Allen, F. Vahid, A. Edgcomb, K. Downey, and K. Miller. An Analysis of Using Many Small Programs in CS1, ACM SIGCSE Technical Symposium on Computer Science Education, 2019.

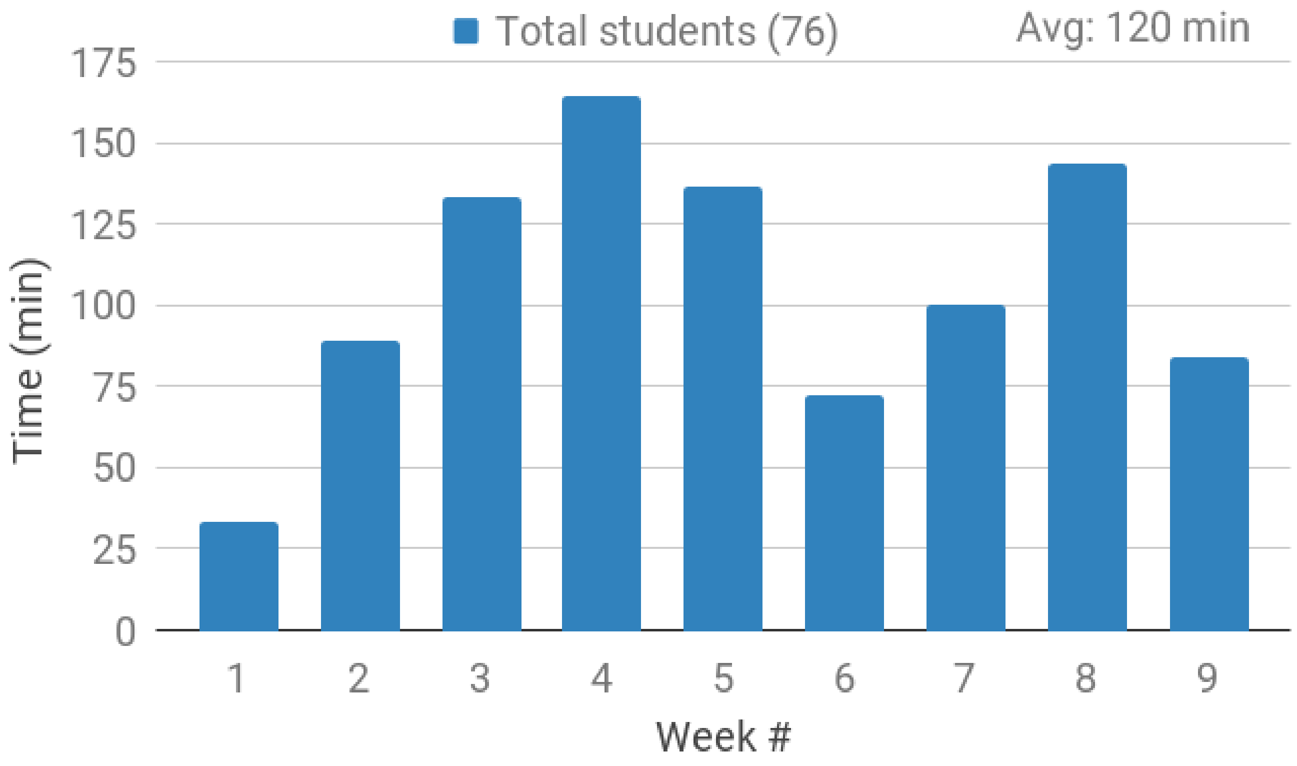
- Methods
  - UCR CS1 Spring 2017 MSP section: 76 students
  - zyLab metadata



	labID	userID	score	maxScore	timestamp
14	CH1 LAB: Formatted output: No parking sign	31228			4/8/2018 22:55
15	CH1 LAB: Formatted output: No parking sign	31228			4/8/2018 22:55
16	CH1 LAB: Formatted output: No parking sign	31228	10	10	4/8/2018 22:55
17	CH1 LAB: Input: Welcome message	31228			4/8/2018 22:57
18	CH1 LAB: Input: Welcome message	31228	10	10	4/8/2018 22:58
19	CH1 LAB: Input: Mad Lib	31228			4/8/2018 23:01
20	CH1 LAB: Input: Mad Lib	31228			4/8/2018 23:02
21	CH1 LAB: Input: Mad Lib	31228			4/8/2018 23:02
22	CH1 LAB: Input: Mad Lib	31228			4/8/2018 23:03
23	CH1 LAB: Input: Mad Lib	31228	10	10	4/8/2018 23:03
24	CH1 LAB: Input and formatted output: House real estate summary	31228			4/8/2018 23:08
25	CH1 LAB: Input and formatted output: House real estate summary	31228			4/8/2018 23:08

Q: How much time do students spend working on MSP assignments each week?

A: At least 120 min / week

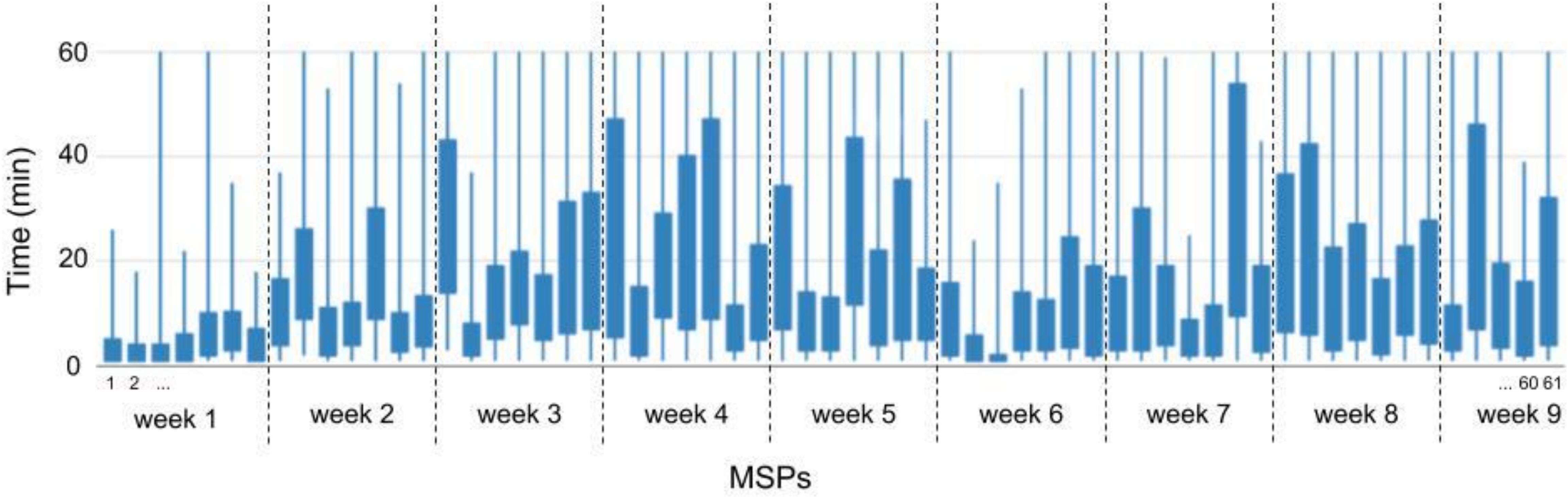


NOTE: ***Underestimate***.

Students with 0 subs or 0 time excluded. Avg is for weeks 2-8.

Q: How much time do students spend working on each MSP lab activity?

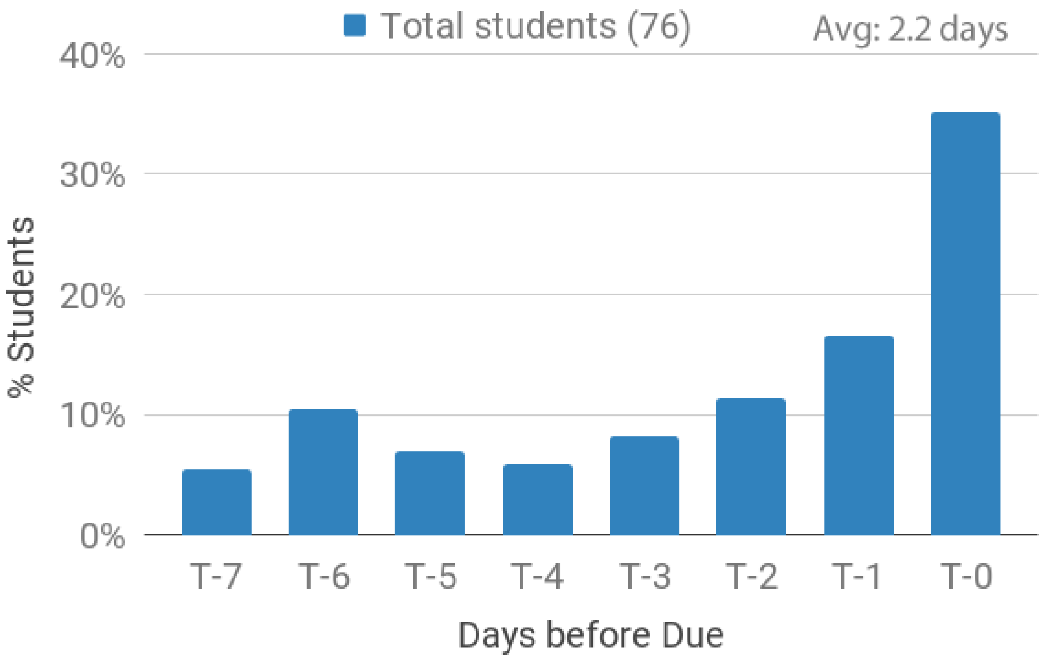
A: About 17 min / MSP



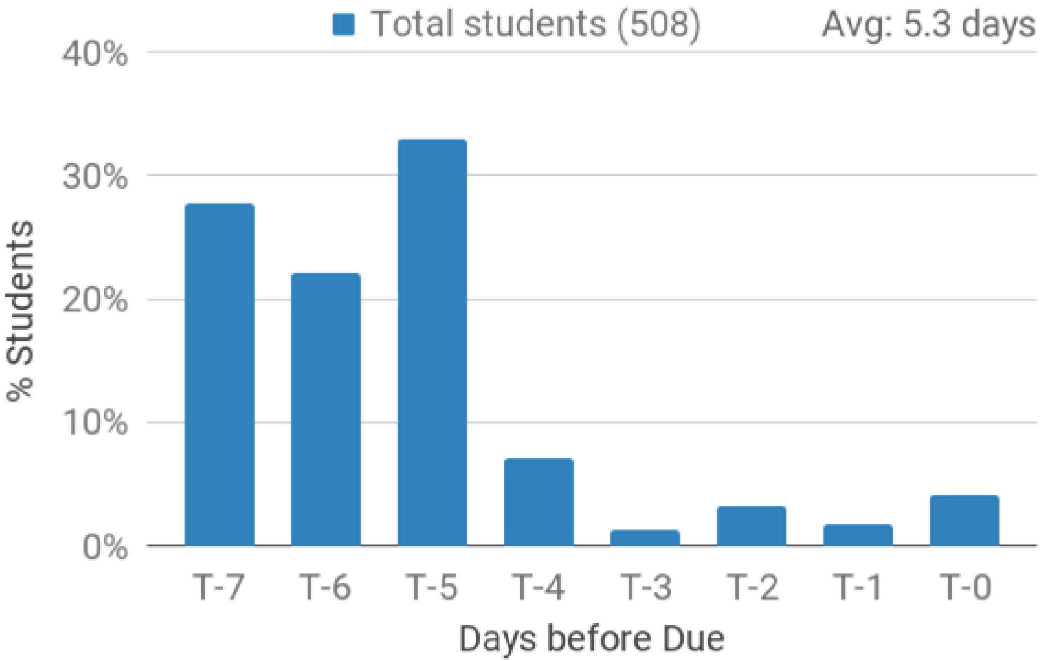
Average time spent per MSP - 17 min / MSP activity (weeks 1 and 9 excluded).

Q: How many days before the due date do students start working on MSP assignments?

A: MSPs started 2.2 days before due date

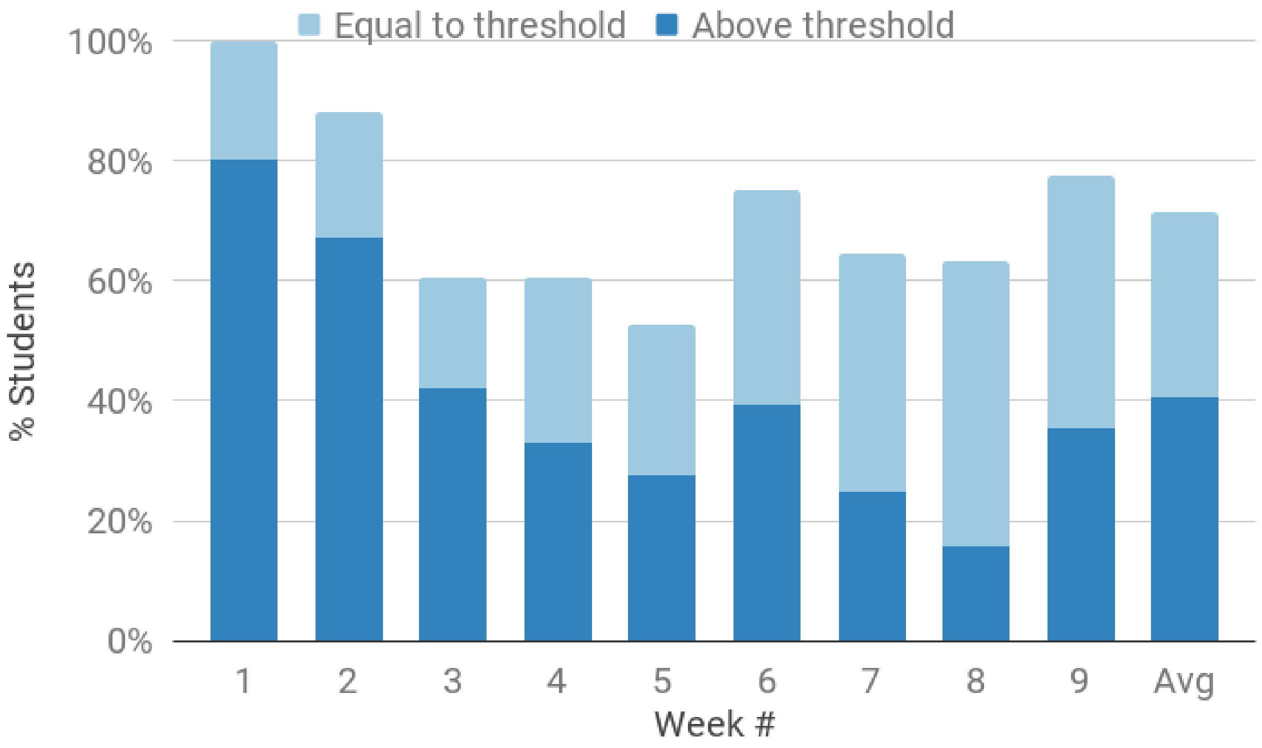


A: With policy adjustment in Fall 2018, started 5.3 days before



Q: Given a full-credit threshold, do students complete more MSP lab activities than required?

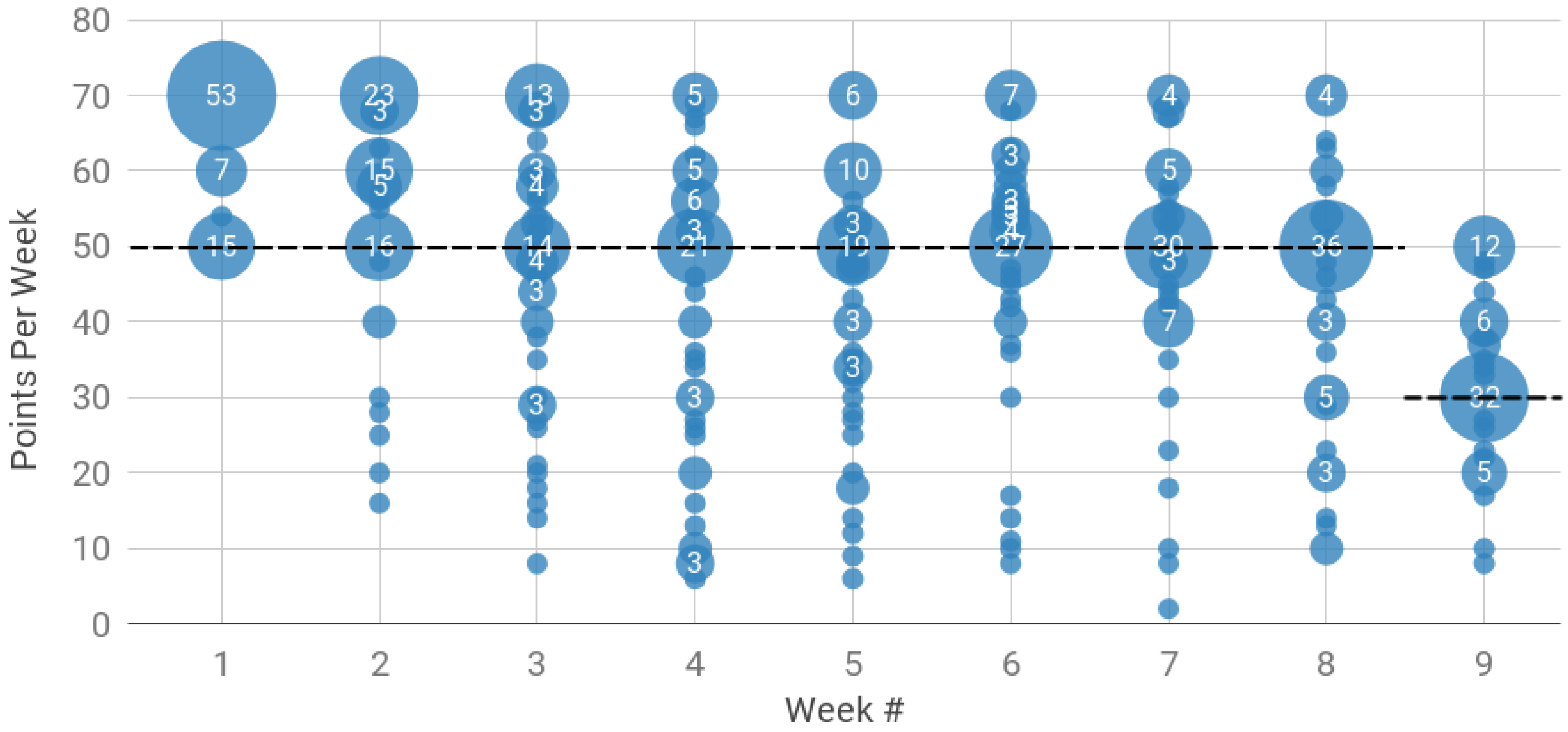
A: 40% of students completed more MSP lab activities than required



No extra credit given for exceeding full-credit threshold

Q: Given a full-credit threshold, how many points do students score each week?

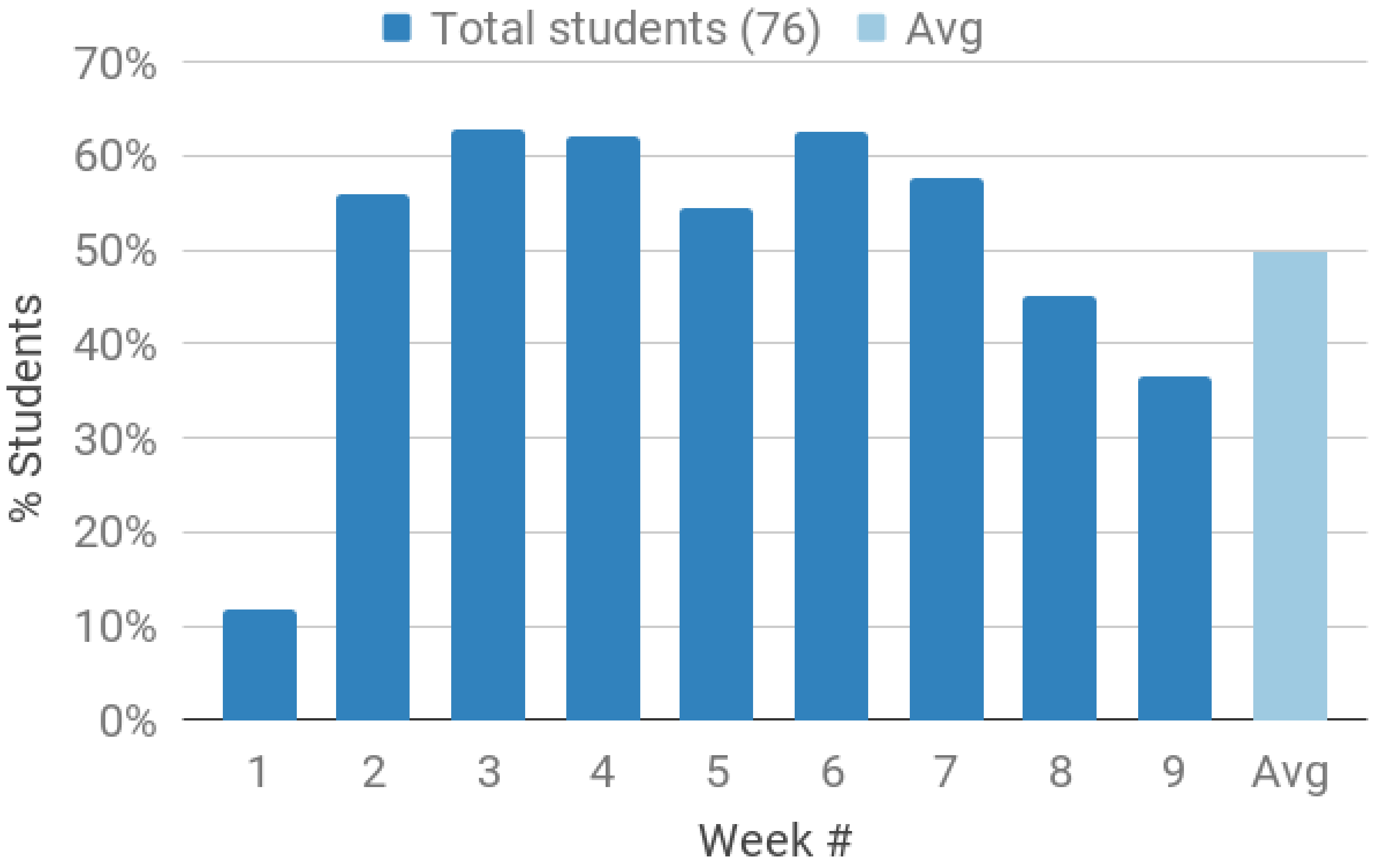
A: Total points per week – Avg 13 more points



Bubble size represents number of students. Dashed line indicates full-credit threshold.
Students who scored 0 points for a week excluded.

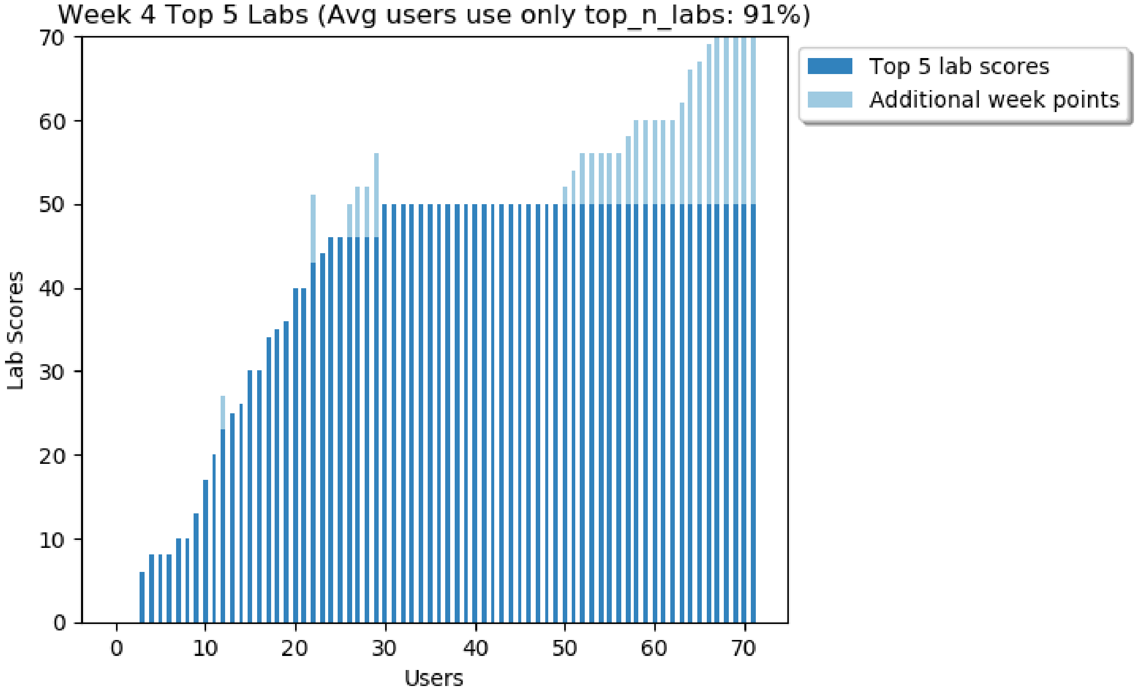
Q: Do students pivot, or help themselves when stuck?

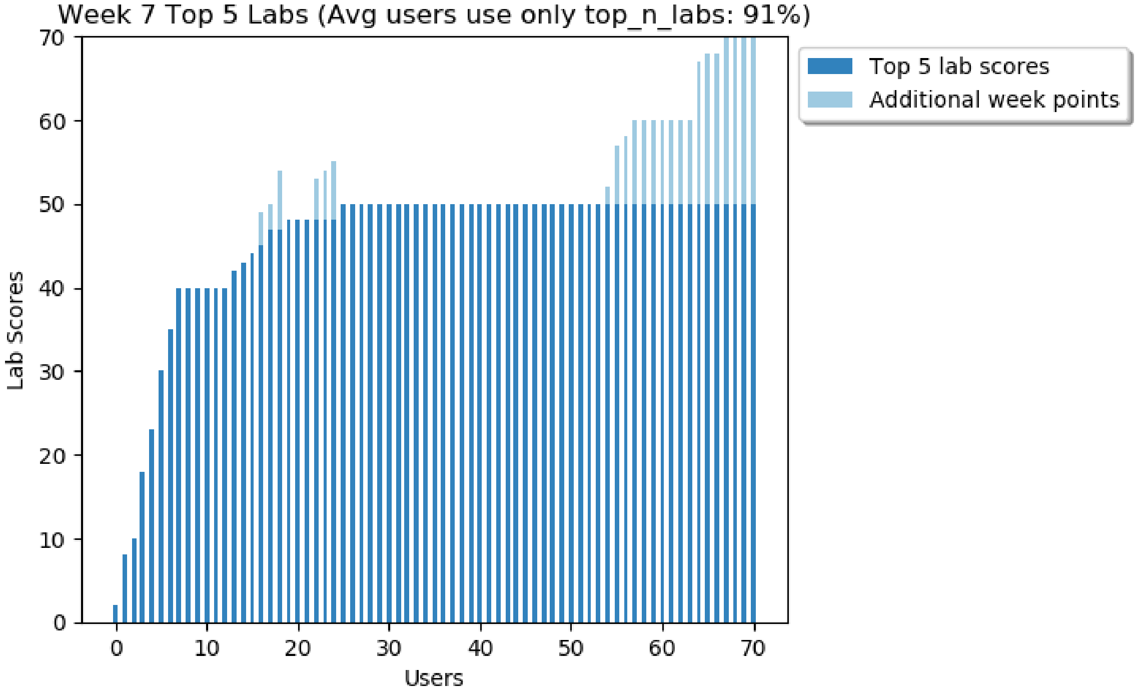
A: Each week, 50% of students pivoted (avg. 1.3 pivots)



Q: Do students skip the ‘hard parts’ of lab activities?

A: ~95% of students score full credit using their top 5 highest scores


Week 4 Top 5 Labs (Avg users use only top_n_labs: 91%)


Week 7 Top 5 Labs (Avg users use only top_n_labs: 91%)

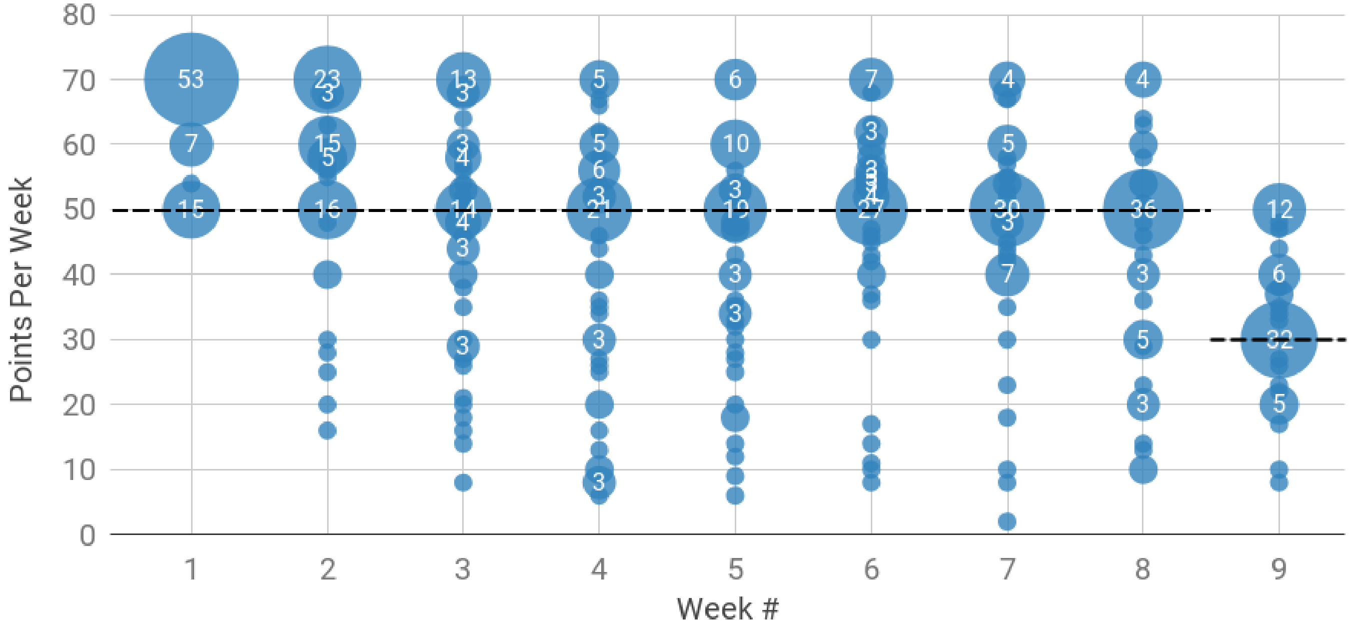
Q: Do students use MSP lab activities to study for exams?

A. Yes, students use MSPs to study for exams

Total number of students	76
Total number of MSPs	61
% of students that used MSPs to study for the midterm	38%
% of students that used MSPs to study for the final	37%
% of students that used MSPs to study for either exam	**54%**
% of MSPs that were used to study for the midterm	97%
% of MSPs that were used to study for the final	90%
% of MSPs that were used to study for either exam	**98%**

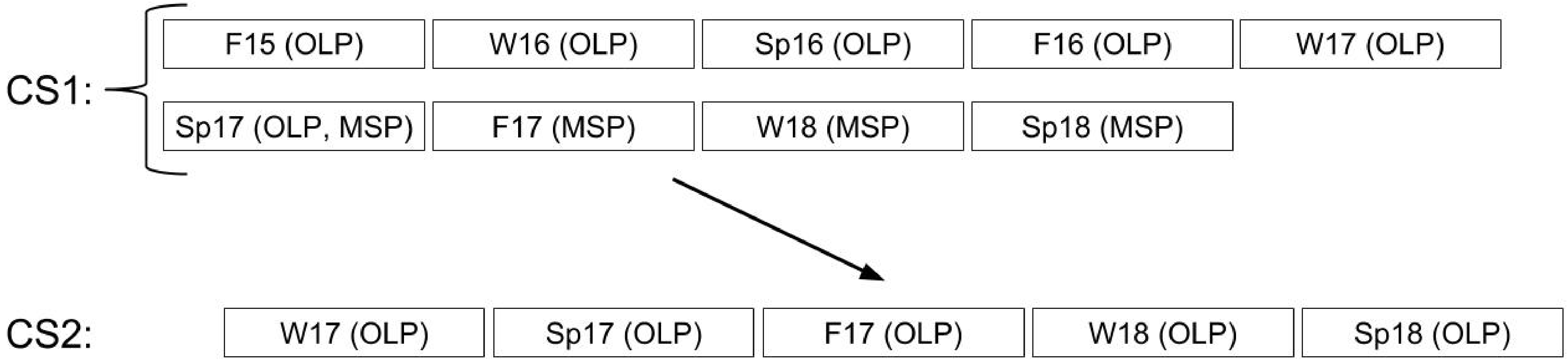
Findings

- Students make good use of MSP assignments
  - Sufficient time
  - Started early
  - Completed more than necessary
  - Pivoted to help selves when stuck
  - Used MSPs to study for exams



CS2 performance by MSP-trained CS1 students

- RQ's:
  - Won't MSP-trained students from CS1 do poorly in an OLP CS2



CS1:

F15 (OLP)
W16 (OLP)
Sp16 (OLP)
F16 (OLP)
W17 (OLP)

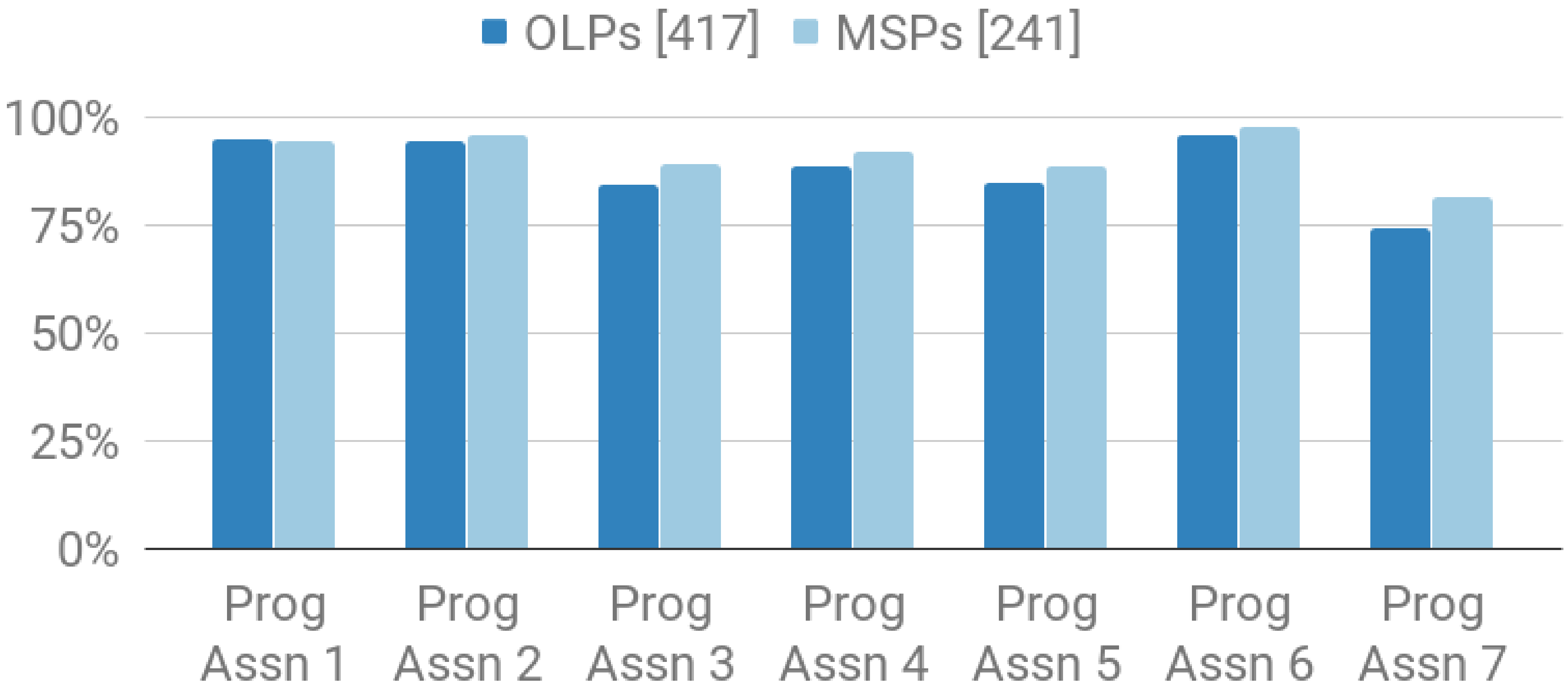
Sp17 (OLP, MSP)
F17 (MSP)
W18 (MSP)
Sp18 (MSP)

CS2:

W17 (OLP)
Sp17 (OLP)
F17 (OLP)
W18 (OLP)
Sp18 (OLP)

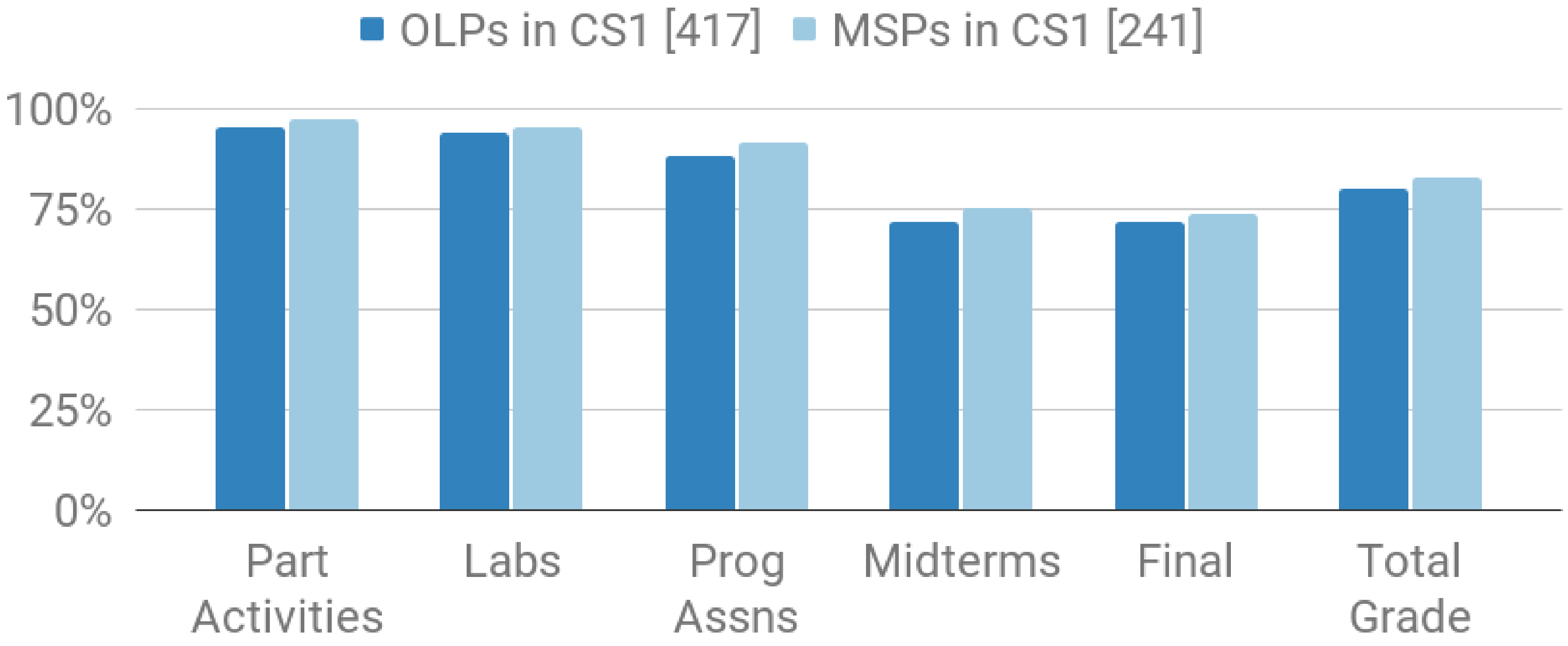
CS2 performance by MSP-trained CS1 students

- CS2 OLP programming assignments

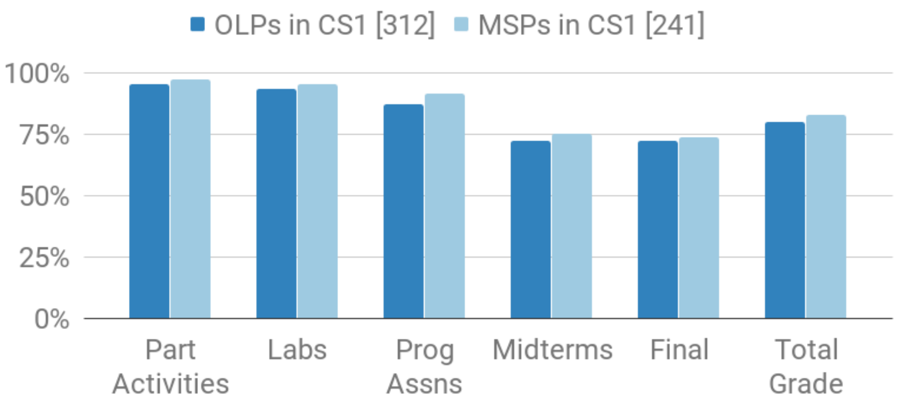


CS2 performance by MSP-trained CS1 students

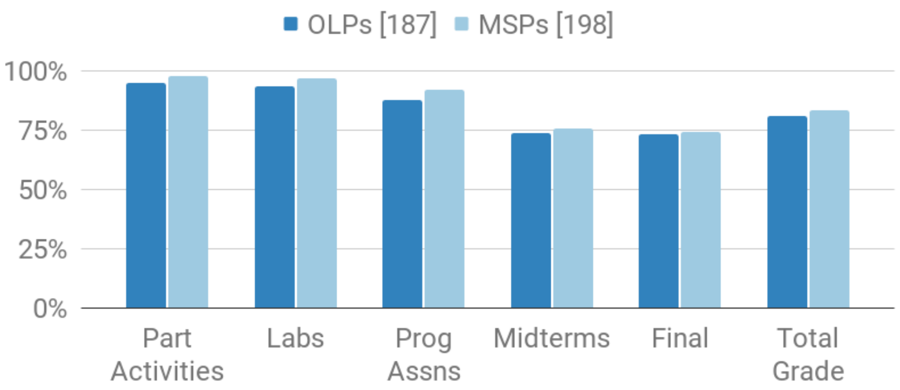
- CS2 class categories



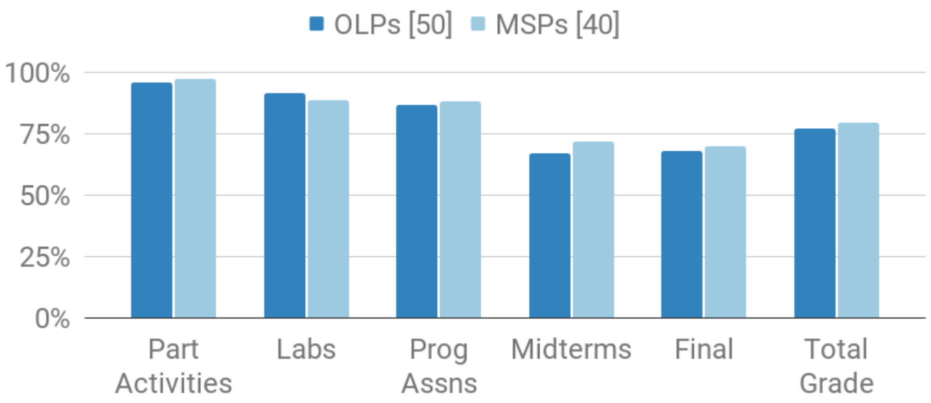
CS2 performance by MSP-trained CS1 students



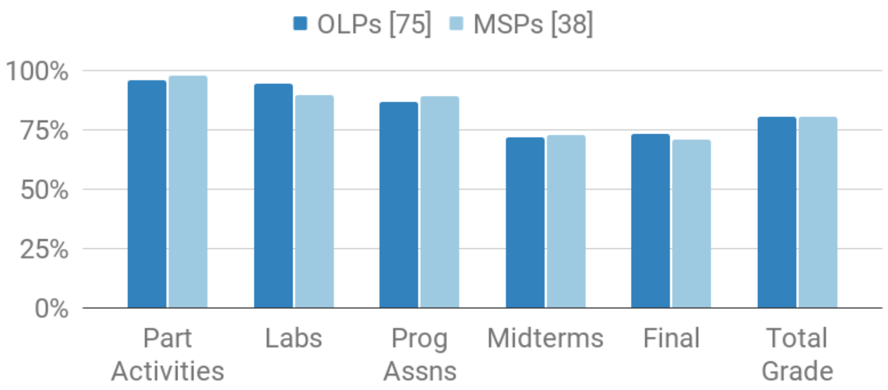
UCR Students only



0 quarter gap



1 quarter gap



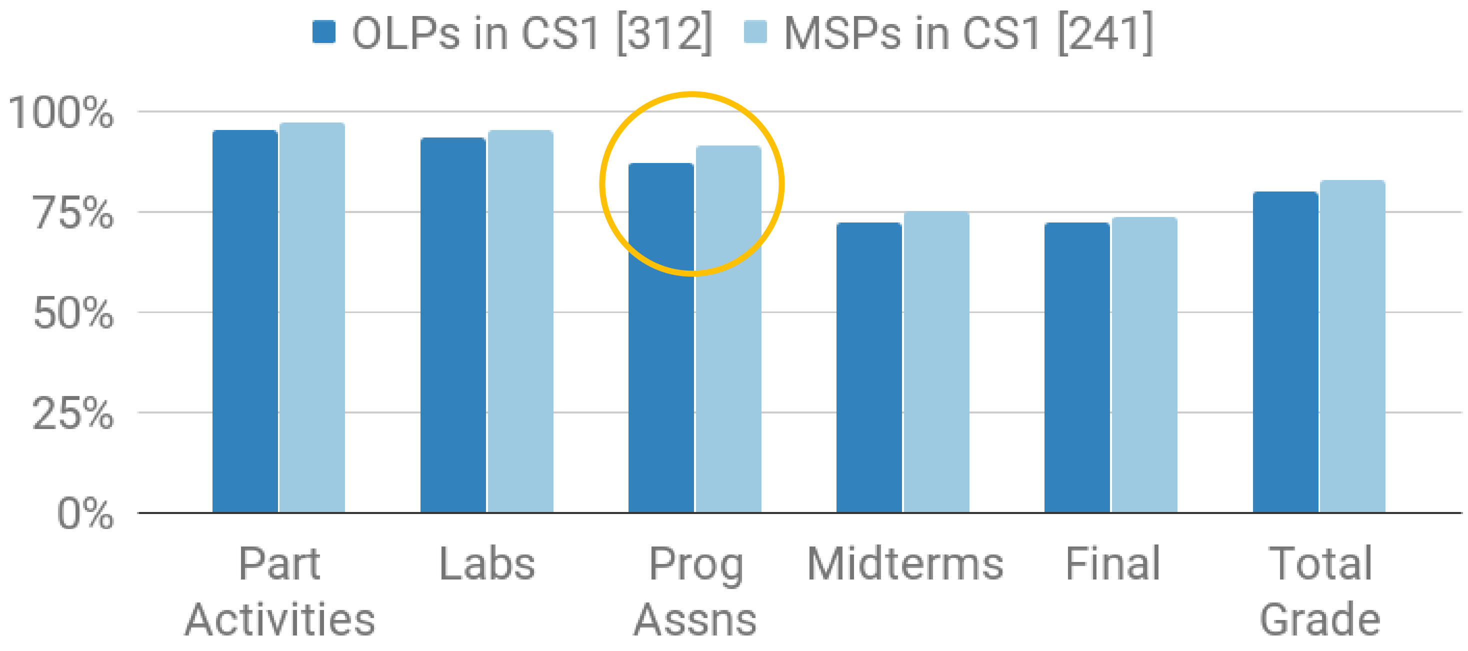
2+ quarter gap

Findings

Q: *Won't MSP CS1 students do poorly in an OLP CS2?*

A. MSP-trained CS1 students do just as well as OLP-trained students in an OLP CS2, in fact slightly better

CS2
scores



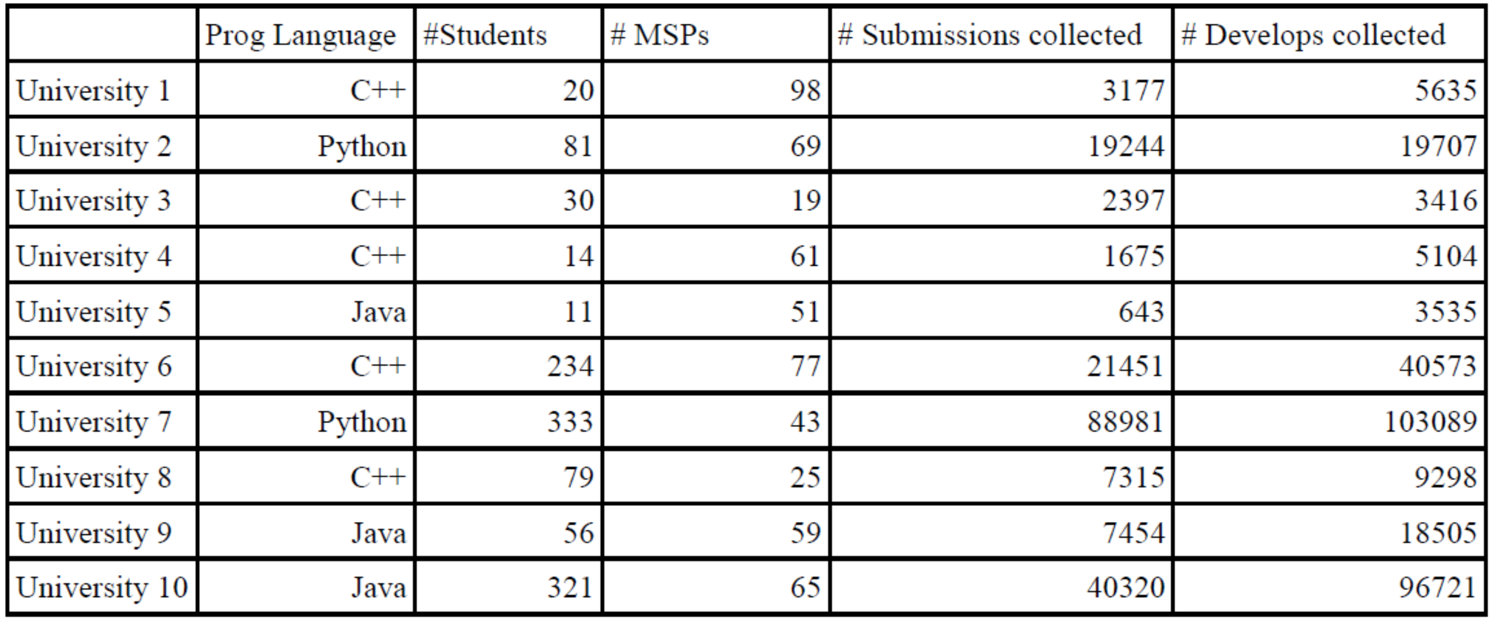
Outline

- Background & Related work
- Our experience with an MSP approach
- **MSP approach across universities & programming languages**
- MSP approach tools & analysis
- Contributions

MSP usage analysis - other universities

- RQ’s:
  - How do students interact with MSP assignments at other universities?

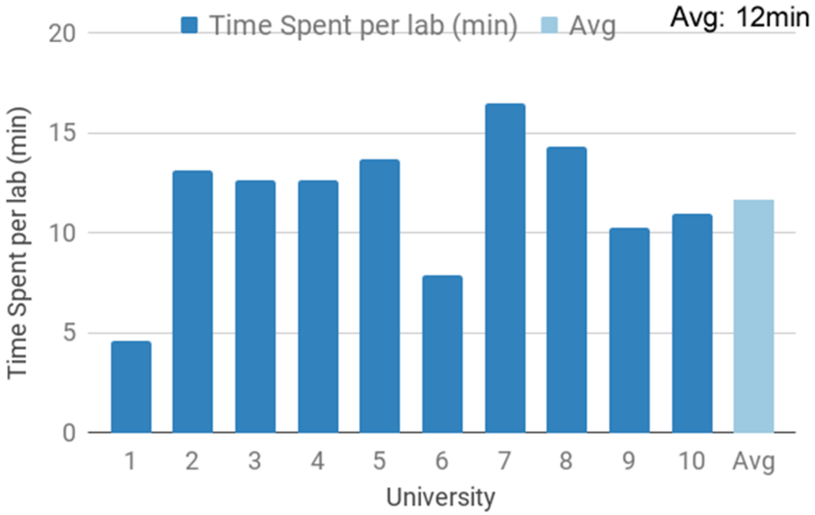
J.M. Allen, F. Vahid, K. Downey, K. Miller, and A. Edgcomb. Many Small Programs in CS1: Usage Analysis from Multiple Universities, Proceedings of ASEE Annual Conference, 2019.

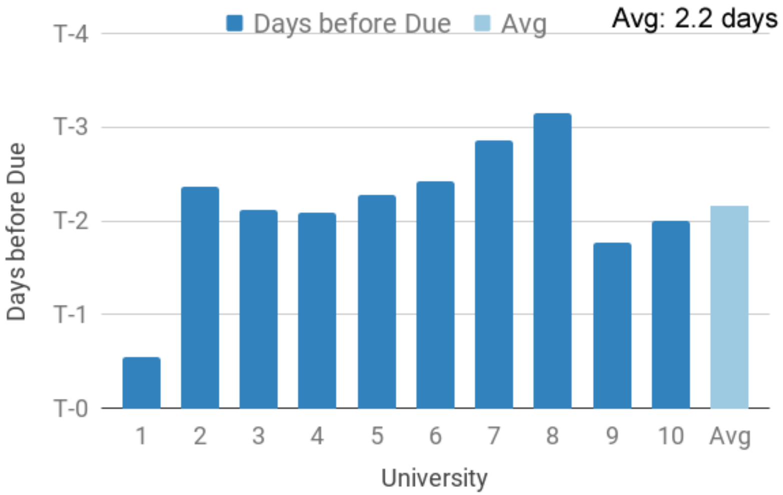


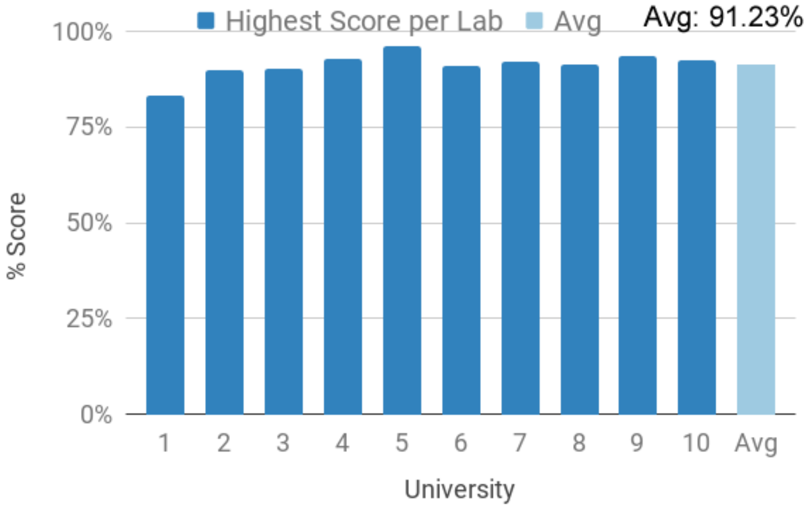
	Prog Language	#Students	# MSPs	# Submissions collected	# Develops collected
University 1	C++	20	98	3177	5635
University 2	Python	81	69	19244	19707
University 3	C++	30	19	2397	3416
University 4	C++	14	61	1675	5104
University 5	Java	11	51	643	3535
University 6	C++	234	77	21451	40573
University 7	Python	333	43	88981	103089
University 8	C++	79	25	7315	9298
University 9	Java	56	59	7454	18505
University 10	Java	321	65	40320	96721

Findings

- Similar results from other universities
  - Spend sufficient time (avg 12min per lab)
  - Start early (avg 2.2 days)
  - Complete most MSPs (avg 91% completion)



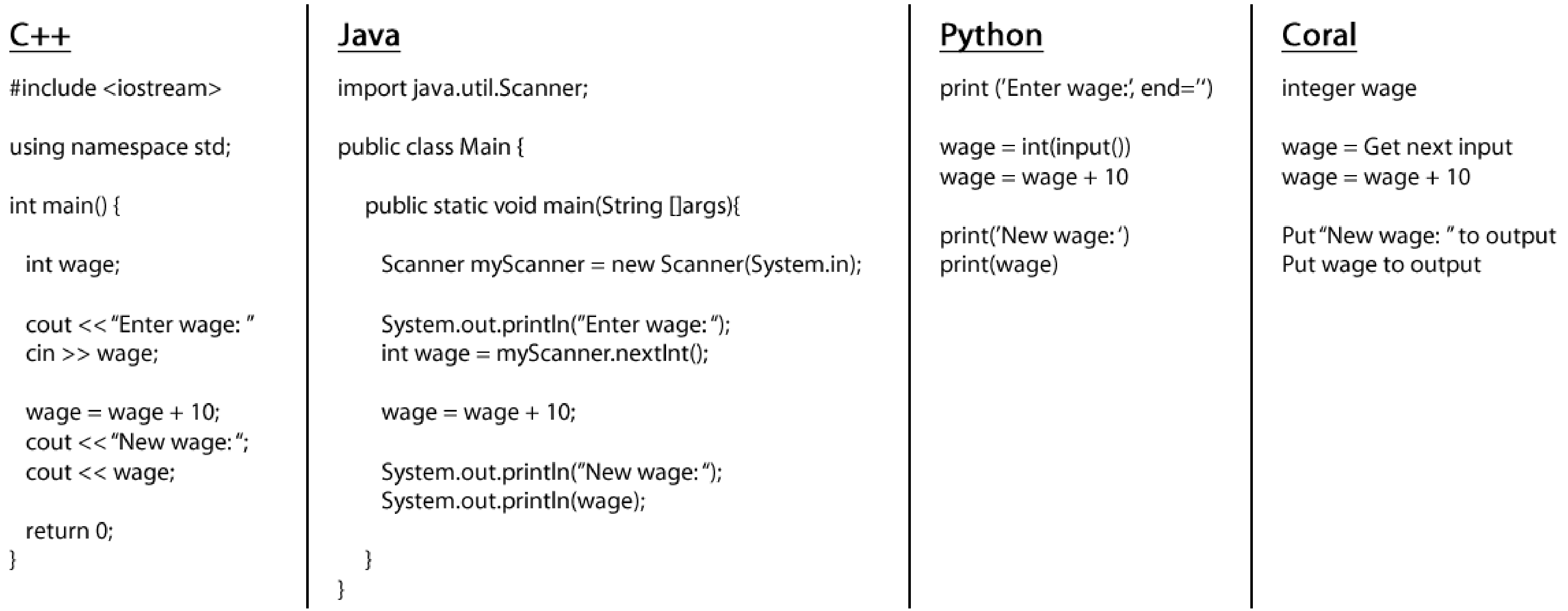




Experience with Coral MSP assignments in CS1

- Coral programming language
  - Ultra simple, pseudocode-like code
  - Designed for learners

J.M. Allen and F. Vahid. An Analysis of Using Coral Many Small Programs in CS1, Journal of Computing Sciences in Colleges, 2021.



C++

```
#include <iostream>

using namespace std;

int main() {

  int wage;

  cout << “Enter wage: ”
  cin >> wage;

  wage = wage + 10;
  cout << “New wage: “;
  cout << wage;

  return 0;
}
```

Java

```
import java.util.Scanner;

public class Main {

   public static void main(String []args){

      Scanner myScanner = new Scanner(System.in);

      System.out.println(”Enter wage: “);
      int wage = myScanner.nextInt();

      wage = wage + 10;

      System.out.println(”New wage: “);
      System.out.println(wage);

   }
}
```

Python

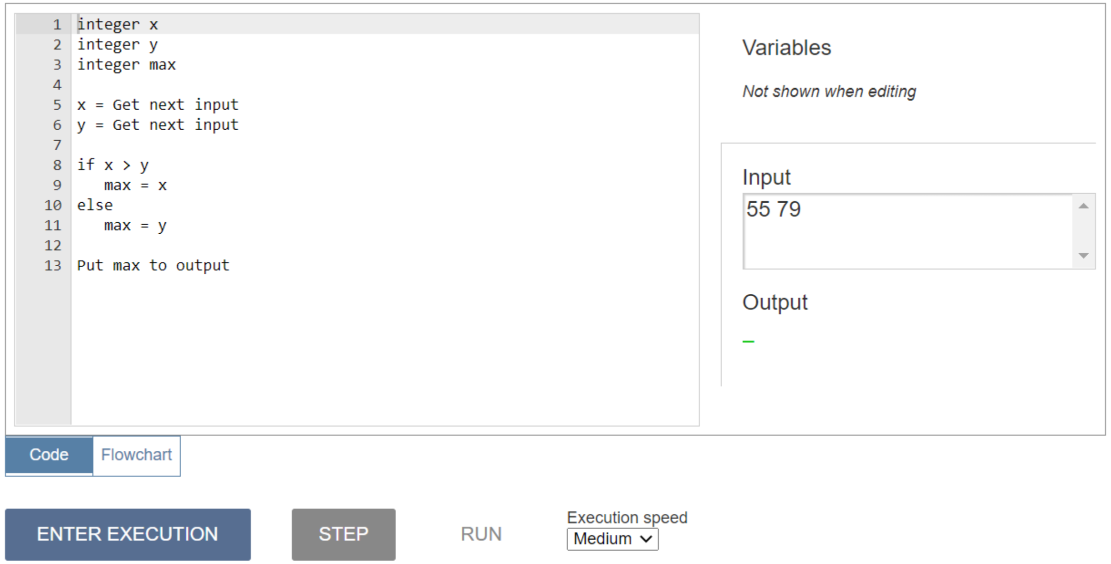
```
print (‘Enter wage:’, end=‘’)

wage = int(input())
wage = wage + 10

print(‘New wage: ‘)
print(wage)
```

Coral

```
integer wage

wage = Get next input
wage = wage + 10

Put “New wage: ” to output
Put wage to output
```

Experience with Coral MSP assignments in CS1

- Web-based simulator
- Web-based flow chart language



```
integer x
integer y
integer max

x = Get next input
y = Get next input

if x > y
   max = x
else
   max = y

Put max to output
```

Variables

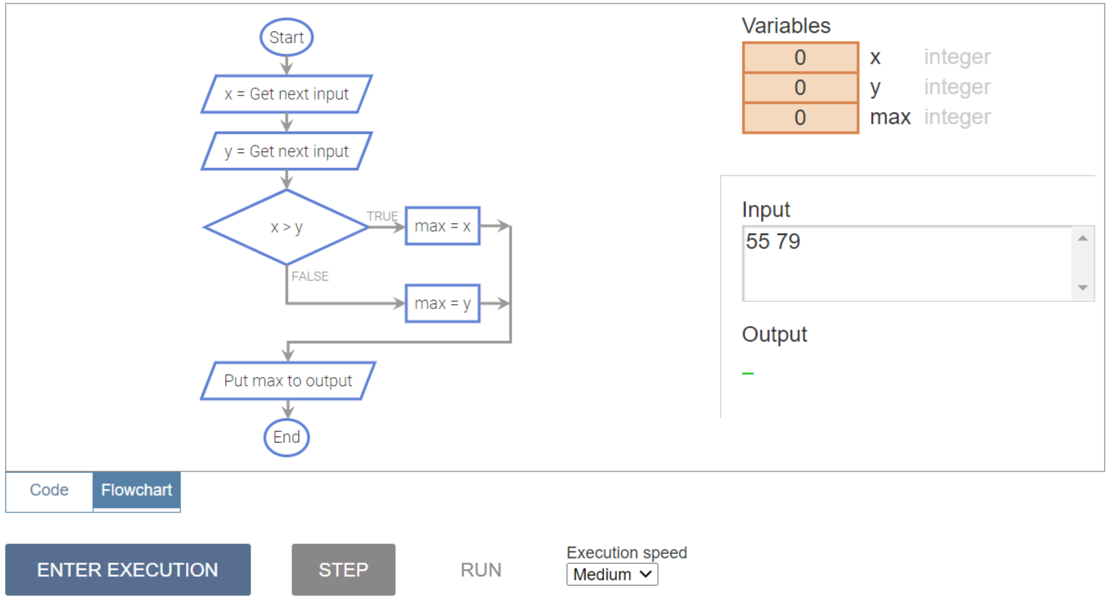
Not shown when editing

Input

55 79

Output

–



Variables

0	x	integer
0	y	integer
0	max	integer

Input

55 79

Output

–

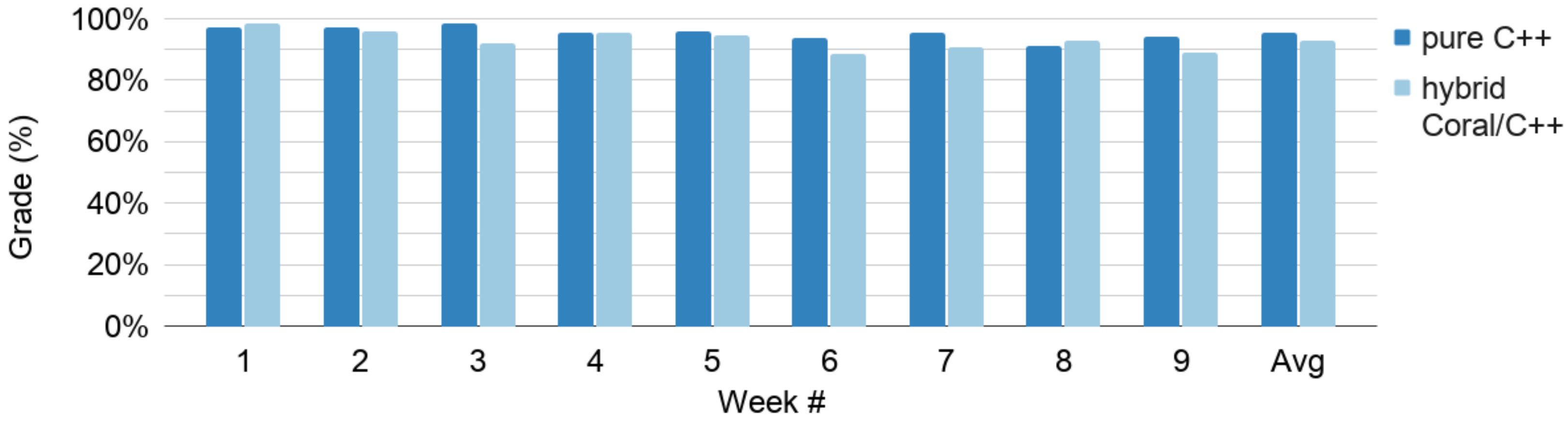
Experience with Coral MSP assignments in CS1

- 3 weeks of Coral, 7 weeks of C++
- Hybrid Coral/C++ vs. Pure C++
  - Grade performance?
  - Time spent?
  - Develops/Submits?
  - Start date?
  - Pivots?

Class category	Pure C++	Hybrid Coral/C++
Total class grade	88%	95%
Final exam	83%%	88%
Midterm exam	83%	95%
Participation activities	94%	95%
Challenge activities	94%	95%
Lab activities	96%	93%

Findings

- Coral/C++ did not harm student grade performance
- MSP usage is healthy
  - Time spent
  - Submit/Develop
  - Start date
  - Pivots



MSP assignments grade performnce

- Easier time teaching programming fundamentals

Outline

- Background & Related work
- Our experience with an MSP approach
- MSP approach across universities & programming languages
- **MSP approach tools & analysis**
- Contributions

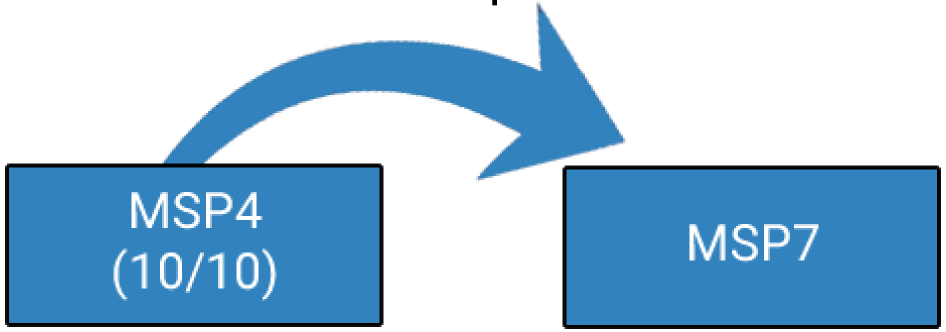
MSP pivot analysis

J.M. Allen and F. Vahid. Analyzing Pivoting Among Weekly Many Small Programs in a CS1 Course, Proceedings of ASEE Annual Conference, 2020.

- RQ's:
  - Do students make use of pivoting with MSP lab activities?

- Pivot definition:
  - When a student switches to a different lab activity before completing the previous one first

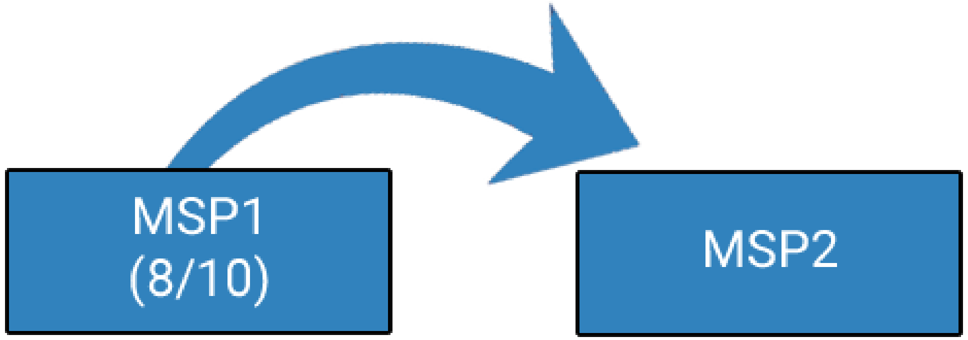
Not a pivot



MSP4
(10/10)

MSP7

Pivot

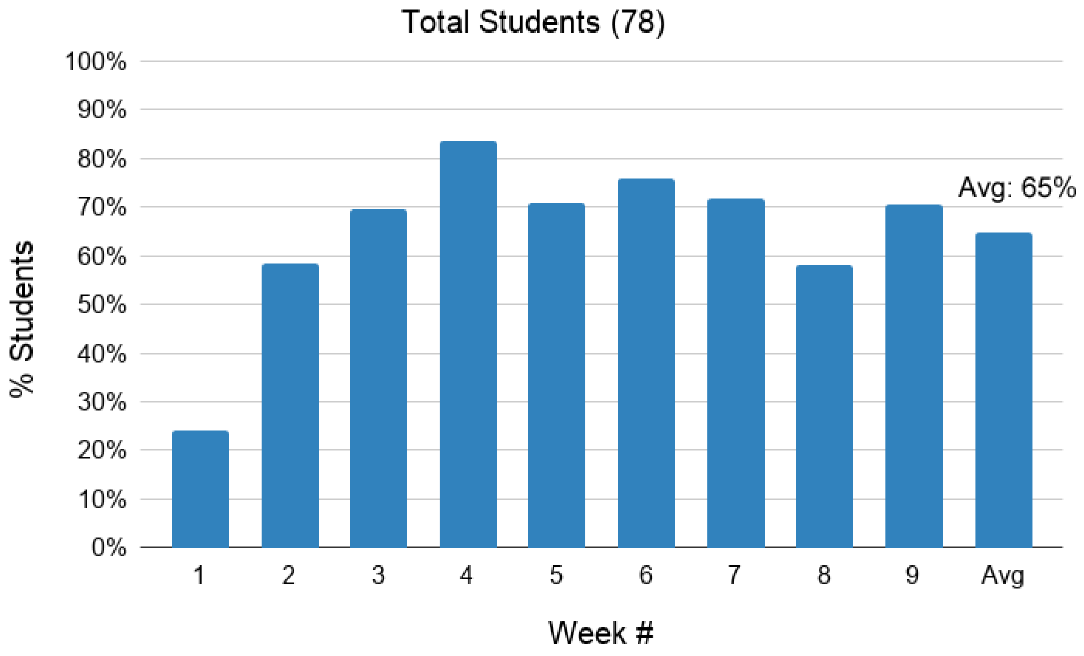


MSP1
(8/10)

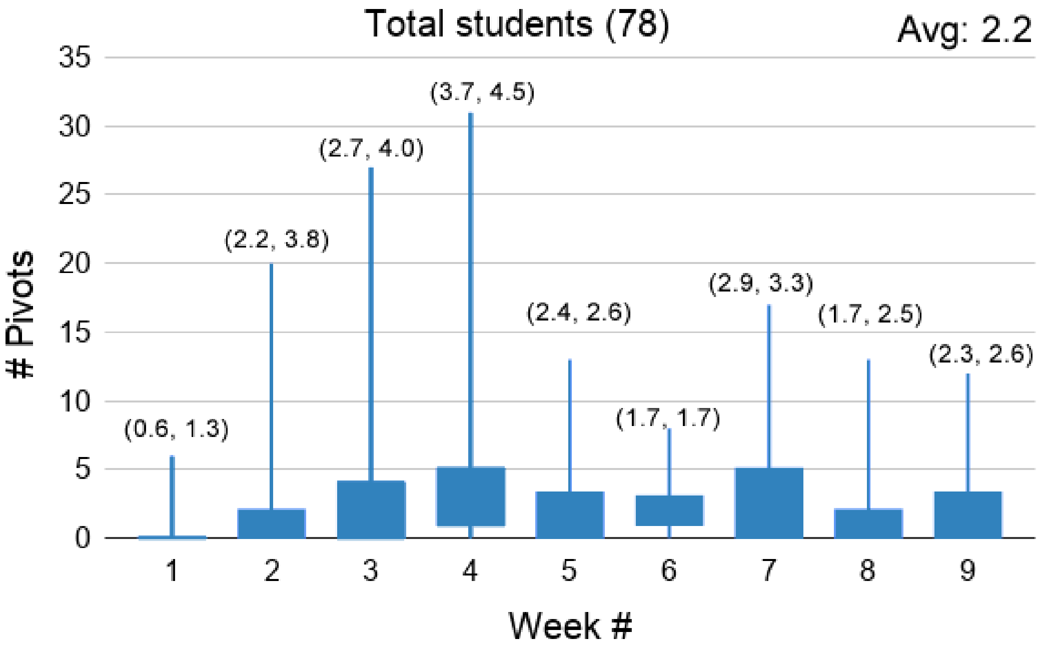
MSP2

Pivot outcomes

% Students that pivot each week

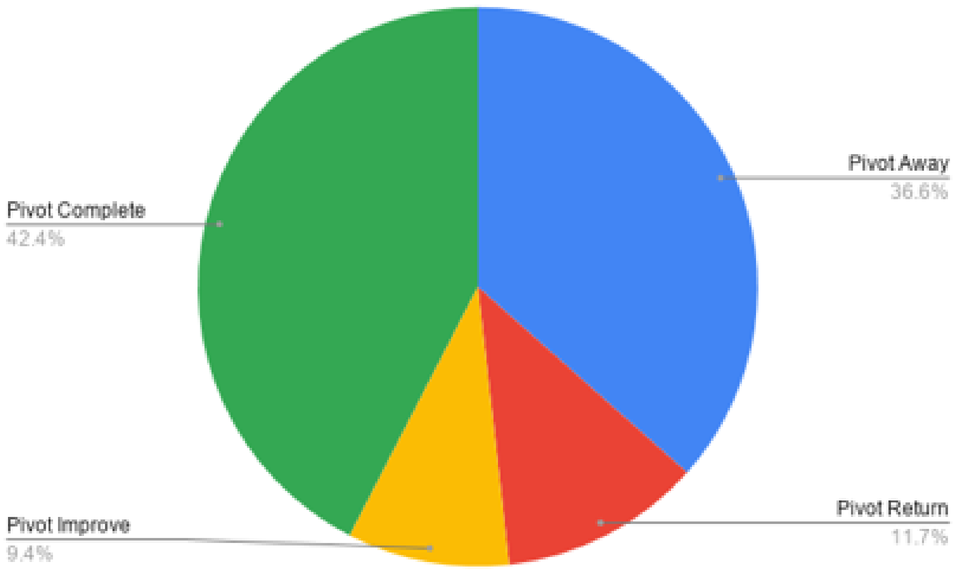


Pivots each week (avg, stdev)



Pivot outcomes

- Pivot none: did not pivot
- Pivot away: pivoted, and did not return
- Pivot return: pivoted, returned, but made no improvement in score
- Pivot improve: pivoted, returned, and improved their previous score
- Pivot complete: pivoted, returned, and completed the lab activity fully (scored 100%)

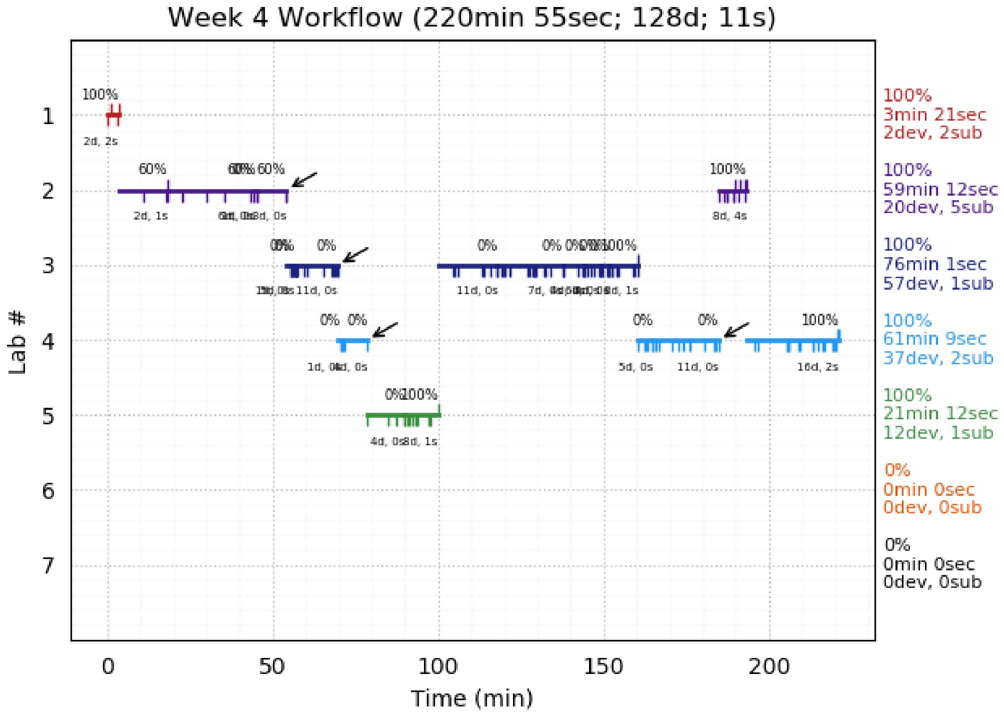


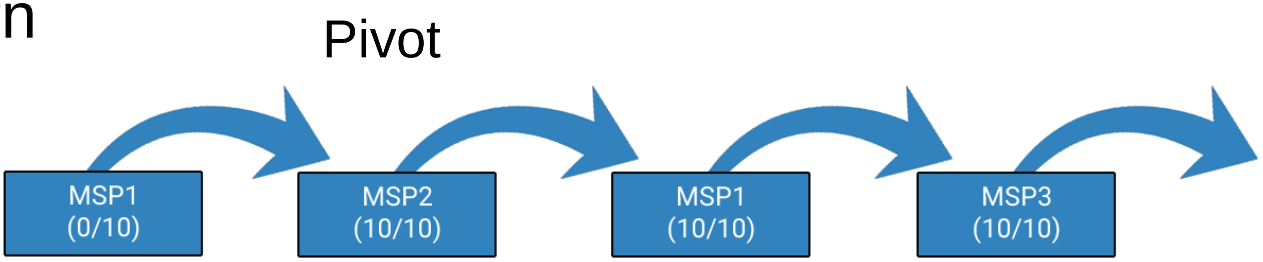
Pivot interviews

- What is most helpful about pivoting?
  - “[I] Get to work on other labs and see what they ask for. [You] get to look at something new and come back with a fresh mind. Get 100 for the new one and come back feeing less frustrated”
  - “The ability to go forward in lab and then come back with some new information that you have learned”
  - “There are times when I get frustrated and can’t get done. [I] tend to move on to see if future code can help to find my error, the [lab] I’m struggling with”

Findings

- Students make use of pivoting
  - 65% of students pivot each week
  - Avg 2.2 pivots each week
- Programming workflow charts help us visually recognize pivot patterns
- Students find pivoting helpful, often using to avoid frustration and learn from other lab activities





Pivot

MSP1 (0/10)

MSP2 (10/10)

MSP1 (10/10)

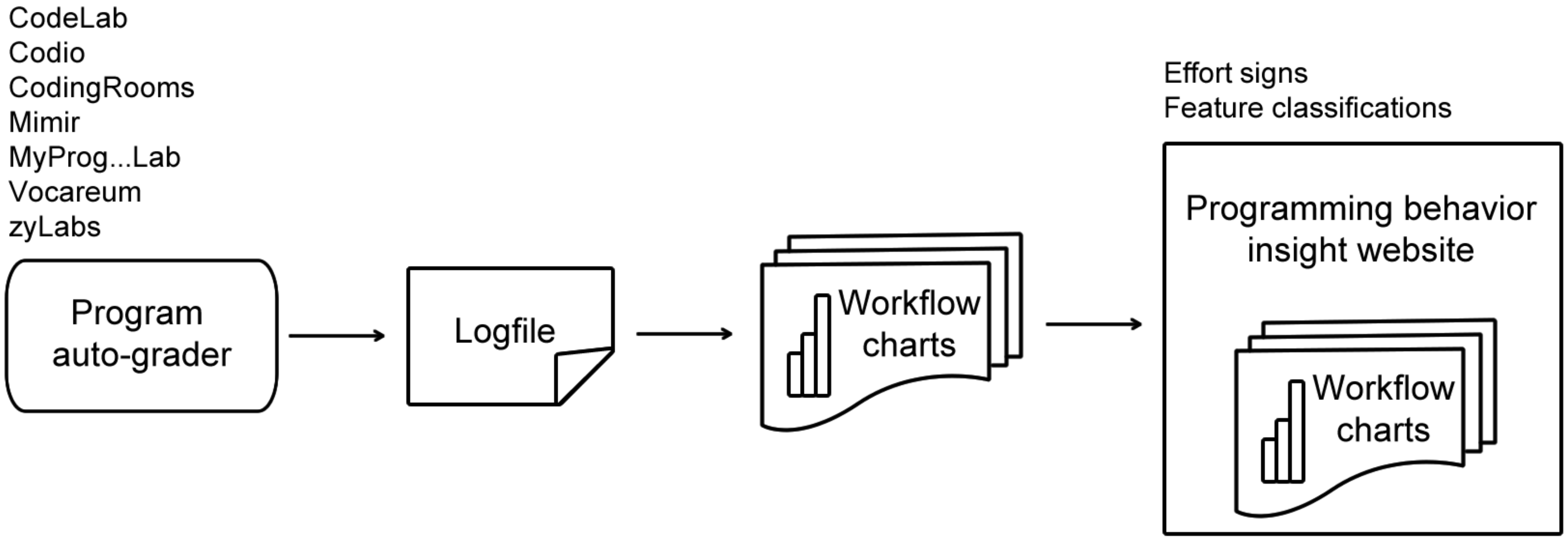
MSP3 (10/10)

Programming workflow charts

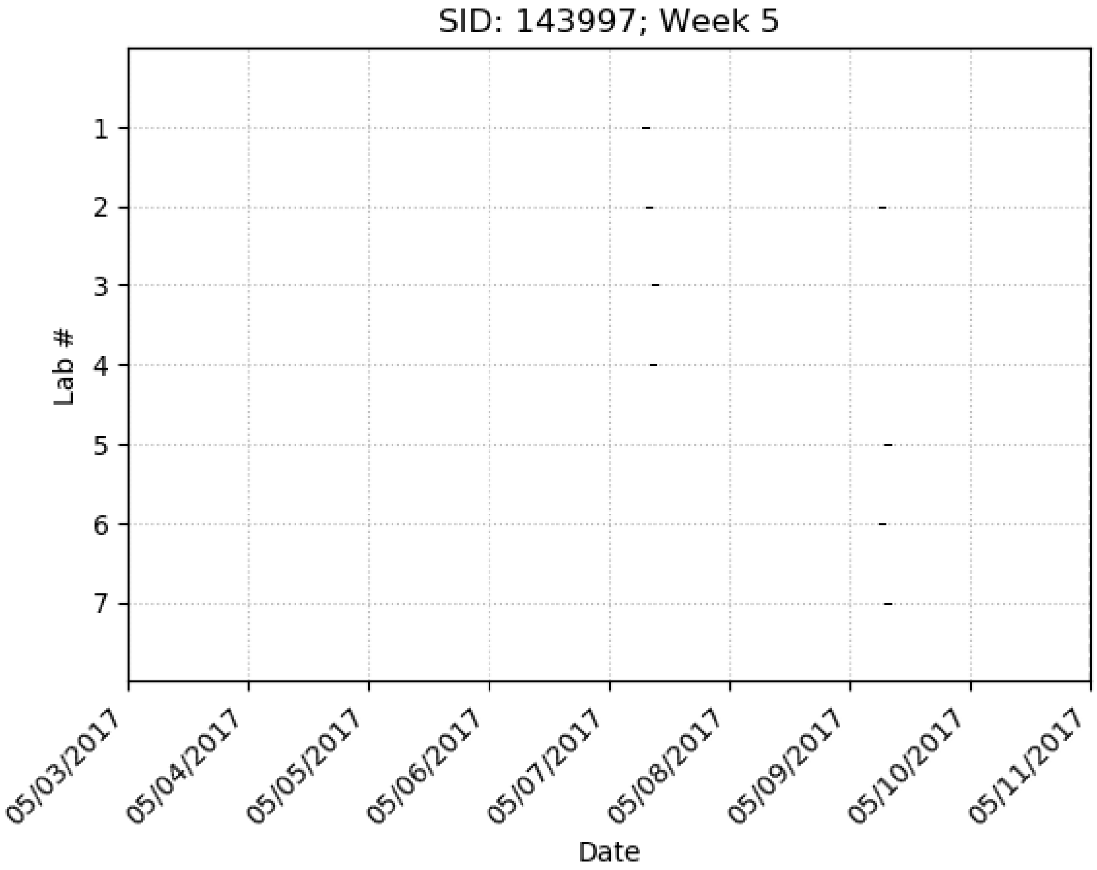
J.M. Allen and F. Vahid. Concise Graphical Representations of Student Effort on Weekly Many Small Programs, ACM SIGCSE Technical Symposium on Computer Science Education, 2021.

Q: Can we visually represent student workflow?

A: Yes - Student workflow charts (GANTT charts)



Version 1: Calendar view (2017)



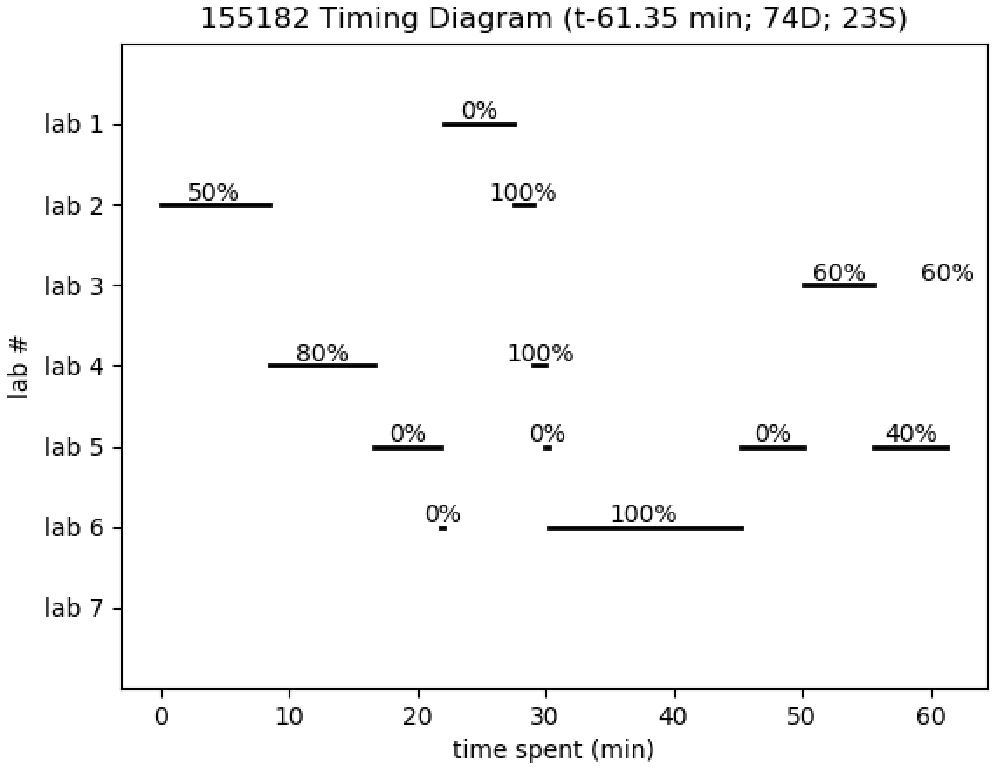
Features

- Weekly calendar view
  - Labs on y-axis, Dates on x-axis
- Horizontal lines to indicate time spent

Tradeoffs

- Pros: weekly view
- Cons: data too small

Version 2: Compressed chart (2018)



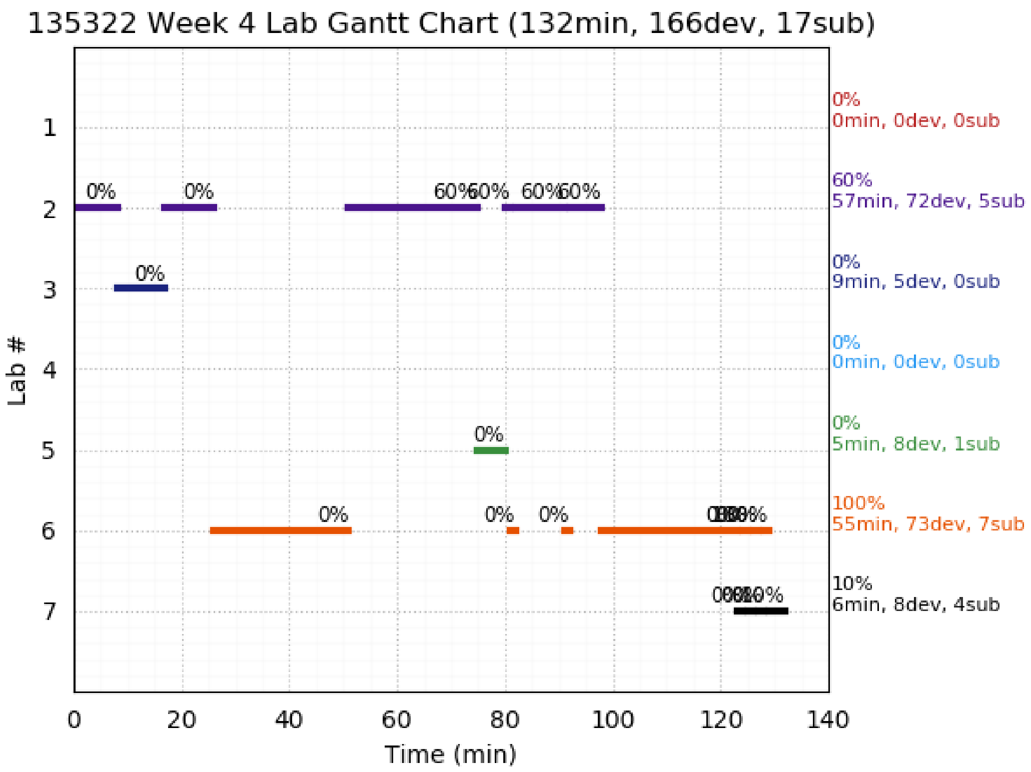
Features

- Total time view
  - Labs on y-axis, Time spent on x-axis
- Horizontal lines to indicate time spent
  - Score earned (%)

Tradeoffs

- Pros: data representation
- Cons: readability

Version 3: Clarity & readability (2018)



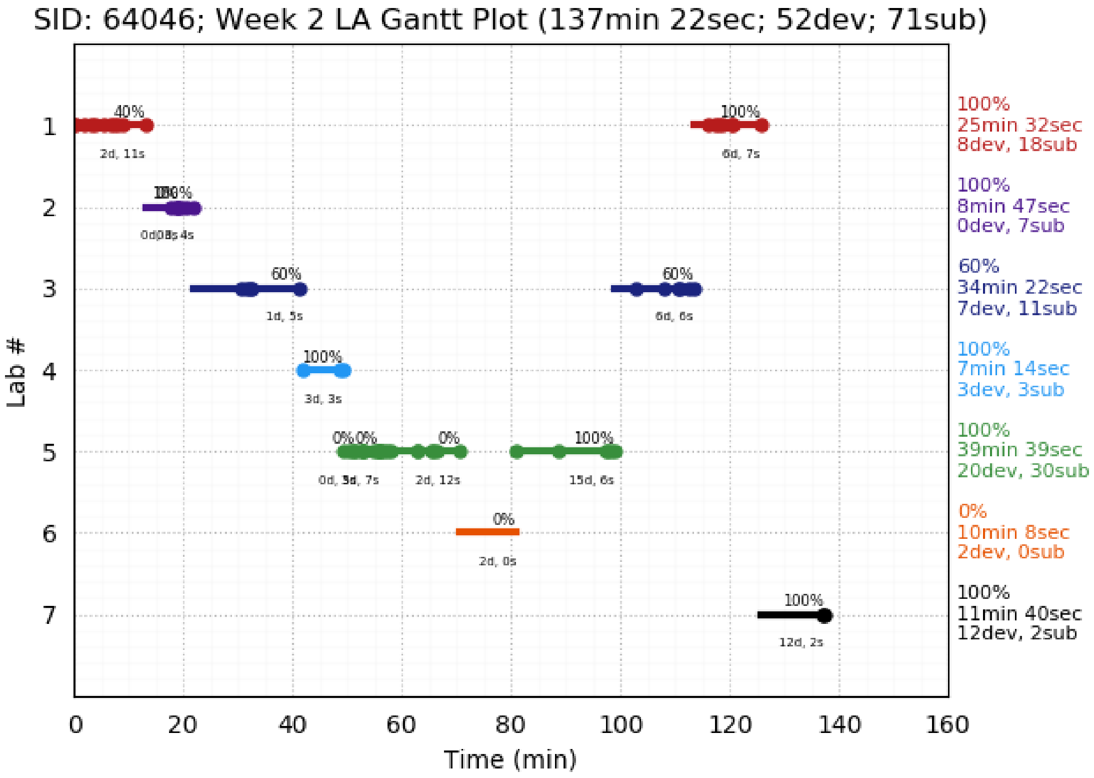
Features

- Colors
- Data summary labels
- Grid
- Updated logic

Tradeoffs

- Pros: readability
- Cons: readability (slight)
- Considerations: line colors & styles

Version 4a: Run type (2019)



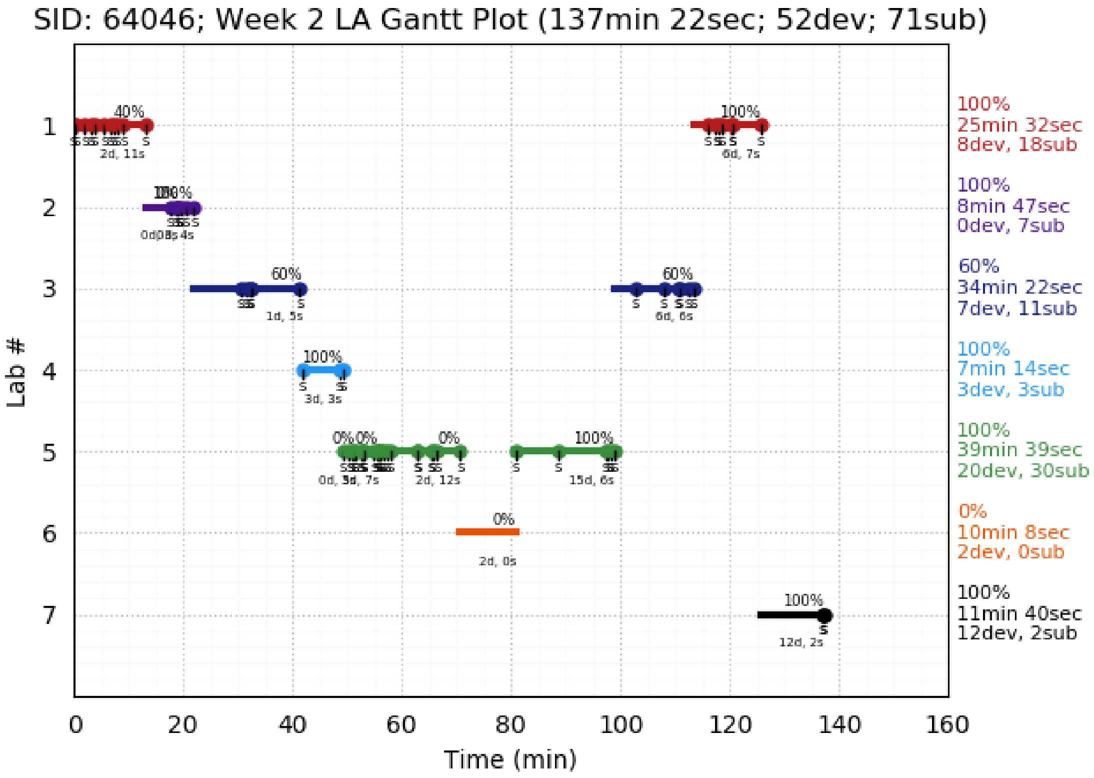
Features

- Develop & submit indicators
  - Text & solid points
- Minor update to labels

Tradeoffs

- Pros: more information
- Cons: clutter, readability, & data representation
- Considerations: indicator shape

Version 4b: Run type details (2019)



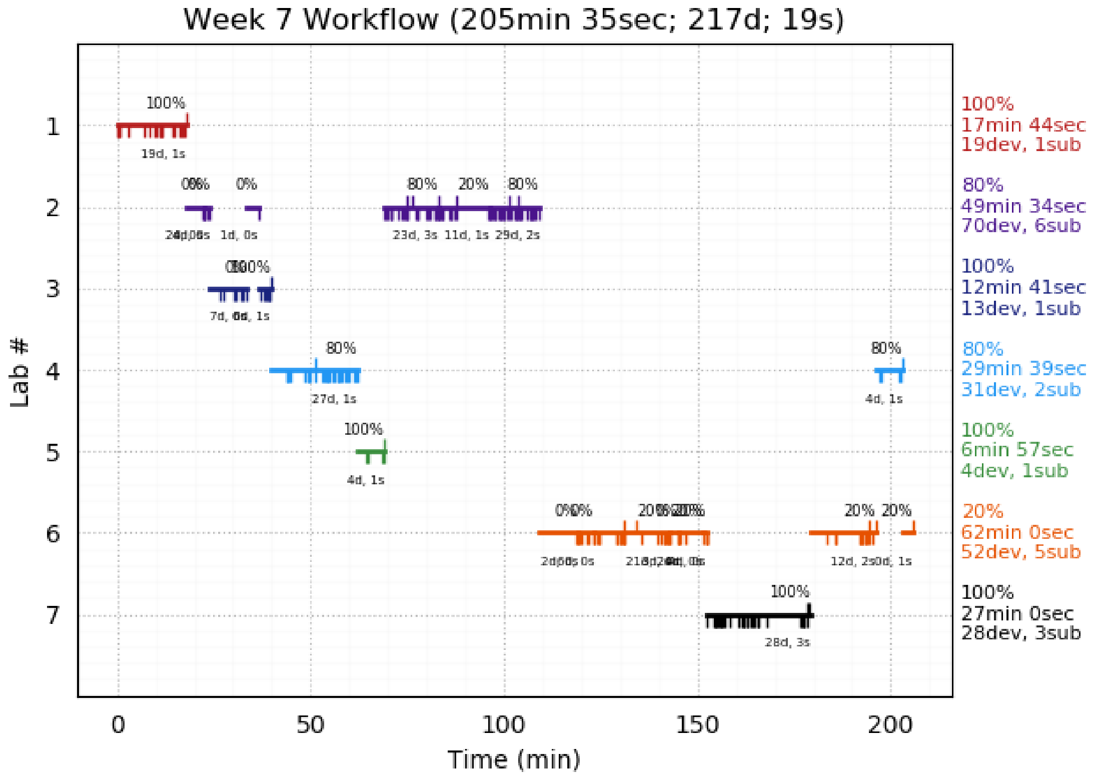
Features

- Develop & submit indicators
  - Text & solid points
  - Character ‘tails’
- Minor update to labels

Tradeoffs

- Pros: more information, data representation
- Cons: extra clutter & readability

Version 5: Tick marks (2020)



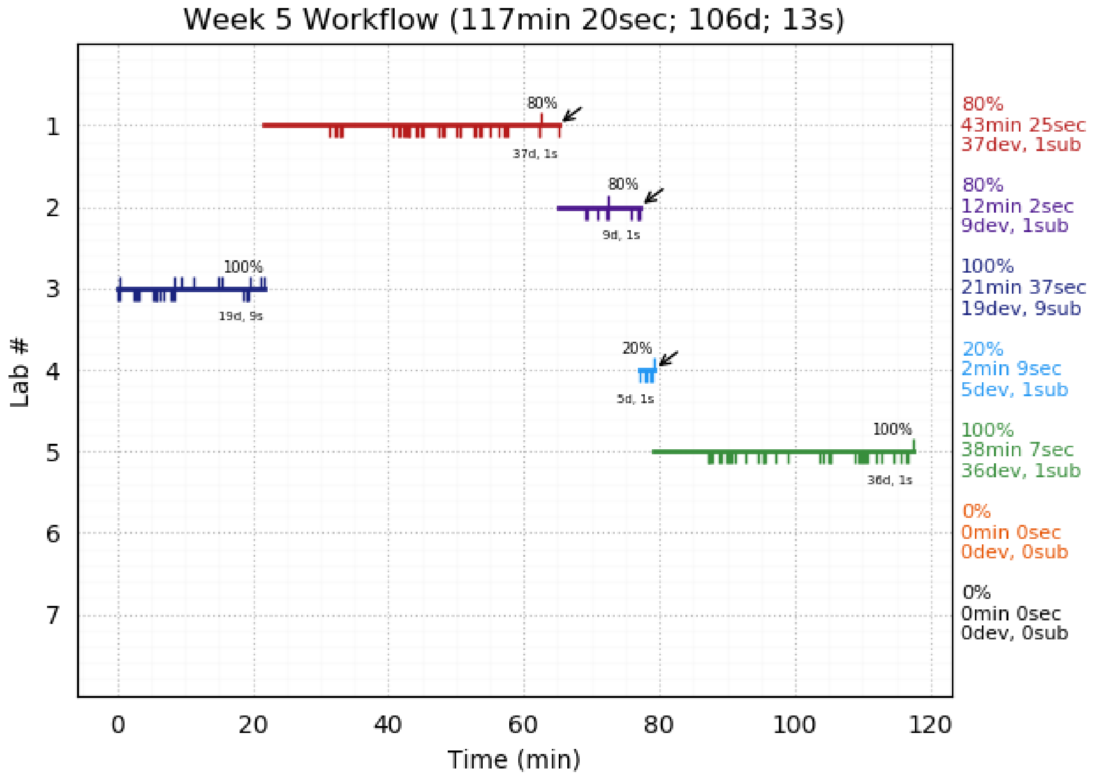
Features

- Develop & submit indicators
  - Tick marks

Tradeoffs

- Pros: more information & readability
- Cons: minor clutter

Version 6: Pivot indicators (2020)

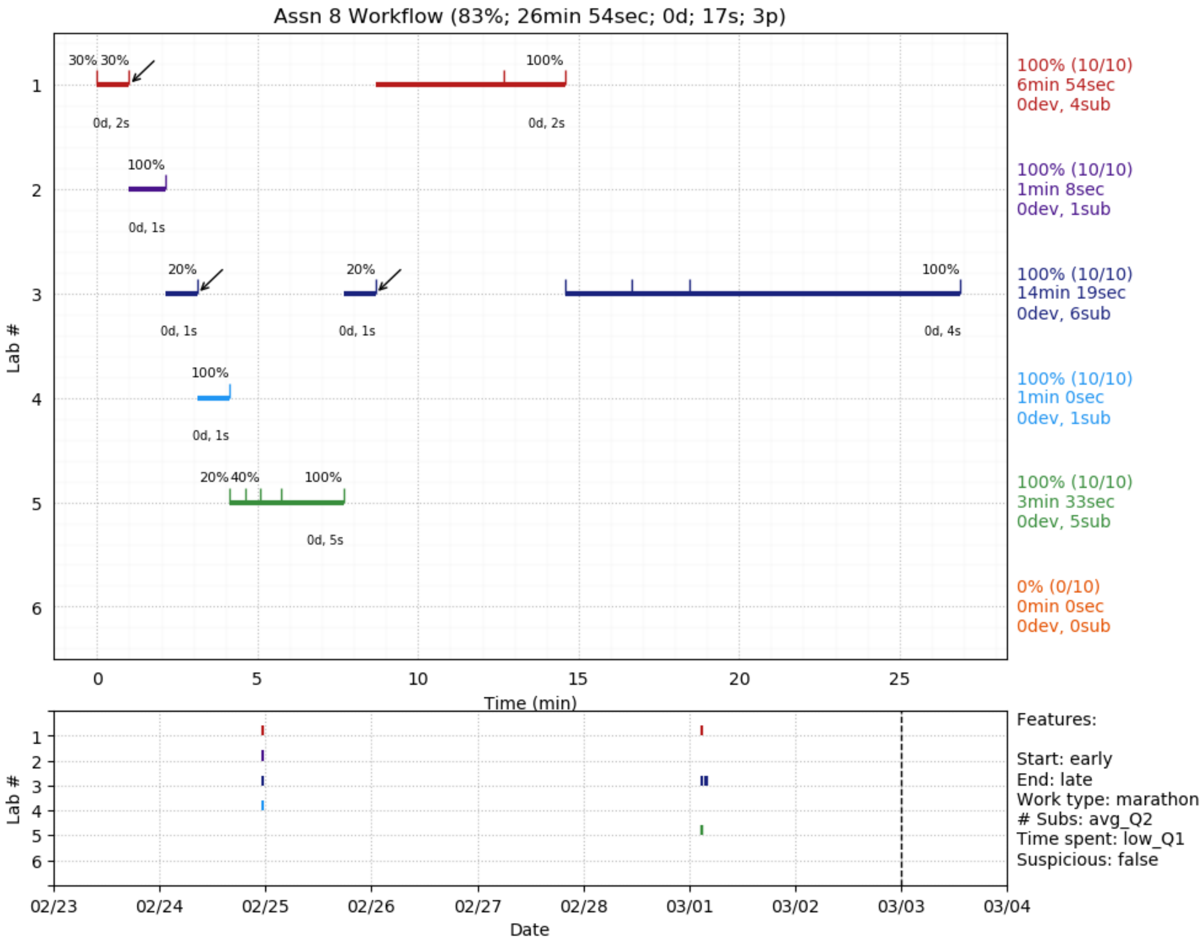


Pivot: A switch between lab activities without completing the current lab activity.

Features

- Pivot indicators
  - Arrow to indicate pivots

Version 7: Dual View (2021)



Features:

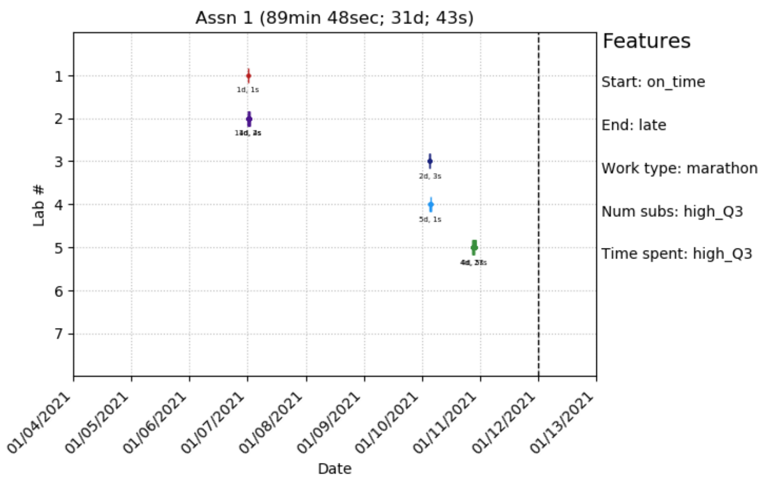
Start: early
End: late
Work type: marathon
Subs: avg_Q2
Time spent: low_Q1
Suspicious: false

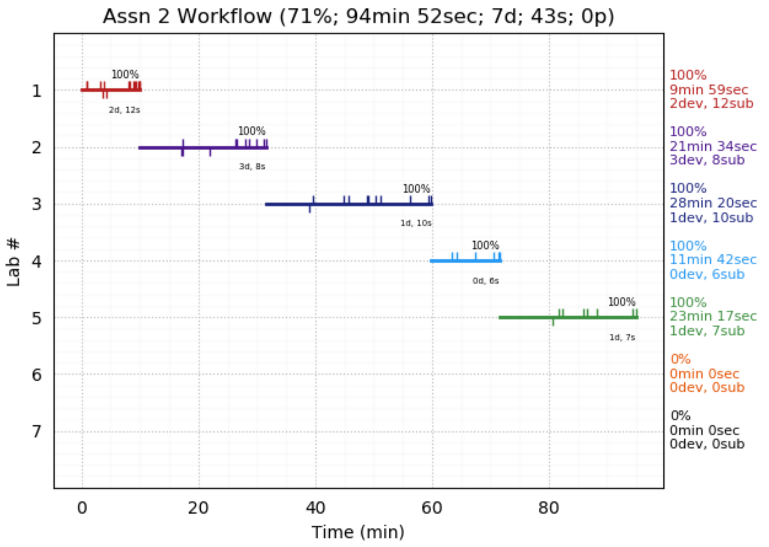
Features

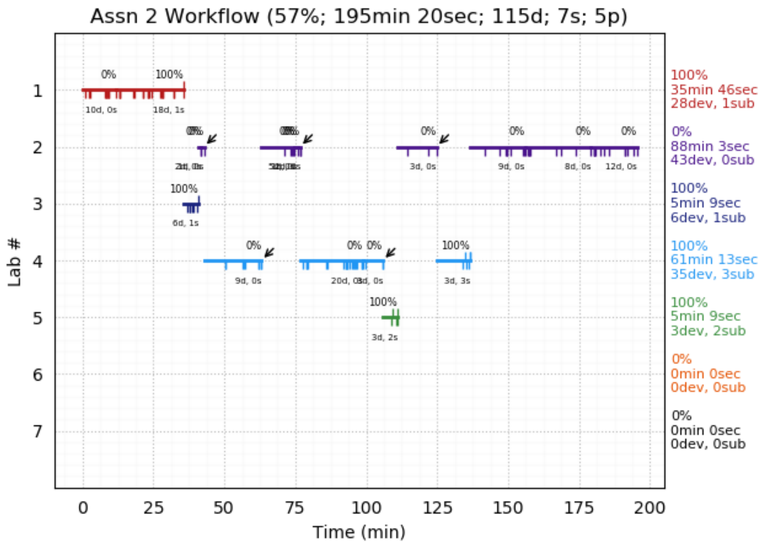
- Dual time view & week view
- Classification features

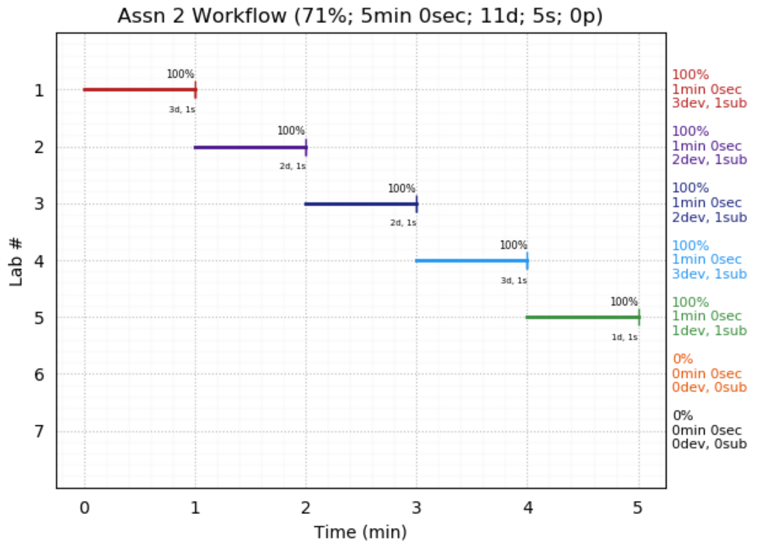
Current uses

- Understanding student effort
  - Normal, struggling, suspicious
- Basic student classifications
- Interactive website

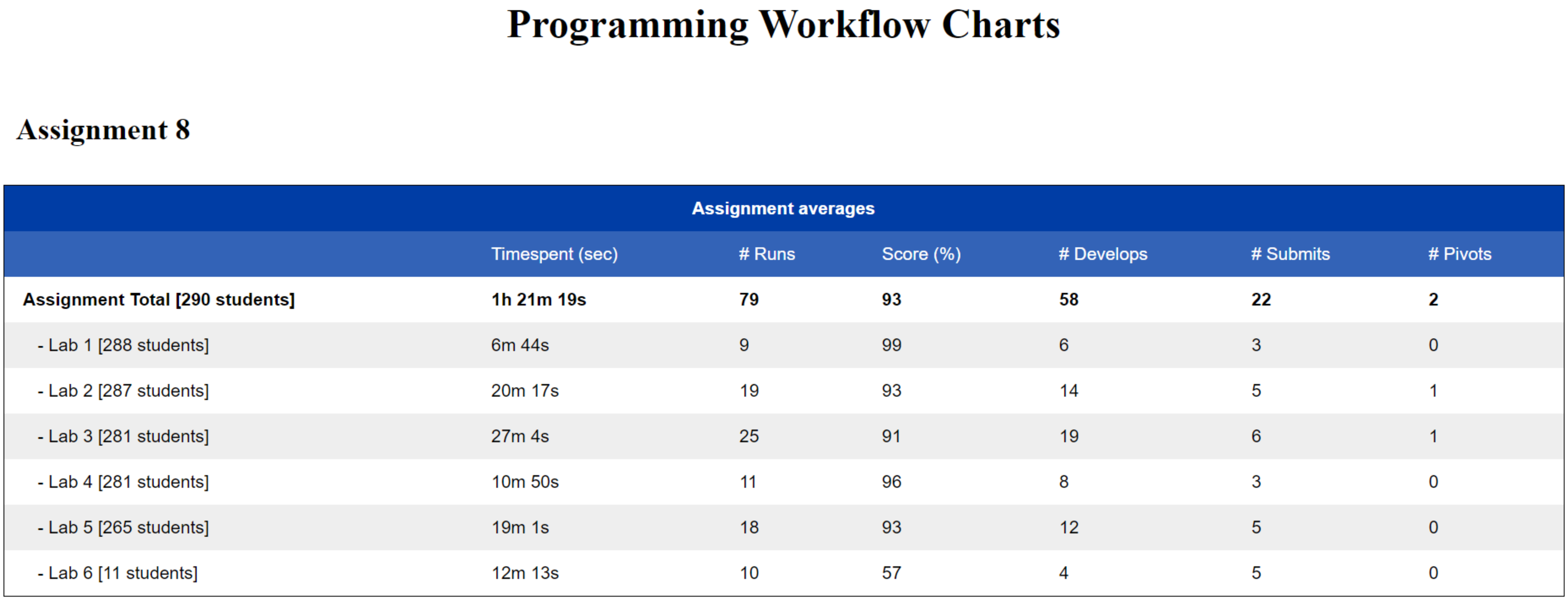








Interactive website

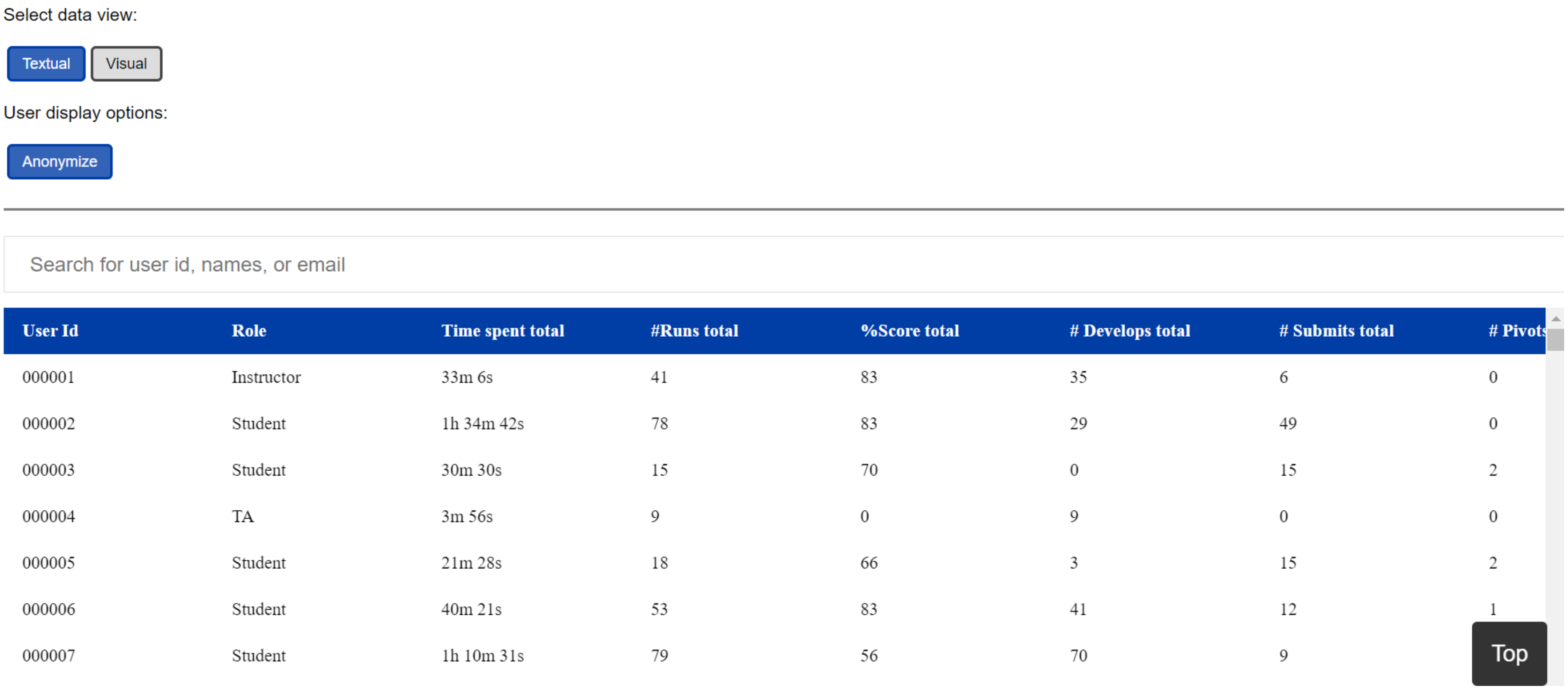


Programming Workflow Charts

Assignment 8

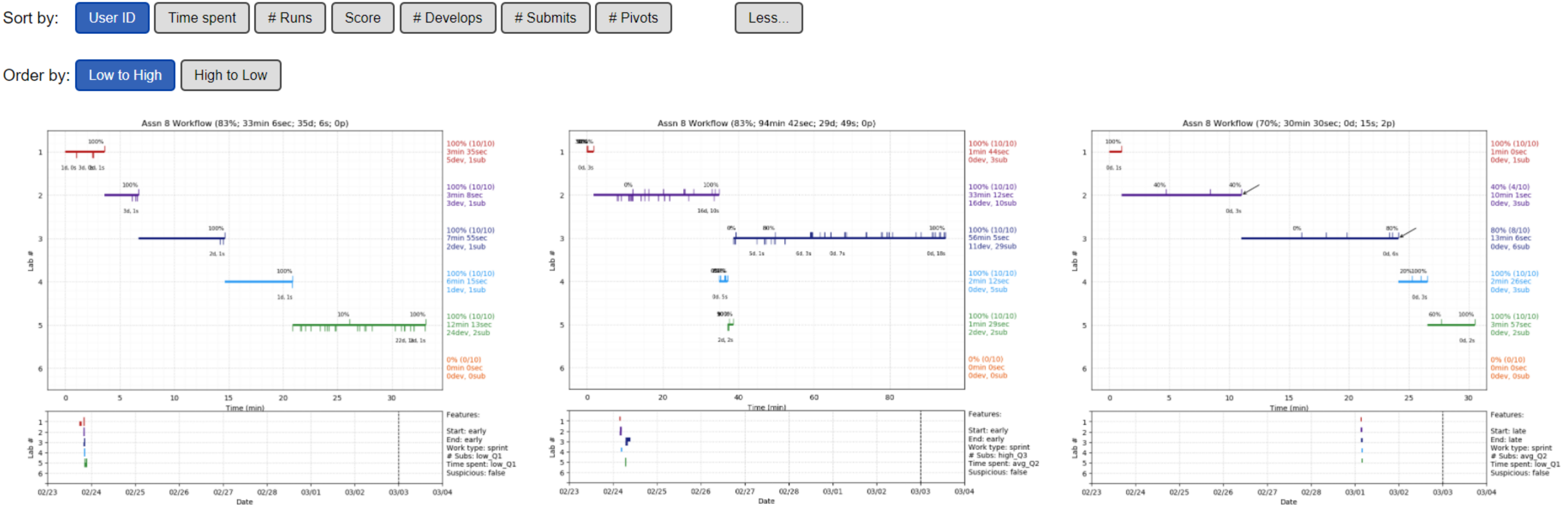
Assignment averages						
	Timespent (sec)	# Runs	Score (%)	# Develops	# Submits	# Pivots
Assignment Total [290 students]	**1h 21m 19s**	**79**	**93**	**58**	**22**	**2**
- Lab 1 [288 students]	6m 44s	9	99	6	3	0
- Lab 2 [287 students]	20m 17s	19	93	14	5	1
- Lab 3 [281 students]	27m 4s	25	91	19	6	1
- Lab 4 [281 students]	10m 50s	11	96	8	3	0
- Lab 5 [265 students]	19m 1s	18	93	12	5	0
- Lab 6 [11 students]	12m 13s	10	57	4	5	0

Interactive website



User Id	Role	Time spent total	#Runs total	%Score total	# Develops total	# Submits total	# Pivots
000001	Instructor	33m 6s	41	83	35	6	0
000002	Student	1h 34m 42s	78	83	29	49	0
000003	Student	30m 30s	15	70	0	15	2
000004	TA	3m 56s	9	0	9	0	0
000005	Student	21m 28s	18	66	3	15	2
000006	Student	40m 21s	53	83	41	12	1
000007	Student	1h 10m 31s	79	56	70	9	

Interactive website



Features:

Start: early
End: early
Work type: sprint
Subs: low_Q1
Time spent: low_Q1
Suspicious: false

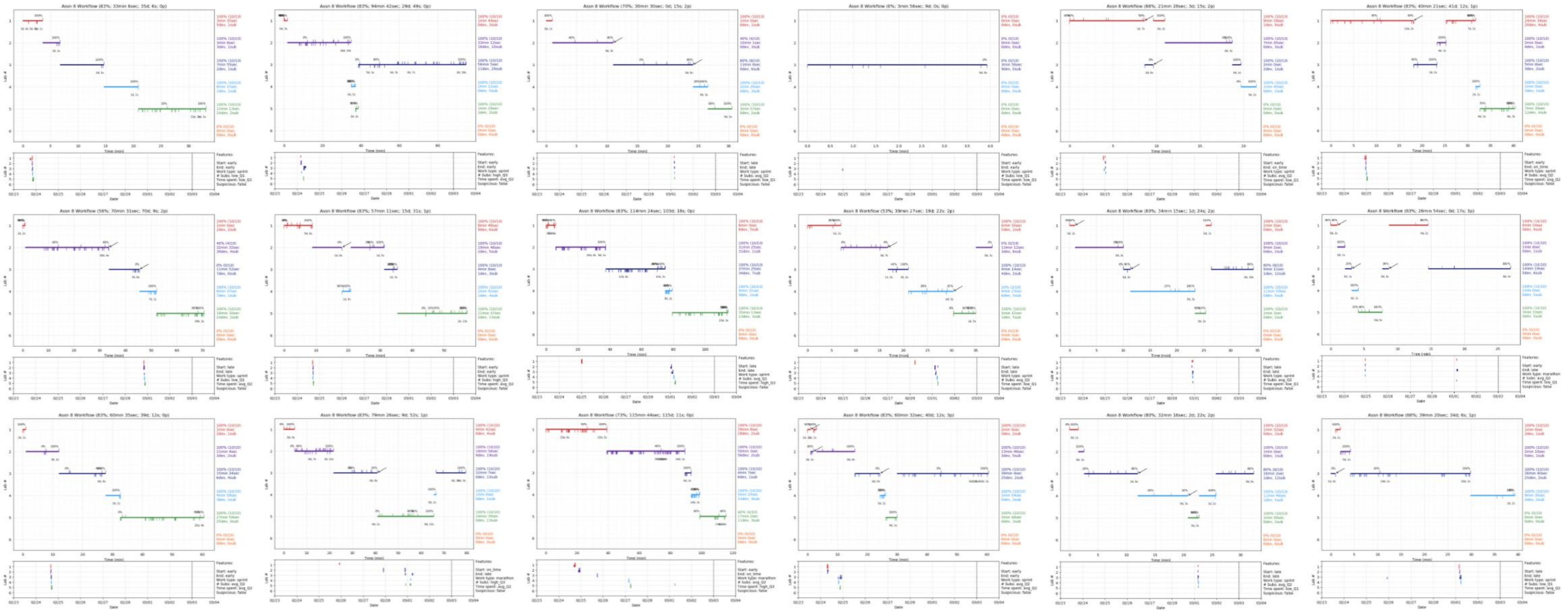
Features:

Start: early
End: early
Work type: sprint
Subs: high_Q3
Time spent: avg_Q2
Suspicious: false

Features:

Start: late
End: late
Work type: sprint
Subs: avg_Q2
Time spent: low_Q1
Suspicious: false

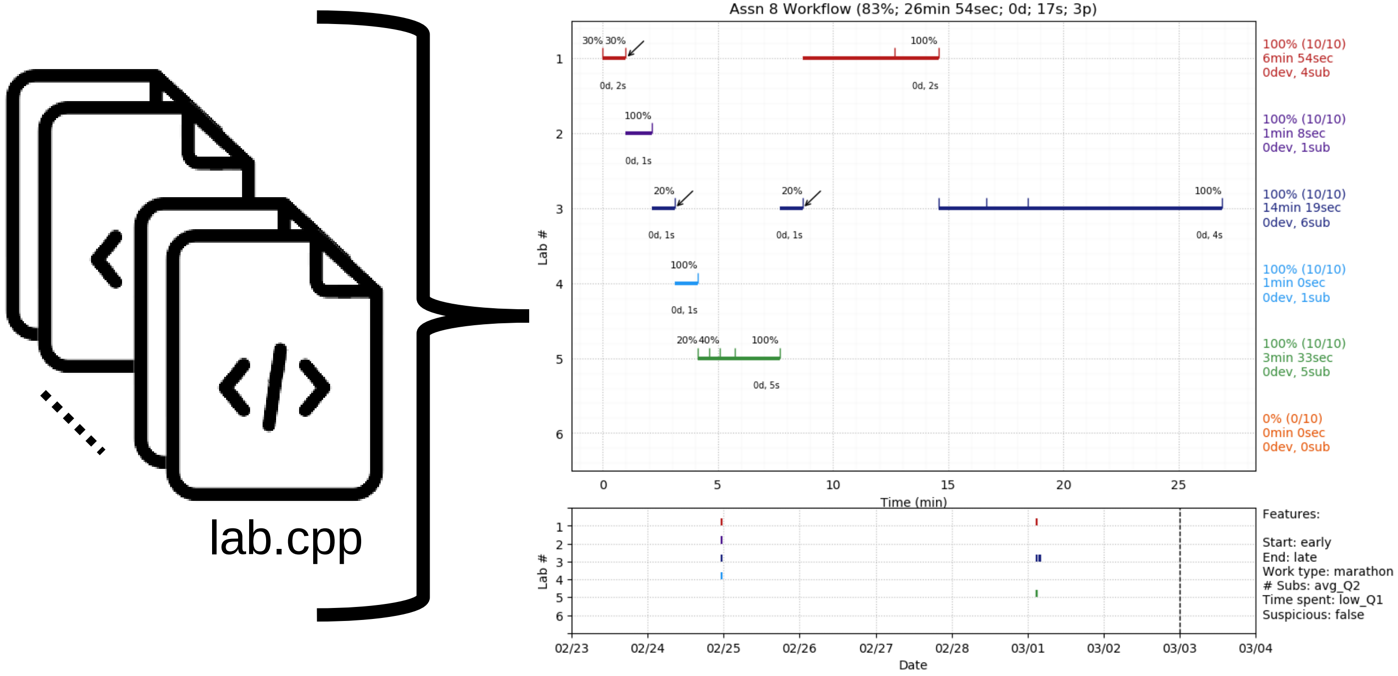
Interactive website



Findings

- Workflow charts are useful in CS1
  - Gain insight on student behavior
  - Recognize typical patterns
  - Show students

Programming Workflow Charts



Contributions

- An MSP approach in CS1 improves student satisfaction and reduces student stress
- Students use an MSP approach to benefit their learning
- An MSP approach can be used across universities and programming languages
- Programming workflow charts provide quick and concise understanding student efforts
- Pivoting reduces students’ frustration
- CS1 DFW rate reduced to 8.4%

Publications

- **Experience with an MSP approach**
  - J.M. Allen, F. Vahid, K. Downey, and A. Edgcomb. Weekly Programs in a CS1 Class: Experiences with Auto-graded Many-small Programs (MSP), Proceedings of ASEE Annual Conference, 2018. (best paper nominee)
  - J.M. Allen, F. Vahid, A. Edgcomb, K. Downey, and K. Miller. An Analysis of Using Many Small Programs in CS1, ACM SIGCSE Technical Symposium on Computer Science Education, 2019.

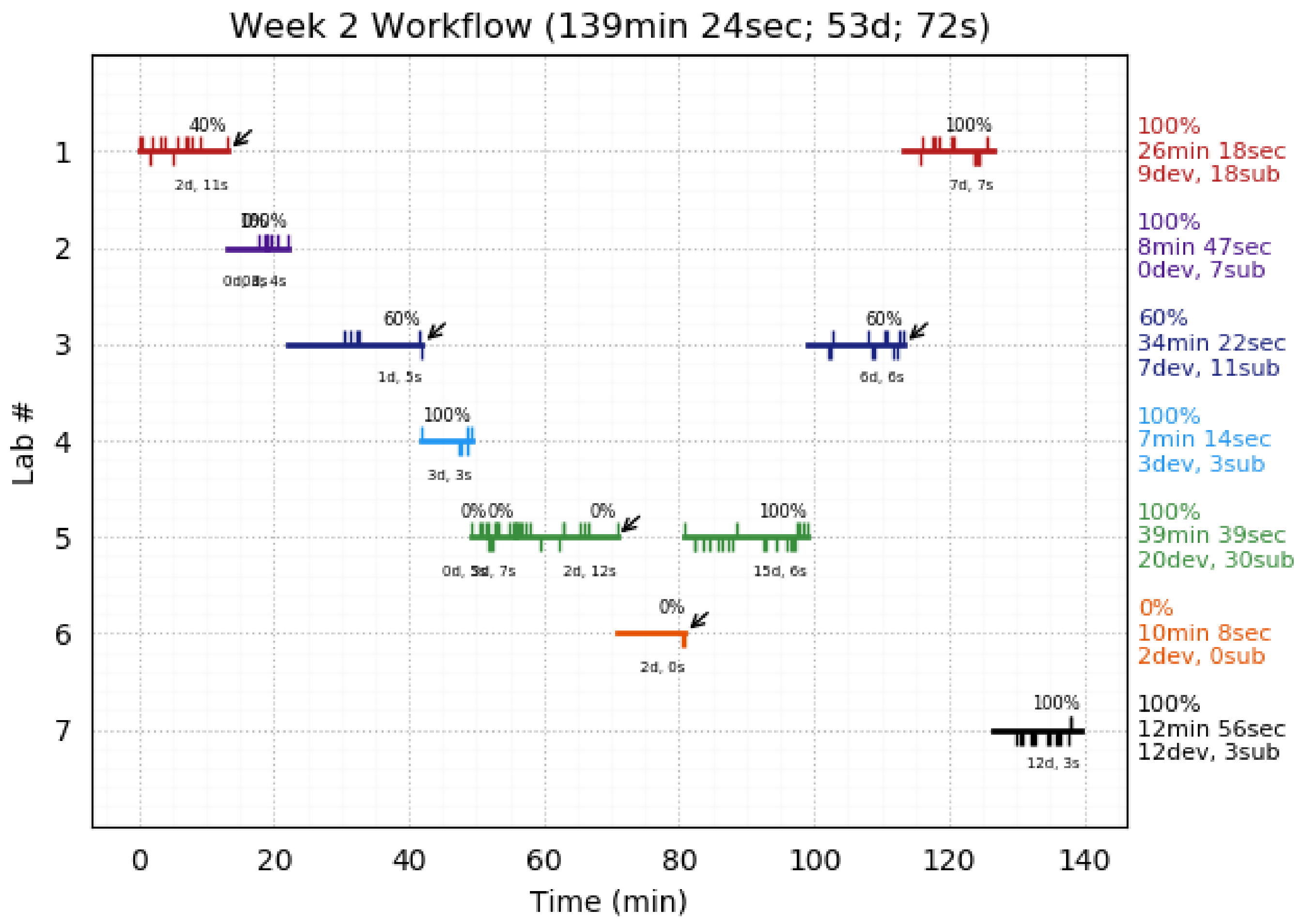
- **Many small programs in CS1: usage analysis from multiple universities**
  - J.M. Allen, F. Vahid, K. Downey, K. Miller, and A. Edgcomb. Many Small Programs in CS1: Usage Analysis from Multiple Universities, Proceedings of ASEE Annual Conference, 2019.

- **Experiences in developing a robust popular online CS1 course for the past 7 years**
  - J.M. Allen and F. Vahid. Experiences in Developing a Robust Popular Online CS1 Course for the Past 7 Years, Proceedings of ASEE Annual Conference, 2020.
  - F. Vahid and J.M. Allen. An online course for freshmen? The evolution of a successful online CS1 course, Proceedings of FYEE Annual Conference, 2020.

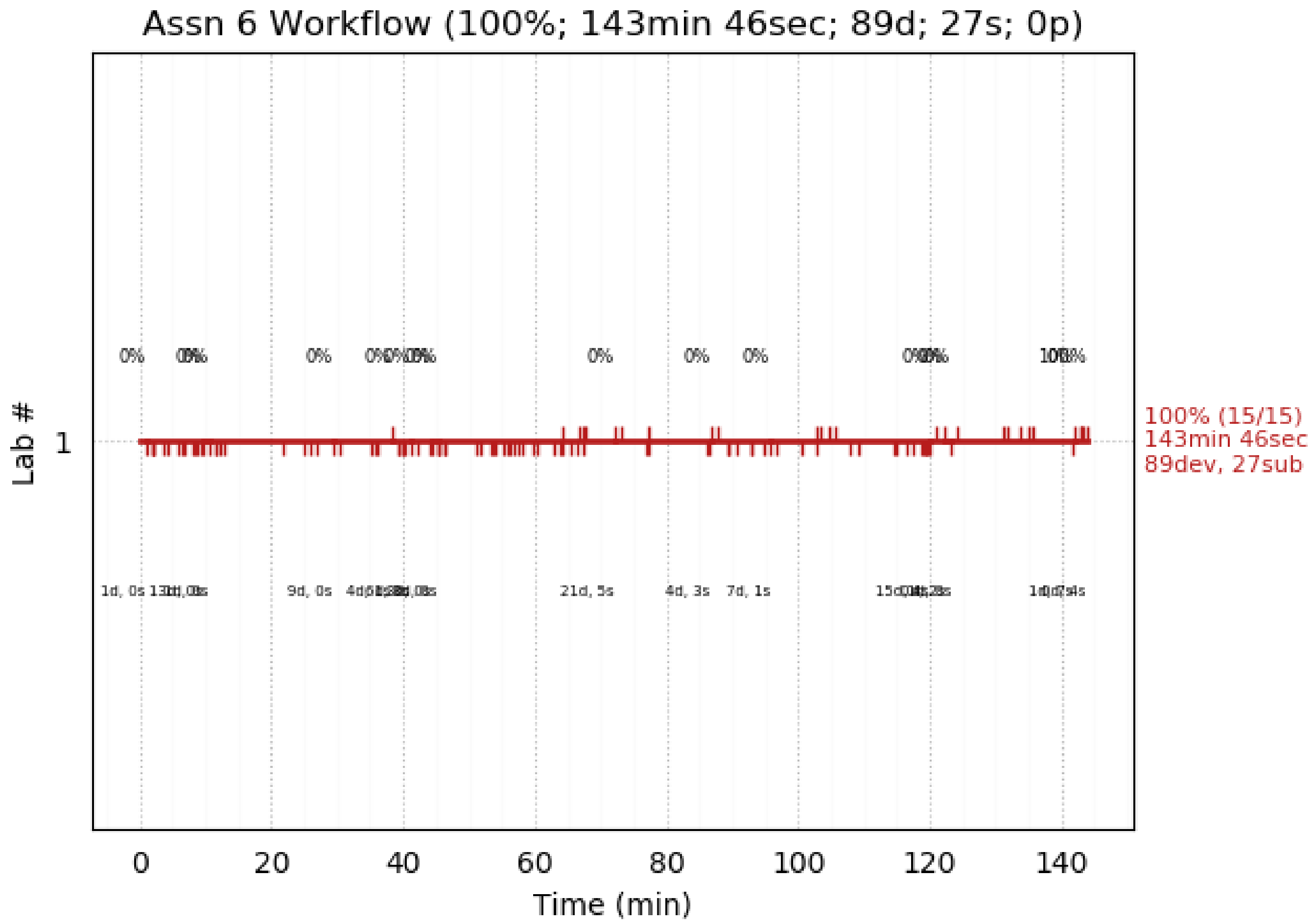
- **Teaching coral before C++ in a CS1 course**
  - J.M. Allen and F. Vahid. Teaching Coral before C++ in a CS1 Course, Proceedings of ASEE Annual Conference, 2020.
  - J.M. Allen and F. Vahid. An Analysis of Using Coral Many Small Programs in CS1, Journal of Computing Sciences in Colleges, 2021.
  - F. Vahid, J.M. Allen, A. D. Edgcomb, and R. Lysecky. Using the free Coral language and simulator to simplify first-year programming courses, Proceedings of FYEE Annual Conference, 2020.

- **Understanding features of an MSP approach**
  - J.M. Allen and F. Vahid. Analyzing Pivoting Among Weekly Many Small Programs in a CS1 Course, Proceedings of ASEE Annual Conference, 2020.
  - J.M. Allen and F. Vahid. Concise Graphical Representations of Student Effort on Weekly Many Small Programs, ACM SIGCSE Technical Symposium on Computer Science Education, 2021.

Appendix A: Workflow chart (MSP)

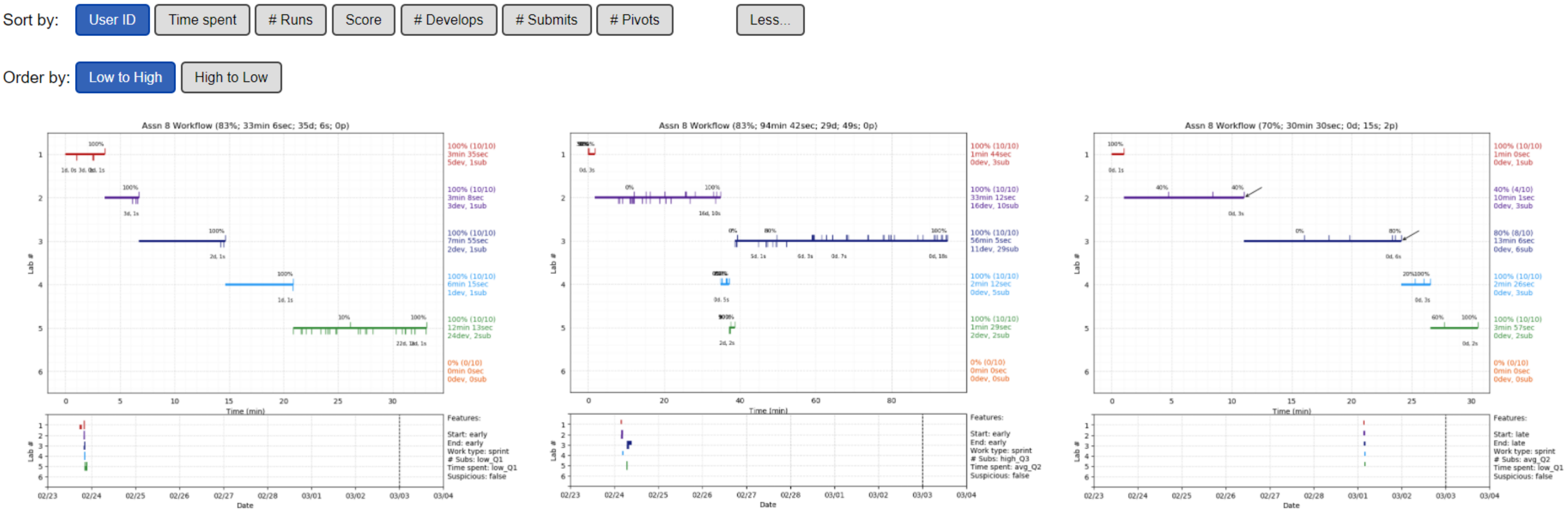


Appendix B: Workflow chart (OLP)



Appendix C: Online webpage

Interactive website


Sort by:
User ID
Time spent
# Runs
Score
# Develops
# Submits
# Pivots
Less...
Order by:
Low to High
High to Low
Assn 8 Workflow (83%; 33min 6sec; 35d; 6s; 0p)
100% (10/10)
3min 35sec
5dev, 1sub
100% (10/10)
3min 8sec
3dev, 1sub
100% (10/10)
7min 55sec
2dev, 1sub
100% (10/10)
6min 15sec
1dev, 1sub
100% (10/10)
12min 13sec
24dev, 2sub
0% (0/10)
0min 0sec
0dev, 0sub
Time (min)
Lab #
Date
Features:
Start: early
End: early
Work type: sprint
# Subs: low_Q1
Time spent: low_Q1
Suspicious: false
Assn 8 Workflow (83%; 94min 42sec; 29d; 49s; 0p)
100% (10/10)
1min 44sec
0dev, 3sub
100% (10/10)
33min 12sec
16dev, 10sub
100% (10/10)
56min 5sec
11dev, 29sub
100% (10/10)
2min 12sec
0dev, 5sub
100% (10/10)
1min 29sec
2dev, 2sub
0% (0/10)
0min 0sec
0dev, 0sub
Features:
Start: early
End: early
Work type: sprint
# Subs: high_Q3
Time spent: avg_Q2
Suspicious: false
Assn 8 Workflow (70%; 30min 30sec; 0d; 15s; 2p)
100% (10/10)
1min 0sec
0dev, 1sub
40% (4/10)
10min 1sec
0dev, 3sub
80% (8/10)
13min 6sec
0dev, 6sub
100% (10/10)
2min 26sec
0dev, 3sub
100% (10/10)
3min 57sec
0dev, 2sub
0% (0/10)
0min 0sec
0dev, 0sub
Features:
Start: late
End: late
Work type: sprint
# Subs: avg_Q2
Time spent: low_Q1
Suspicious: false
02/23
02/24
02/25
02/26
02/27
02/28
03/01
03/02
03/03
03/04