Concise Graphical Representations of Student Effort on Weekly Many Small Programs

Joe Michael Allen[1], Frank Vahid[1,2]
[1]Computer Science and Engineering, University of California, Riverside
[2]zyBooks, Los Gatos, California
jalle010@ucr.edu, vahid@cs.ucr.edu

ABSTRACT

In recent years, hundreds of CS1 classes have adopted a many small programs (MSP) approach to weekly programming assignments. The MSP approach involves assigning students several smaller programming assignments per week, for example 5-7, versus the traditional one larger program (OLP) per week. This shift is largely made possible by easy-to-use program auto-graders that have arisen in recent years. Such auto-graders make grading so many programs feasible, while also providing students with immediate feedback. The MSP approach has been shown to yield advantages that include earlier starts, reduced anxiety, increased confidence, the ability to switch to another program if stuck, better exam performance, and less attrition, with analysis showing students easily transition to larger programs later. We desired to gain insight on how our CS1 students were working through our weekly MSPs. Thus, in 2018, we began exploring automated creation of concise representations of student behavior while they developed their programs, what we call "workflow charts". We used a popular commercial auto-grader that has a built-in development environment and provides detailed log files of every program compile/run by each student. We describe the goals of such a representation, the evolution of our representation to its current status, various design trade-offs, our current usage, and numerous possible future uses in CS1 classes. We plan to create a website for any instructor to upload such log files to gain insight on their own class' performance.

CCS CONCEPTS

•Social and professional topics~Professional topics~Computing education~Student assessment•Social and professional topics~Professional topics~Computing education~Computing education programs~Computer science education~CS1•Social and professional topics~Professional topics~Computing education~Computing education programs~Software engineering education

KEYWORDS

Student effort, Many small programs, MSPs, Graphical representation, CS1

ACM Reference format:
Joe Michael Allen and Frank Vahid. 2021. Concise Graphical Representations of Student Effort on Weekly Many Small Programs. *In Proceedings of the 52nd ACM Technical Symposium on Computer Science Education (SIGCSE '21), March 13--20, 2021, Virtual Event, USA.* ACM, New York, NY, 6 pages. DOI: https://doi.org/10.1145/3408877.3432551

SIGCSE '21, March 13–20, 2021, Virtual Event, USA.

ACM ISBN 978-1-4503-8062-1/21/03. https://doi.org/10.1145/3408877.3432551

1 Introduction

Beyond seeing final submissions, many instructors want insight into how students went about the process of writing their code -- when did they start, how often did they test, how correct was their code along the way, how much time did they spend overall, etc. As such, some now require students to use version control software like github, to at least see some versions of the code during development. However, program auto-graders provide a distinct opportunity for such insight, having grown tremendously in use in recent years, including new commercial tools like zyBooks [1], Gradescope [2], Mimir [3], Vocareum [4], CodeLab [5], and MyProgrammingLab [6]. Some of those also have development environments so that all a student's programming activity can be recorded: "develop" runs while the student is still developing and testing their code, and "submit" runs where they submit code for auto-grading. Non-commercial systems also record develop runs and/or submit runs, like Runestone [7] and BlueJ [8]. Such recording opens new possibilities for instructors to gain the desired insight in student coding.

Meanwhile, hundreds of schools, including ours, have converted to a "many small programs" approach (zyBooks alone reports over 200 schools; many more exist). Thus, not only do we want insight into our students' programming process, but we want that for 7 programs per week, to see which they started on, how they switched between programs, and so on. A table of statistics is too hard for an instructor to process and loses too much information. Thus, in 2018, we began developing a script to process the log files from the popular auto-grader that we use and convert to a graphical representation that we call "workflow charts". We have found those charts provide instructors with tremendous insight, allowing a quick determination of how a class is doing (starting on time? spending sufficient time?), but also to quickly see a

particular student's effort (such as when a student comes to office hours for help, or is requesting an extension) -- and even to detect some cheating cases. We even pull up the charts for the class and use them as a springboard to dive into a particular student's code (if they offer). Students find the workflow charts "cool", and we believe such charts, if used properly in a class, may even reduce some cheating in the future due to showing students that instructors can see their effort.

2 Background

2.1 Many small programs (MSP) approach

In 2018, we switched our CS1 course to use a many small programs (MSP) approach. An MSP approach involves assigning students multiple small programming assignments, for example 5-7, each week instead of only one large programming assignment (OLP) each week. For our CS1, we chose to assign students 7 programming assignments worth 10 points each, and only required them to earn 70% of total points for full credit. We switched after seeing research [9, 10] that an MSP approach yields benefits such as earlier starts, reduced anxiety, increased confidence, the ability to switch to another program if stuck, better exam performance, and less attrition, with students shown to easily transition to larger programs later.

2.2 Program auto-grader

We used an interactive learning system (textbook + auto-graded homework) with a built-in program auto-grader by zyBooks. The auto-grader is easy to use; it takes us an average of 30 minutes to create a new programming assignment, with no special training. The auto-grader provides students immediate score feedback. The auto-grader has an optional built-in development environment (IDE), and we require students to use that IDE for all MSP programming.

2.3 Data collection

To collect the data required to generate our workflow charts, we obtained from zyBooks log files for all lab activities. The file was in csv format and contained all develop and submit runs for every lab activity in our class. A develop run is when a student tests their code in the built-in IDE without receiving a grade. A submit run is when the student submits their code for grading. Each student activity entry contains metadata such as the title of the lab activity, the user ID, a timestamp, a link to the source code for that run, and for submit runs also contains a score, a max score, the list of test cases including which were passed or failed.

2.4 Time spent calculations

An integral calculation for all workflow charts is the time spent by students on each programming assignment. To calculate time spent, we gathered all student activity and calculated the difference between timestamps. Each difference was then summed together to yield a final calculation of the total time spent. Note, that if the difference between two timestamps exceeded 10-minutes, we excluded the time from our calculations to be conservative as the student likely took a break or went to work on something else. Furthermore, we cannot capture the time a student spent working before their first activity. As such, our data is likely an understatement.

3 The evolution of our workflow charts

Our motivation for creating these representations was to understand how students were interacting with the MSPs. Based on end-of-the-quarter grades, we had seen that students were earning good grades and doing well on exams, but we lacked insight on questions like: How much time are students working on MSPs each week? What days did they work? Were students doing MSPs in the order we listed them, or were they jumping among them? How often were they doing develop runs versus submit runs?

We decided to pursue a graphical representation of the data, to gain quick and concise insights into student effort on weekly MSPs. We used a Gantt chart as the initial motivation behind developing our workflow charts. A Gantt chart is a visual view of tasks scheduled over time [11]. Such a chart highlights important information like the start of a task, the end of a task, and the time spent per task in a single view.

Note that some figures in Section 3 that show the evolution of our charts may differ in example as we do not have records of all previously used iterations.

3.1 Version 1 -- Calendar view

Figure 1 shows Version 1 of our workflow chart. Our initial thought was to display the data using a weekly calendar view to see data on all weekly lab activities for each student each week.

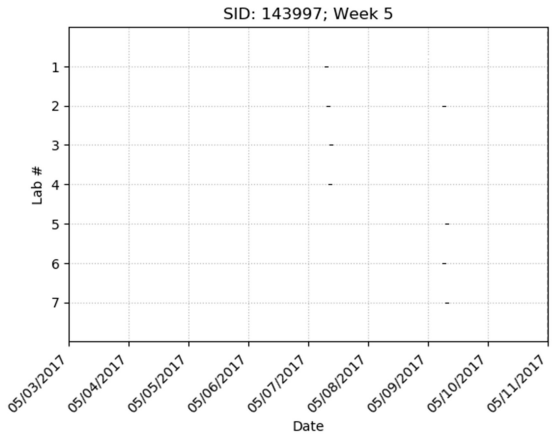

Figure 1: Version 1 of the workflow chart. An expanded calendar view with lab activities on the y-axis and days on the x-axis. Horizontal lines added to indicate when students worked.

As we were using the MSP approach, we had assigned students 7 lab activities per week. On the workflow chart, the lab activities

are listed on the y-axis in ascending order and dates for the week are listed on the x-axis in ascending order. Horizontal lines are added to indicate the times students spent working on each lab activity. Each chart has a title with the student ID (anonymized) and the week the chart was generated for.

Unfortunately, upon initial inspection, the data is very hard to read and at a quick glance, it may even seem like the student did no work for the given week. In actuality, the data is present, but since the chart covers 7-days, the time increments in which the student worked are so small in comparison that they are almost not even visible on the chart. Using the calendar view did not work as we intended, and we needed a better way to represent the data in a compressed way.

3.2 Version 2 -- Compressed chart

For Version 2, we needed a better way to represent the data for a given week. We decided to compress the chart by now considering total time spent during the week instead of spreading out the data across the entire week as in Version 1. We still displayed the lab activities on the y-axis, but switched to showing total time spent on the x-axis. Horizontal lines were still used to indicate time spent per each lab activity. Additionally, we put a percentage above each horizontal line to indicate the highest score a student earned after that session of working on that lab. Each chart has a title that summarizes data for the week, including the student's ID, the total time spent working on lab activities for the week, and the total number of develop runs (D) and submit runs (S). Each chart is read from left to right and from top to bottom. Figure 2 shows Version 2 of the workflow chart.

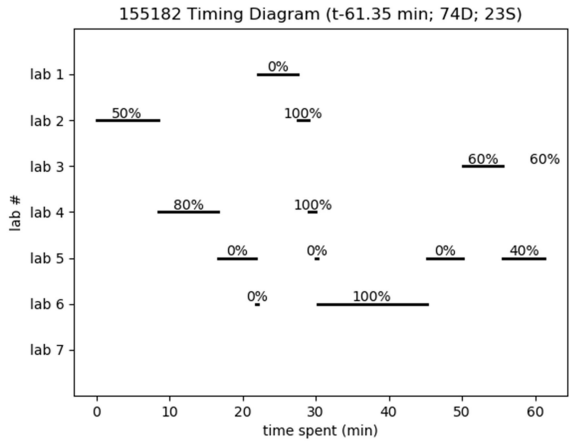

Figure 2: Version 2 of the workflow chart. Compressed chart only considering total time spent represented by a black horizontal line per lab activity and a completion score above.

Version 2 of the workflow chart provided insight into a students' workflow (how they worked on each lab activity during the week), but we soon found ways to get more information onto the chart while maintaining readability.

3.3 Version 3 -- Color / Score per submit run / Statistics per lab

Version 3 of the workflow chart improved clarity and readability. We added color to distinguish data for each lab activity, so when looking at charts for multiple students, an instructor could get a quick sense of which lab took most time -- if seeing a lot of orange, an instructor might know that lab 6 was the most time consuming. Next, we added labels on the right of the chart to summarize data for each lab activity, including the lab's final score, the time spent, and the total numbers of develop and submit runs. We added a grid to enable more accurate readings. Finally, we made a change to the way we considered student work sessions throughout the week. This change is represented in the chart by some horizontal lines having multiple final score percentages listed above them. This will be explained later. The title of each chart was changed for improved readability. Figure 3 shows Version 3 of the workflow charts.

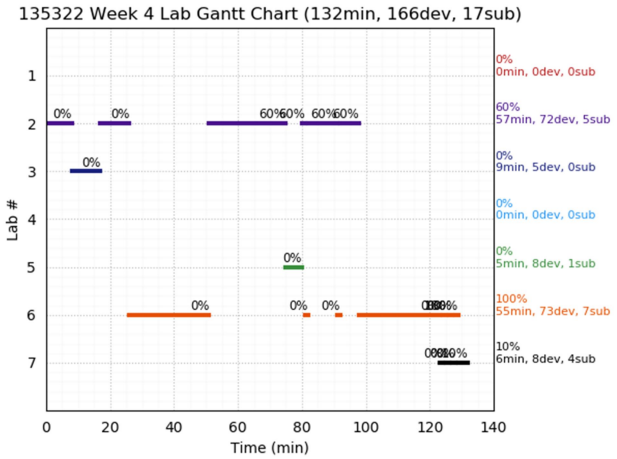

Figure 3: Version 3 of the workflow chart, adding color, summary statistics on the right, gridlines, and more submit scores.

Version 3 of the workflow chart required many design considerations. First, when thinking about how to clearly denote which data corresponded to each specific lab activity, we thought of using color, line styles, or a combination of both. Different line styles proved to yield a cluttered appearance, and some were hard to distinguish. They also didn't enable easily seeing the most/least time-consuming labs across multiple students. A tradeoff here relates to some people potentially having less ability to distinguish color, and loss of info when printed in black and white. A second design consideration was related to the grid. Adding the grid added more clarity to the chart, but in earlier iterations, the grid also decreased data visibility. We initially set the grid color to be too dark and also with a higher volume of tick marks that were unnecessary. After testing different color shades and tick mark frequencies, we chose a lighter color for the grid and reduced the tick marks to achieve the accuracy we wanted.

Finally, we changed the way we thought about how to represent students working on each lab activity, referred to as student work sessions. At first, we considered a student work session to end when the student began working on a different lab activity i.e. submitted code for lab 1 and then developed code for lab 2. Upon deeper analysis, we recognized a scenario where students would begin working on a lab activity, leave to take a break, and then return to work on the same lab activity. We consider this scenario important to denote, so we considered a work session to also end if the time between two activities was more than a 10-minute threshold. We thus showed the score at the end of every session, which is why the figure above shows multiple 60% values on a single bar of lab 2, for example. This distinction does lead to some clutter if the student has many work sessions back-to-back (as can be seen in Figure 3 for lab 7), but we felt the distinction helped instructors to better understand student workflow patterns.

3.4 Version 4 -- More develop/submit details

Version 3 provided the foundation for all the following updates of our workflow chart. As we used these charts for analysis in our teaching each quarter, we noticed a lack of insight on student behavior during each work session. Version 3 summarized data for each lab activity at the end of the week, but not during the week. As such, in Version 4, we wanted our chart to add further insight into student develop and submit runs during work sessions. To accomplish this, we added indicators on the time spent data lines for when a submission took place. These are indicated in a few different styles as seen in Figure 4 and Figure 5.

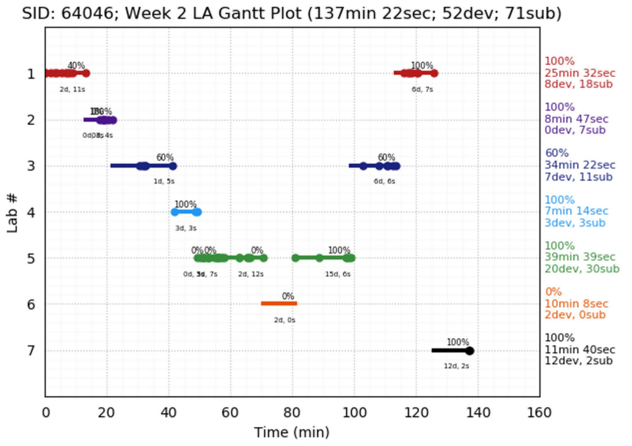

Figure 4: Version 4a. Used large filled in points to indicate a submit run, added text to summarize student activity per work session, minor adjustments to chart labels.

We also added text data on the number of develop and submit runs during each work session underneath each time spent line. Finally, we made minor adjustments to the labels on the right of the chart such that each feature was on it's own line for additional clarity.

Version 4a shown in Figure 4 uses large filled in points to indicate submit runs. Using this indication style made it easy to see submit runs, but added clutter due to the size of the points. Also, this indication did not show develop runs.

Another approach we took, seen in Figure 5 Version 4b, uses a small point with a tail and a character label listed below to denote a develop or submit run. A develop run is indicated with the 'D' character and a submit run is indicated by the 'S' character. By reducing the size of the point and adding a character, the clutter was lessened and the distinction was clear. Unfortunately, with the additional markings, it became difficult to visually separate an 'S' from a 'D.'

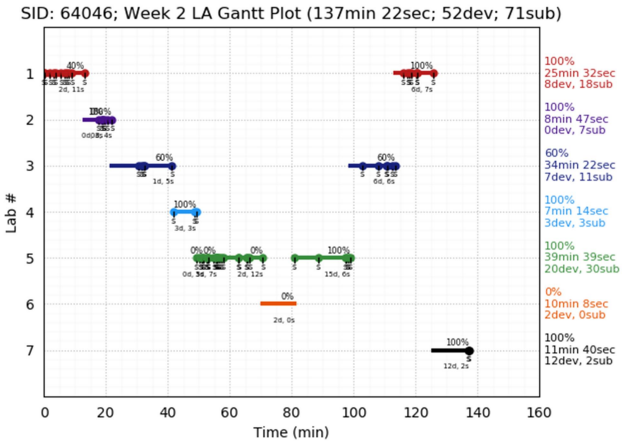

Figure 5: Version 4b. Used small points with a 'S' label to indicate a submit run and a 'D' label to indicate a develop run. Other updates are similar to Figure 4.

We also experimented using other shapes as indicators like squares, diamonds, stars, 'X's,' and open points, but none worked out. In both versions, the text indications for total develop and submit runs below the data lines were helpful. There were some situations where this data would overlap, making some content difficult to read, but this didn't happen very often.

3.5 Version 5 -- Tick marks for develop runs and submit runs

In Version 5, we solved how to effectively display develop runs and submit runs during weekly work sessions. Instead of using points to indicate a develop run or a submit run, we used small tick marks: A tick above the line indicates a submit run and a tick below the line indicates a develop run. This style of indication is simple and quickly understood. Even with a high density of student activity, the chart was still readable. Figure 6 shows Version 5.

There was one other design consideration we tested for Version 5. Before putting straight tick marks above and below the data line, we used straight and diagonal tick marks to indicate a develop run and submit run respectively. This worked when the density of

activity was low, but became difficult to differentiate a straight tick from a diagonal tick when the density was high.

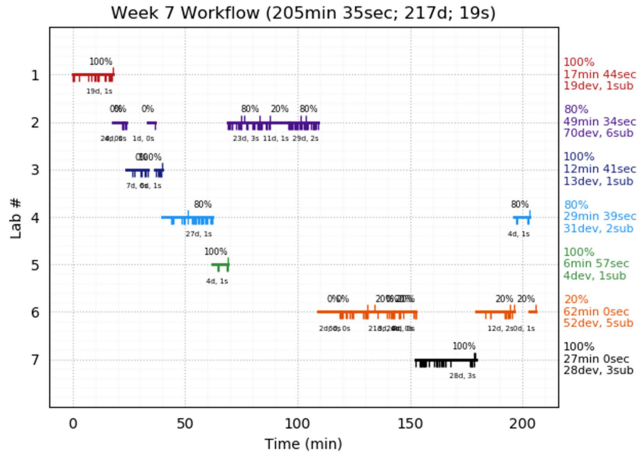

Figure 6: Version 5. Added tick marks below the time lines to indicate develop runs, and tick marks above the time lines to indicate submit runs.

3.6 Version 6 -- Pivot indicators

Version 6 is the current version of the workflow chart that we use today. A final addition is the ability to identify pivots. One unique benefit of the MSP approach is the ability to pivot, which is when a student switches from one lab to another before finishing the first lab. If a student gets stuck on one lab, they can just move on to another, often coming back to finish the earlier lab (or having gotten help in the meantime). We added arrows on the workflow chart to indicate when pivoting occurs. Figure 7 shows Version 6 of our workflow chart.

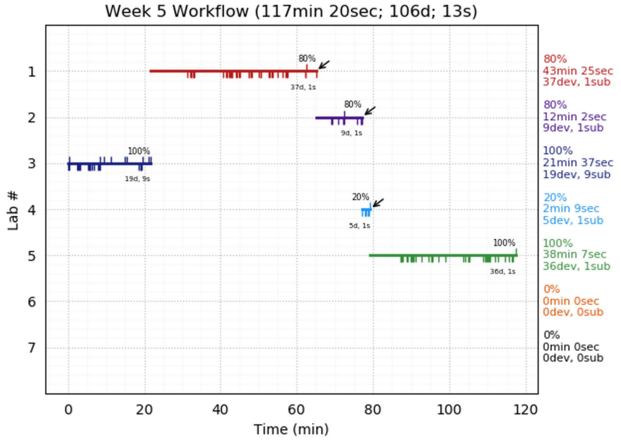

Figure 7: Version 6. Added arrows to indicate pivots.

In Figure 7, the first pivot can be seen on lab 1 since the student only scored 80% and then switched to work on lab 2. Pivoting arrows are useful if a teacher is interested in them, but adds a small amount of clutter. As such, we added a flag to control generation of these pivot arrows on the workflow chart.

4 Current uses and discussion

Version 6 of the workflow charts has the information we desired, available at a quick glance. We can see summary data for the week, specific data for each lab activity, and can even see special information like pivots. Section 4 discusses our primary uses of these workflow charts.

4.1 Understanding student effort

From the beginning, our motivation was to create a visual representation of data to understand student effort on our MSPs. These workflow charts help us to quickly and accurately see lots of meaningful data in a single location. We can pick any week of the quarter and any student and see why they may be struggling or even performing better than other students in the class. We have already used these charts for many analyses regarding research, individual student considerations, and to generally improve our MSPs and our CS1.

4.2 Detecting unallowed collaboration

In 2017, we began allowing our students to collaborate when working on lab activities. We allow students to collaborate only if they do a majority of the work and they indicate on their submissions who they worked with. We have a variety of ways to ensure each student is submitting ethical work, and among them are using these workflow charts to visually notice any irregularities. Recently, we started showing students these generated charts and having them call out any charts that look 'weird' as a participation activity.

4.3 Student classifications

One other way that we've begun using these charts is to create student classifications to help us identify students that may be struggling. In a 10-week quarter, we typically generate over 1,000 workflow charts. If we can use these charts to make meaningful and accurate classifications, then we can identify struggling students early and provide additional resources that will help them succeed. Some classifications that we are currently using are when do students begin working, do students complete all lab activities in a single day or spread them out, and how much time do students take to complete all lab activities.

4.4 Website

We are creating a website to share these workflow charts with our students and the community. Instructors can upload the auto-grader's log files for a week's labs, and the charts are automatically generated on a new webpage. There will also be sorting functionality integrated into the webpage so instructors can point out key features like students who spend the most amount of time working or students who complete the assignment with the least submits or develops. In the future, we may investigate integrations directly with the auto-grader so that no log file uploading is necessary. Figure 8 shows a screenshot of workflow charts from multiple students which will eventually be featured on a webpage.

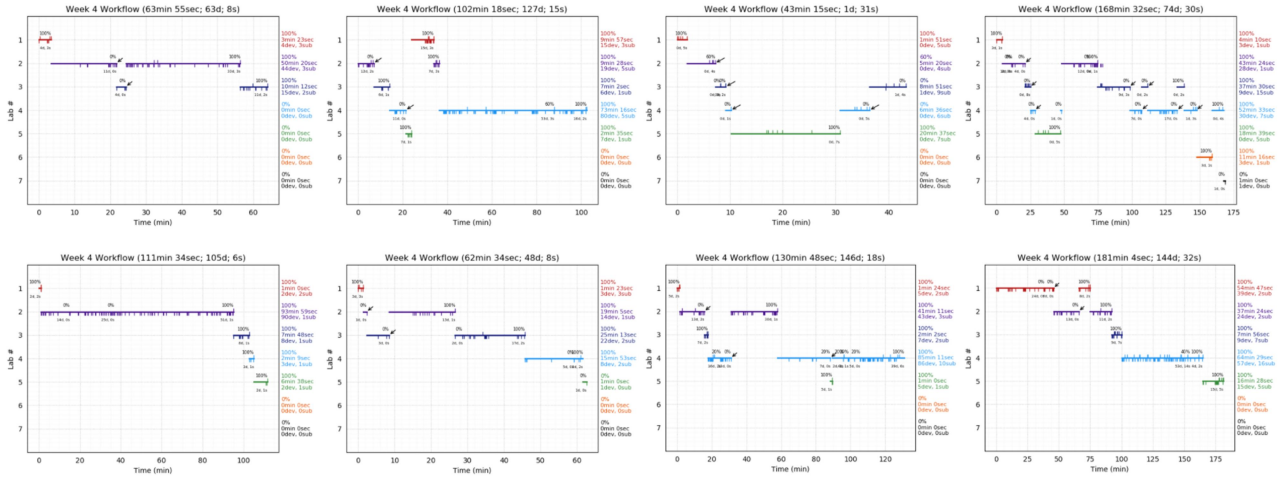

Figure 8: Workflow charts for multiple students

4.5 Future improvements

These workflow charts have evolved since 2018, but we are considering further improvements. First, we would like to add an indication of when students began struggling with lab activities compared to their peers. This would make comparison analysis easier when looking at a single week for an entire class. Second, we would like to further develop and use the classification system for these charts. As of now, we are using basic classifications, but if they could be more robust and accurate, we could create impactful intervention techniques for struggling students. Third, we would like to incorporate the original calendar view from Version 1. A weekly calendar view provides important data that'd we'd like to capture if possible, so deciding on the proper way to do so is a goal for Version 7.

5 Conclusion

We described the evolution of a graphical representation, called "workflow charts", of student effort on weekly many-small-programs. We have used these charts in our teaching each quarter, to help provide insight into our class, get a quick feel for a particular student's effort when they come to office hours (for example), and even to help us decide to investigate potential cheating when a student's workflow chart shows almost no effort but high scores. The paper focuses on introducing the concept of such charts as a tool for teachers and showing the evolution of the design; that evolution may be of interest in itself, as more education-focused tools focus not necessarily on algorithms or traditional considerations but rather focus heavily on design considerations. We have found such charts quite useful in our teaching, but we encourage future work (and plan to conduct some ourselves) that demonstrate specific benefits, like detecting struggling students, or reducing cheating.

REFERENCES

[1] zyBooks.https://www.zybooks.com/. Accessed: August, 2020.
[2] Gradescope. https://www.gradescope.com/. Accessed: August, 2020.
[3] Mimir Classroom. https://www.mimirhq.com/. Accessed: August, 2020.
[4] Vocareum. https://www.vocareum.com/. Accessed: August, 2020.
[5] Codelab. https://www.turingscraft.com/. Accessed: August, 2020.
[6] Pearson's MyProgrammmingLab https://www.pearsonmylabandmastering.com/northamerica/myprogrammminglab/. Accessed: August, 2020.
[7] Runestone. https://runestone.academy/runestone/default/user/login?_next=/runestone/default/index. Accessed: August, 2020.
[8] BlueJ. https://bluej.org/. Accessed: August, 2020.
[9] 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.
[10] 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.
[11] Gannt Chart. https://www.projectmanager.com/gantt-chart. Accessed: August, 2020.