Enabling Grouping and Sorting of Student Submissions in Classroom Learning Partner

Sarah Scodel

Massachusetts Institute of Technology

Class of 2013

May 16, 2013

1. Introduction

In the past decade there have been numerous advances in educational technology and many believe that technology is the future of education. Dr. Kimberle Koile's Classroom Learning Partner Group of the MIT Center for Educational Computing Initiatives has been developing tablet-pc-based classroom interaction software that allows teachers to display lessons in "electronic notebooks" on student tablet pcs and on a public display [Koile & Rubin 2013; Rubin, Storeygard, & Koile 2013]. The teacher can annotate these lessons with "digital ink", and students can solve problems using the digital ink on their own tablets and wirelessly submit the answers to the teacher. Because students are encouraged to submit multiple answers if they think of alternate ways to solve problems, a teacher may receive many submissions. In a class of 25 students, for example, a teacher easily could receive 30 to 50 responses for a single problem. Thus, a teacher needs a way to quickly and easily view the student submissions without becoming overwhelmed.

The following scenario illustrates several ways in which the software can help teachers view submissions by means of tagging student submissions, e.g., as correct, incorrect, and then sorting submissions by those tags. This functionality would allow the teacher to easily find

pedagogically interesting examples of student work to show her class, identify students who are struggling, and categorize student work.

Ms. Smith wants the students in her math class to practice working on simple word problems. She uses her tablet to create five electronic notebook pages each containing a word problem. Her students open their electronic notebooks on their tablets, work on the problems, and submit the notebook pages back to her. She receives many submissions for each page and wants to display three submissions for each page on the projector as examples to her class. As her students work the problems, she starts to read through the submissions, marking which ones she things will be good examples to display for the class. She "stars" examples as she continues to circulate among the students. Finally having chosen 15 examples, she sorts the student work into starred and un-starred submissions, and begins displaying the examples to the class for discussion.

Ms. Smith then wants to see who solved the problems correctly, so she reads through the submissions, marking each as correct or incorrect. She can sort the submissions by their correctness, see how many students understood and successfully solved the problem, and identify which students may need help. Later, after she has given the class new problems to solve, she can see if the students who got the previous problems incorrect can now solve the new problems correctly. Because she can quickly see how many submissions are correct for each page, she can determine which problems were most difficult for students and focus more on explaining the related concepts.

My research centered on creating an easy to use user interface for the teacher that supported sorting student work by multiple criteria and tagging submissions. Students may be absent or may forget to submit certain pages, so I added the ability for the teacher to see a list of students who have not submitted answers for a certain page. Because it may be beneficial for

students to work in groups and submit one submission for the entire group, I added the ability for the teacher to sort the submissions by groups, as well as sort individual work by group membership. Finally, I added the ability for teachers to tag individual submissions and then sort all student submissions by the values of those tags.

In Section 2 of this report I give background information on the Classroom Learning Partner software and how it is used. I describe the previous state of the program and identify the areas for improvement. Section 3 describes my contributions to the software and the manner in which the contributions extend and build upon the previous version. I discuss the design and implementation of features that give a teacher feedback about which students have not submitted answers, and that enable teachers to sort by group submissions and tag both individual and group submissions. In Section 4 I describe how these features were used in the classroom, and I propose future work in Section 5.

2. Background

Classroom Learning Partner (CLP) is a tablet-pc-based program designed to improve classroom interactions between students and teachers by enabling teachers to create an electronic notebook containing problems for their student to work. The students can write on their notebook pages using digital ink and submit those pages back to the teacher. When I joined the project in January 2011, the teacher was able to receive multiple submissions from multiple students and sort those submissions by the student's name or time the submissions were received. Figure 1 below shows the user interface (UI) that enables teachers to view multiple student submissions. The teacher could select a page from the preview panel of pages on the left, and tap on the person icon for a page in order to view previews of submissions for that page. Tapping on any

page, either the original notebook page or one of the submitted pages, would display that page in the main display window on the right. A combo box at the top of the submissions panel enabled teachers to sort all the submissions by student name or the time the submission was received.

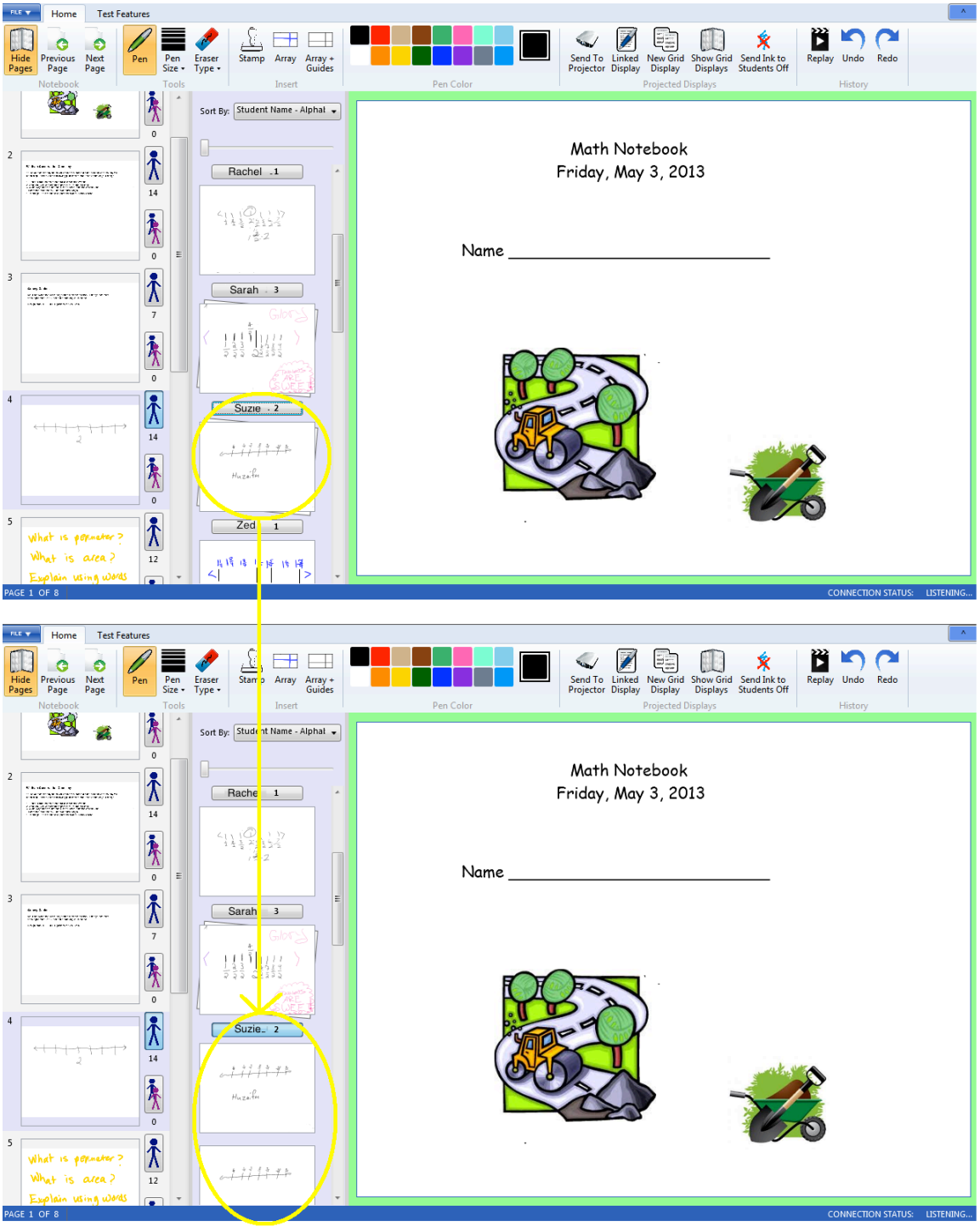

Figure 1. Two views of the teacher's user interface for looking through submissions. The teacher has selected to view the submissions for the first page of the notebook. In the top view, the teacher sees only the most recent submission for each student. In the bottom view she has clicked on Suzie's name to show all her submissions.

For each page, each student had a label with his or her name and their number of submissions, followed by a small preview of their most recent submission. Tapping the label with their name on it expanded the student's submissions to show all of their submissions sorted by the time received. If the teacher wished to hide all except the most recent submissions, she could tap on the label again to hide all but the student's most recent submission.

3. Contributions

In the January 2011 UI, the teacher had no explicit way of seeing which students in her class had not submitted any work for a particular page. In a previous version of CLP if a student had not submitted any work a blank page was displayed under that student's name, but this scheme was not very space efficient and did not provide a way for the teacher to see quickly and without scrolling all students who had not submitted work. Figure 2 shows this UI.

In the January 2011 UI, the teacher was not able to have students work in groups and submit only one answer per group, with the answers sorted by group, rather than by the individual who happened to submit the answer for the group. Individual student answers also could not be sorted by group, with one "stack" of student answers for each group.

My work aimed to improve grouping and sorting in the CLP UI by:

- Displaying the names of students who have not submitted
- Enabling sorting and grouping according to the groups of students the teacher designated
- Creating a new architecture enabling teachers, and potentially students, to tag individual and group submissions pages, and enabling teachers to sort by those tagged values.

The following sections describe these contributions.

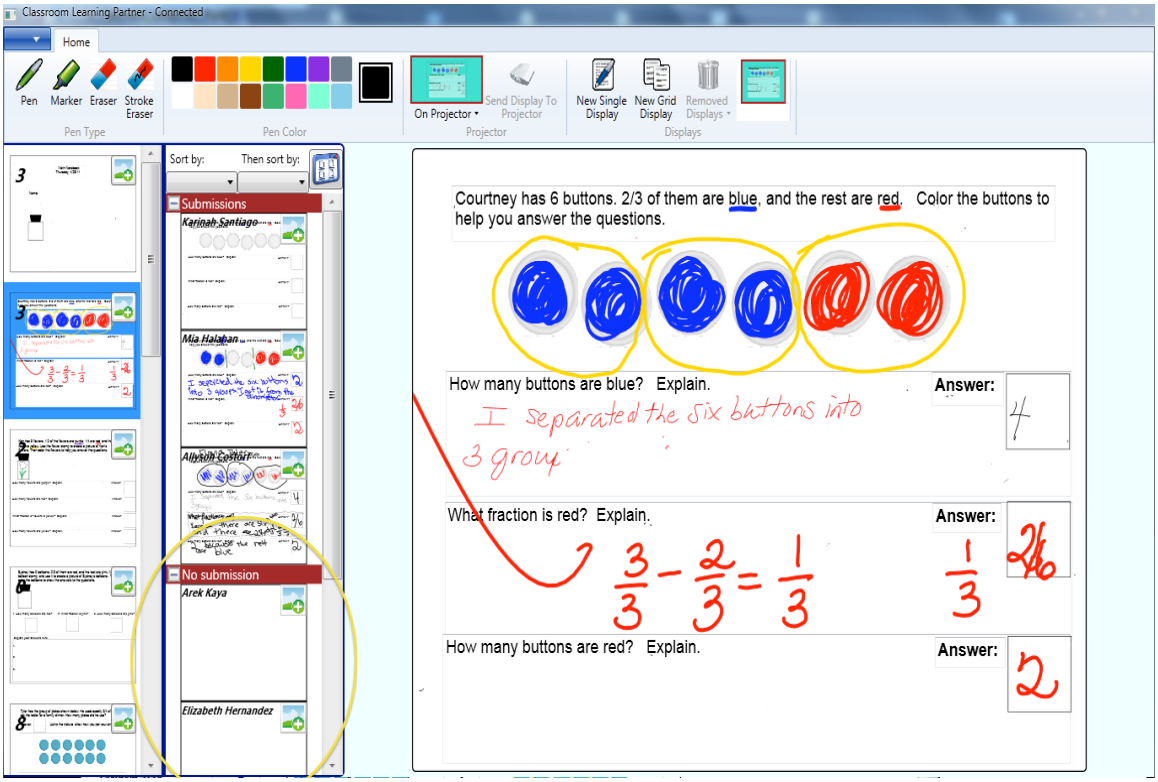

Figure 2. January 2011 UI: One blank page is shown for each student who has not submitted anything

3.1 Submissions vs. No Submissions

3.1.1 Purpose

Teachers using the CLP software typically have classes of over 20 students, and since students can submit more than one answer per problem, the number of submissions a teacher has to deal with for a particular lesson can get quite large. It is, therefore, important to enable the teacher to quickly and easily view student submissions. It also is very useful for a teacher to quickly and easily know how many and which students still have not submitted answers to particular problems so that the teacher can pace the class accordingly and identify which students might need help or encouragement. The teacher will know, for example, if she has to give the students more time to solve the problems and when she may be able to move on to the next problem earlier than she had planned.

3.1.2 Implementation

When a teacher starts CLP and opens up an electronic notebook at the beginning of a lesson, a list of all the students in the class is instantiated (currently from a file on the desktop of the teacher's tablet). As students work problems in their copies of the notebook on their tablets, they submit the pages to the teacher's tablet. The teacher's tablet maintains a list of these submitted pages, with each page tagged with a "SubmitterName" property indicating the name of the student who submitted it. I added a field, called "StudentsWithNoSubmissions", for each notebook page on the teacher's machine. The field is initialized to contain the names of students who do not match the "SubmitterName" property of any of the already submitted pages. Each time a student submits a page, their name, is removed if it exists, from the "StudentsWithNoSubmissions" field. At the bottom of the preview panel for a page's student submissions is a section labeled "No Submissions" under which appears a list of all the students who have not submitted anything for that page. Figure 3 below shows the previous user interface with blank pages for each student who has not submitted and the new interface I created.

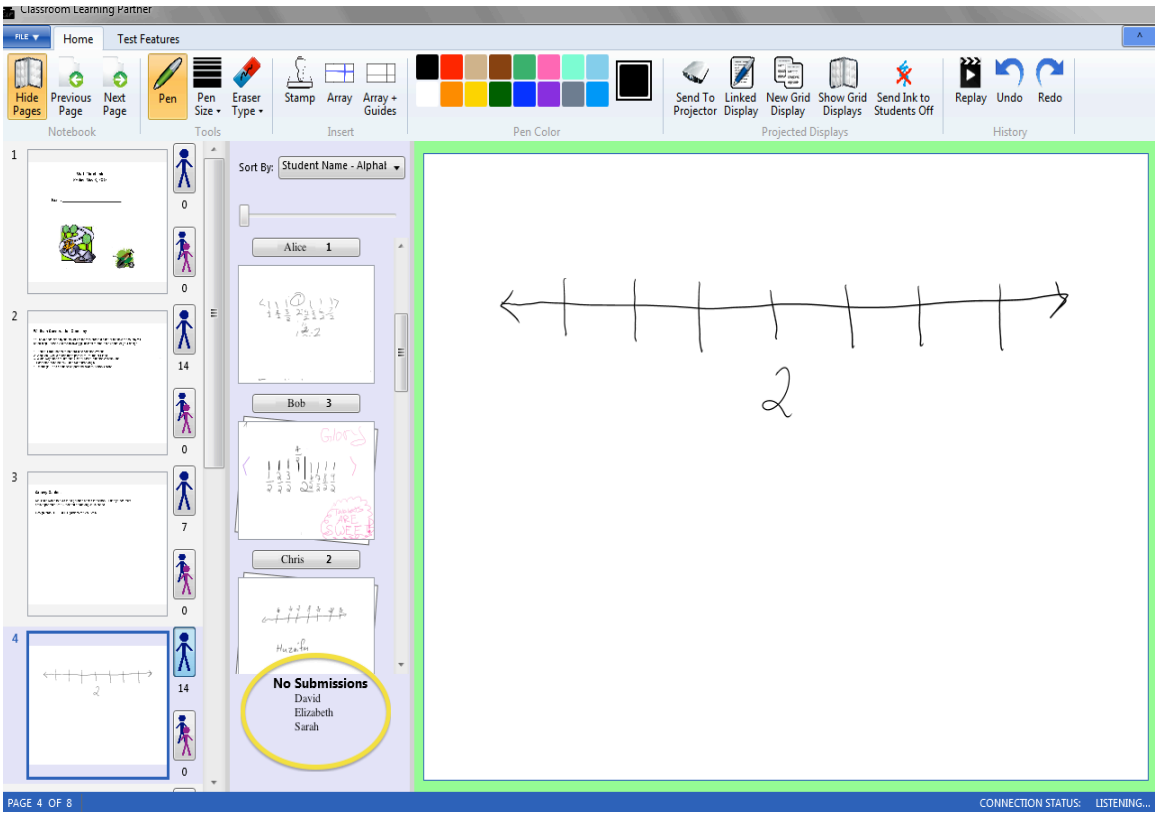

Figure 3.A list at the bottom of the submissions preview panel displays the names of students with no submissions.

3.1.3 Design Tradeoffs

When displaying the names of students without any submissions I had to consider both how and when to display the names. In the January 2011 version there was a blank page under a students name for every student who had not submitted, but this took up a lot of space and did not provide a way for the teacher to see all the names together. To avoid these problems, I opted to display a list of names at the bottom of the preview panel. So that the names would always be visible, I shortened the scroll bar on the preview panel. While this scheme might require the teacher to have to scroll more in order to see the submission pages, in practice the teacher who trialed this feature in her classroom did not mind.

In addition to sorting submissions by student name, the teacher can sort pages by group name. (See next section for details.) With typical class sizes of between 20 and 30 students, the

number of groups, ranging in size usually from two to four, is small enough that no listing of group who have not submitted work is necessary; a teacher can quickly scan the list of group names to determine if any are missing submissions.

3.2 Group Submissions

3.2.1 Purpose

Students often learn through collaboration with their peers, and teachers often will have students work together in small groups to solve a problem. While each student has his or her own tablet, it would be useful to designate groups of students and have one answer submitted per group, e.g., by a student designated as group submitter. The teacher then might want to sort work by group rather than by individual student.

3.2.2 Implementation

I decided that it would be useful to sort both group submissions and individual submissions according to which group a student was in. In order to designate groups, the file containing the list of students needs also to include a group name (or number) for each student. In the UI, we created new options in the sort combo box: "Submissions by Group Name" and "Group submissions" display individual submissions sorted by group, i.e., all student submissions for members of a group are "stacked" together; and group submissions sorted by group, i.e., order the answers submitted for each group, respectively. Figure 4 shows individual submissions sorted by group.

When sorting individual submissions by group there is a large label at the top giving the name of the group and specifying that we are looking at individual submissions, followed by a list of submissions for that group sorted by student name alphabetically. Tapping the label with the group name on it can collapse the list of submissions; tapping again expands the list. If the submissions are not collapsed, all students in the group will be shown along with their number of

submissions and a preview of their most recent submission. Again tapping on their name label toggles between showing and not showing all submissions.

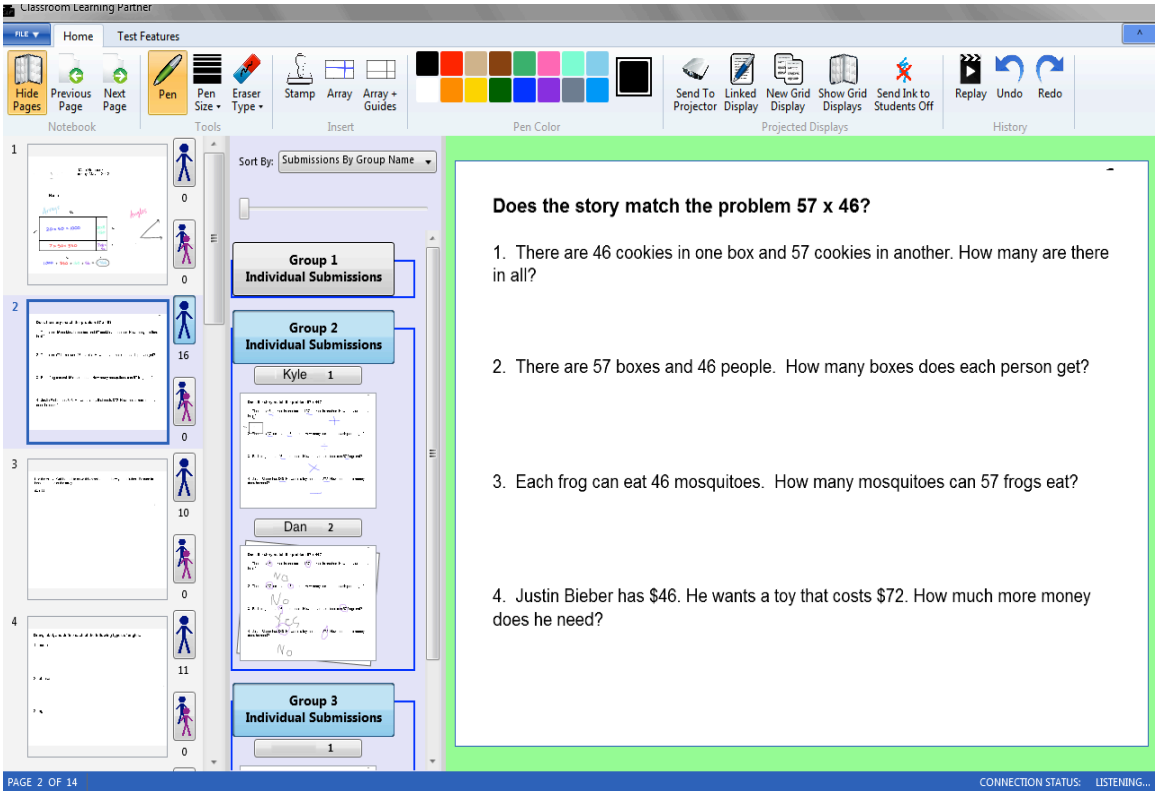

Figure 4: Here we see individual submissions for three groups. The submissions for the top group have been collapsed and the submissions for the next 2 groups are sorted by student name. In the bottom group all of Kyle's submissions are shown together and all of Dan's submissions are shown together.

When sorting group submissions by group there is a top label with the group name, below which is a preview of the most recent group submission along with the name of the student who submitted it. Each group of submissions is collapsible so that if a teacher has finished looking at a particular group's work she can collapse the submissions from that group in order to easily look at submissions from other groups. Because groups may have several submissions by the group submitter, as well as accidental submissions from other students not designated as the group submitter, tapping the label above the first group submission will expand the rest of the submissions so that the teacher can see all previous submissions sorted by the time they were

submitted, as well as identify which students may be submitting for the group by accident. Figure 5 shows the collapse of the submissions in one group as well as the expansion of submissions in another group.

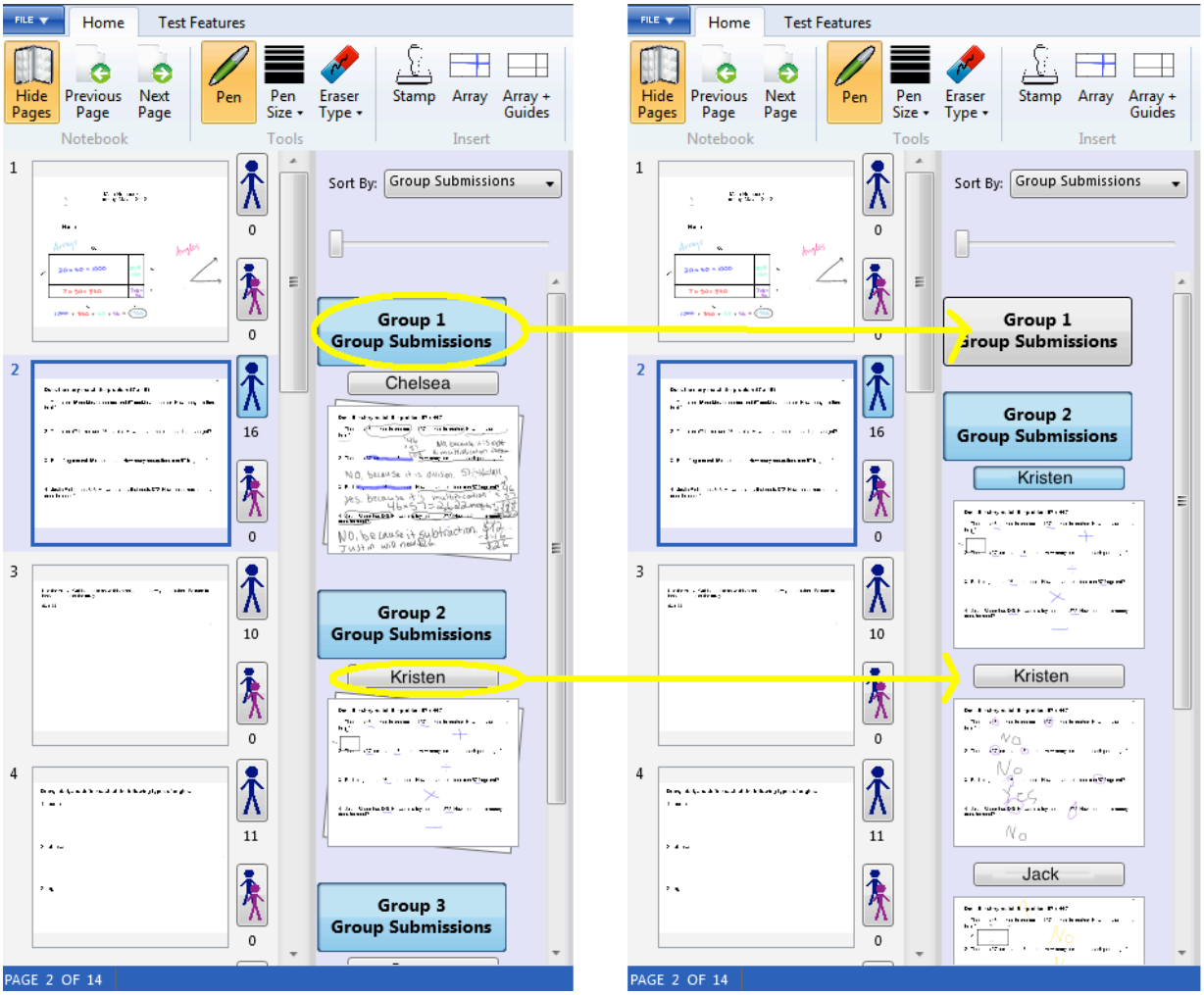

Figure 5: Group Submissions. The view on the right shows the initial view as soon as the teacher selects Group Submissions in the combo box. In the view on the right the submissions from the top group have been collapsed and the submissions from the bottom group have been expanded to show all submissions sorted by time.

3.2.3 Design Tradeoffs

There were several considerations made in choosing this UI for groups, including whether to sort individual submissions as well as group submissions according to the group to which a student belongs, whether to show individual names above group submissions, how to distinguish which student was designated as the group submitter, and whether to show accidental

submissions. It seemed useful also to sort individual submissions by group because although the students may not have collaborated on those submissions, their previous collaboration on other problems could lead to similar methods in solving problems, or the teacher might easily identify if that group worked well together or if there were some members of the group who struggled at expressing their answers while others excelled. Thus if the teacher selects to sort by "Group submissions" only group submissions are shown and individual submissions are filtered out, and if the teacher selects "Submissions by Group Name" as her sorting function, only individual submissions are shown with group submissions filtered out.

Individual students names are displayed above group submissions because students not designated as the group submitter often hit the "group submit" button by accident, so there had to be a way to distinguish the intended group submissions. Occasionally a student might hit the group send when trying to hit the individual send, so if a student was missing individual submissions the teacher could easily check to see whether the student submitted a group submission by accident.

In order to distinguish which student was the official group submitter, I considered creating an extra property on the CLPPage object representing the submissions pages such as "IsGroupSubmitter" so that we could filter out the group submissions that were not submitted by the group submitter. I also considered adding an asterisk or other identifier to the student names list to identify the designated submitter. Since the teacher might want to change the designated submitter fairly often, however, or even allow the students to choose for themselves who would be the group submitter, it may not be easy to know the designated submitter ahead of time. Instead, the software considers the person who submitted the most recent submission to be the group submitter. If only one student submitted group submissions, it is easy to tell which student

is the submitter. If multiple students submitted, the teacher could just tell the group to have the group submitter submit once more, and that submission then would be the most recent.

3.3 Teacher Tagging

3.3.1 Purpose

When evaluating many pages of student work, the teacher may not be able to remember which submitted pages are correct and which are incorrect, and it may be hard to remember which pages would make good examples to show the class. The ability to tag student work as correct or incorrect, star certain submissions that may be useful to show the class later, or tag student work according to other attributes is very valuable in analyzing student work. A teacher also may want to sort student work by whether it was correct or incorrect so she can see trends in work that was incorrect, or see all the different strategies used by students who solved the problem correctly. We needed a way for the teacher to be able to save tags on submissions and view those tags later, as well as group and sort student submissions according to the tagged values.

3.3.2 Implementation

There were three main steps in enabling tagging of student work: designing the architecture used to create the tags and add them to student pages, design the UI to enable the teacher to tag student submissions, and finally design the UI for enabling the teacher to view the tags and group and sort submissions based on the tagged values.

There could be many different types of tags, including those specified by a teacher, such as correctness, starred, page-topic; or those added to a page as the result of machine interpretation of the student work on the page, aka domain-interpretation tags. (See [Koile & Rubin 2013] for discussion of machine interpretation of student work.) The *TagOptionValue* is a new class containing the fields *value* and *icon* indicating a potential value for a tag and an icon,

which is linked to a visual representation of the value (such as a check mark for a value of correct or question mark for a value of unknown). To support the different types of tags I created a new interface, called *TagType,* which had the following fields:

- *Name* – either correctness, starred, page-topic, or domain interpretation
- *InElevatedMenu* – a boolean value of either true or false indicating whether this would be displayed directly or indirectly in the UI showing all the tags (described in the next section)
- *AccessLevel* – either teacher, student, researcher, or database indicating who gets to see the tag
- *ExclusiveValue*- a boolean value of true or false indicating whether the tag can have multiple values. A correctness tag for example would have an *ExclusiveValue* of true since a page cannot be both incorrect and correct at the same time, while a page-topic tag might have multiple values
- *ValueOptions*- a list of *TagOptionValue* objects indicating the potential values the tag could have, such as correct, incorrect, or unknown for a correctness tag

Once I had defined this interface I created *CorrectnessTagType, StarredTagType, PageTopicTagType* and *DomainInterpertationTag* classes that implemented this TagType interface.

Finally, in order to support the creation of tags, I defined a new class called *Tag*, which had the following fields:

- *Origin* – Either teacher, student, or auto indicating whether the tag originated from the teacher, from the student whose work was tagged or from machine interpretation
- *TagType* – Either *CorrectnessTag, StarredTag, PageTopicTag* or *DomainInterpertationTag* indicating the type of tag

- *Value* – a list of *TagOptionValue* objects, all of which had to be in the *ValueOptions* field of the *TagType* object indicating acceptable value for the type of tag

In order to support adding and saving tags I added a field to each *CLPPage* called *PageTags* containing a list of tags for that page. Because any submission page is either correct, incorrect, or unknown and will be either starred or unstarred, each *CLPPage* object was initialized with its *PageTags* field containing a tag of *CorrectnessTagType* with a value of unknown and a *StarredTagType* with a value of unstarred. Once the teacher received the submission, she could use the UI to change the value of those tags to correct or incorrect, or change certain submissions from starred to unstarred. If she wished to add a different type of tag to a page, she could add a tag to that list of tags after the page had already been created using the CLP authoring UI. As long as the notebook is saved, the list of tags persists, and the next time she opens up the notebook all the tag values will be there.

Once the tag architecture was set up so that tags could be created and modified, I had to create a simple and intuitive user interface for the teacher to add tags or change the value of existing tags for every page. In order to mark pages as correct, incorrect, or unknown or toggle the values of the starred tag from unstarred to starred, I created a new class, called *HoverBoxView* which is a visual representation of all the tags on the page. This visual representation is shown below every page in the submissions list and has buttons with the icons listed in the *TagOptionValue* classes for correct, incorrect, and unknown or starred and unstarred. Clicking the buttons toggles the value of the tag and updates the *Value* field of the tags contained in the *PageTags* field of that submissions page. The current value of the tag is indicated by a purple background of the icon for the current value of a correctness tag, or an empty star for an unstarred tag value and solid yellow star for a starred tag value. To see other tags such as page-topic tags, an icon with three ellipses is shown; hovering over it displays a list of all the tags.

Thus the *HoverBoxView* serves as both a display of current tags as well as an interface for changing tags.

In order for the teacher to add a tag with type *PageTopicTagType* to a submission, she could simply enter a new page-topic tags in the authoring UI's page-editing mode by clicking the "Add a Page Topic" button. Existing page-topic tags are saved and appear in the text field; to add a new page topic the teacher simply enters comma-separated values of new page topics.

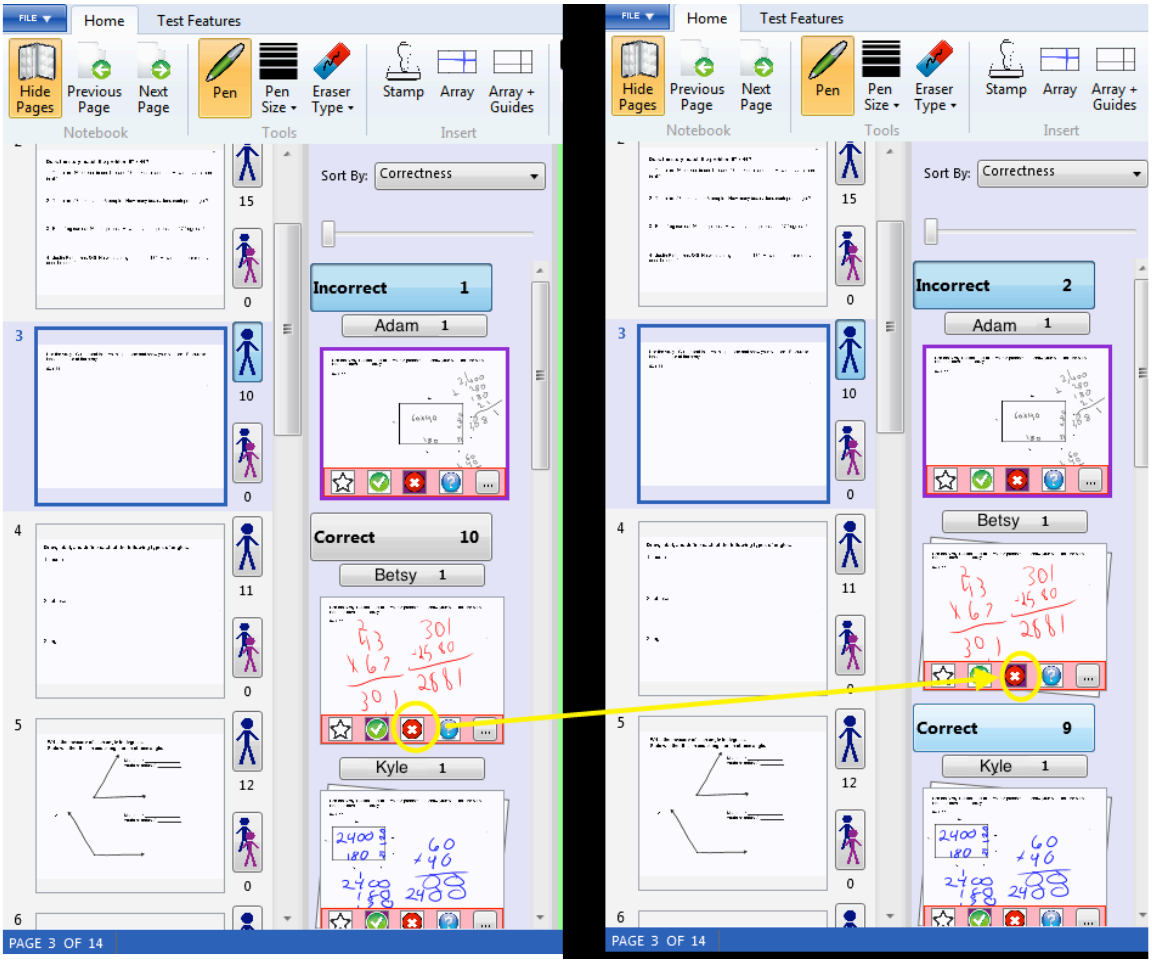

Figure 6: Submissions panel sorted by correctness. Clicking on the incorrect icon for Betsy's submission in the view on the right toggles the value of the correctness tag from correct to incorrect, and the UI automatically moves Betsy's submission from the group of correct submissions to the group of incorrect submissions.

In addition to simply displaying the value of tags for a given page, I added the ability to sort by tagged values. The combo box at the top of the submissions box is populated by all the names of the tags on the pages. If the teacher decides to sort by tag value, the submissions

appear with a large label for every possible tag value, with the student names and a preview of each submission with that value below it. As in sorting by group, all tags with a certain value can be collapsed so the teacher can easily see other submissions. If a teacher changes the value of tag the UI will be updated in real time, such that submissions can be moved between groups if the teacher changes her mind about a submission. Figure 6 shows this happening.

3.3.3 Design Tradeoffs

There were a few considerations when designing the UI for the teacher to view and modify tags such as which tags to display visually and whether to show multiple submissions for each student when sorting by tagged value. Because each student submission page is initialized with both a correctness tag with value unknown and a starred tag with value unstarred, each submission page always has exactly one value for a correctness tag and one value for a starred tag even if the teacher does not explicitly give the page a tag value. For other tags, such as page topics, there can be zero, one, or many values of that tag. Because correctness and starred tags have only a few options for their values and can only have one value at a time, those icons were easy to display so I put them on the panel displaying the current tags. There were only a few options (correct incorrect and unknown for correctness and starred and unstarred for starred) so it would be easy to display all possible values in a panel. Page-topic and domain-interpretation tags would likely have many potential value options and many values at the same time so it would be hard to display them. Thus those types of tags are displayed in an expandable list upon hovering over the ellipses tag.

Because each student may submit multiple submissions, whether on purpose or by accident, I considered only showing up to one submission of each tag value for each student (so if a student had two incorrect submissions and three correct submissions for a particular page, only one incorrect and one correct submission would be shown). Because students may reach a

correct answer by using completely different logic or may submit multiple incorrect submissions that are wrong for different reasons, however, I decided that it would be useful to show all submissions for a student when sorting by tagged value, but initially have only the most recent submission shown for each student and the rest expandable. Further testing in classrooms will give us feedback from teachers about these options.

4. Uses and Reception in Classroom

The user interfaces for sorting submissions by student work and displaying the names of students who had no submissions were tested in a fourth grade science classroom in March and April. The teacher found that the sorting of group submissions by group was helpful, but students would often hit the group submit button instead of the individual submit button or vice versa. Based on that feedback we decided to be enable sorting of both group and individual submissions by group since submissions would occasionally be incorrectly marked as group or individual.

The UI that displays which students have submitted and which have not was testing in a fourth grade math class in May, and the teacher found the list of students who had no submissions at the bottom of the submissions panel very useful. She was quickly able to identify which students had not submitted answers and went to talk with them individually to see if they were having trouble. On one of the days in her classroom, she used a smaller tablet than I used for the initial UI design, so she could only see one or two submissions at a time since the list took up so much space. Based on that interaction I changed the length of the list of students to be dependent on the overall size of the submissions scrollbar so that the proportions would be the same no matter the size of the screen. Overall the teacher received my changes positively and

the feedback I received gave me insight into future improvements and extensions that could be made to the UI.

5. Future Work

The architecture for tagging that I implemented can support many different types of tagging. Currently only teacher tagging of correctness, starred, or page-topic is implemented, but the TagType interface can be extended easily to support student tagging or automatic machine interpretation tagging. Only a new class implementing the *TagType* interface would have to be created since all pages now have a list of tags, and it is trivial to add a new tag to the *PageTags* field of a CLPPage. This feature could be very useful because a teacher could specify a type of tag, such as strategy or difficulty, and then give potential values that would be displayed to the students. The students then could indicate to the teacher which strategy they used in their problem solving or how difficult they found the problem, and the teacher could easily group and sort submissions by those values. Machine interpretation code also could analyze student work and automatically populate a tag with values and add it to the list of tags for a certain page.

The UI for sorting tags could be improved by always displaying correct submissions first, then incorrect submissions, and finally submissions with unknown values. It may also prove useful to have a two level sorting system in which the teacher can first elect to sort by one criterion such as student name or group name, and then sort by tag values such as correct, incorrect or unknown, or starred and un-starred. There are many different ways of displaying all the submissions, and future work should focus on expanding the functionality of sorting and grouping while not making the interface unnecessarily complicated.

6. References

Koile, K. and Rubin, A. Machine interpretation of students' hand-drawn mathematical representations. In *Proceedings of WIPTTE (Workshop on the Impact of Pen and Touch Technology on Education)*, 2013.

Rubin, A., Storeygard, J., and Koile, K. Supporting special needs students in drawing mathematical representations. In *Proceedings of WIPTTE (Workshop on the Impact of Pen and Touch Technology on Education)*, 2013.

7. Acknowledgements

I would like to thank my 6.UAP thesis advisor Dr. Kimberle Koile for her constant mentorship and encouragement through the project. Her tireless work in organizing trips to test the software in the classroom and collaborating with the teachers to make the software as useful as possible was truly inspiring. She always checked in with us every week to make sure we were on track and was always available for any help we needed. I would also like to thank Steve Chapman, the lead software developer, for all his help when I was coding. Steve was always available on gchat for any questions I might have, and worked incredibly hard to bring all parts of the project together.