Teaching Tools

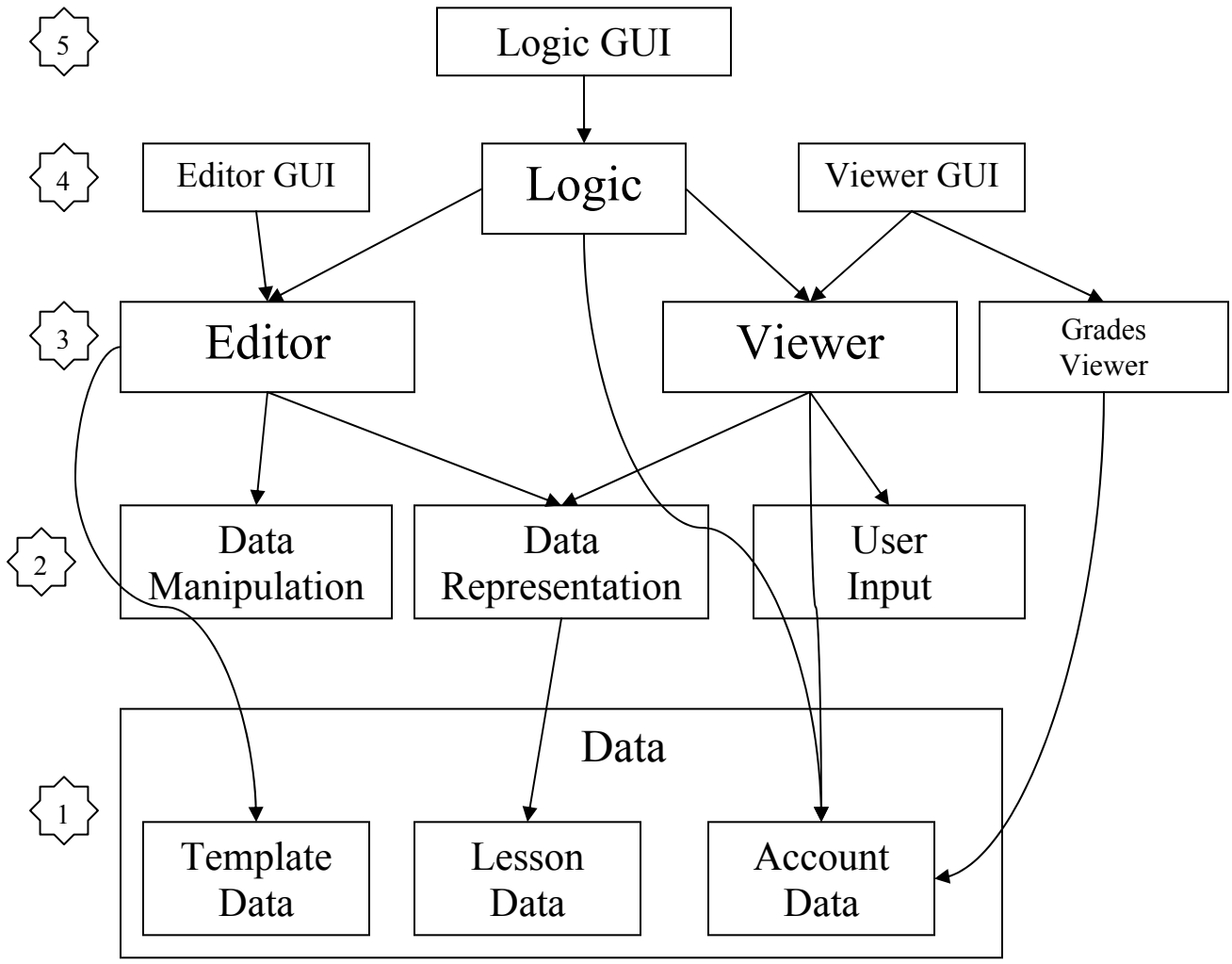
Top Level Design

Louisa Rosenheck

February 21, 2003

1 Levelized High-Level Component Diagram

1.1 Diagram



1.2 Diagram Component Descriptions

1.2.1 Logic

The Logic is in charge of the flow of control of the program. It accesses the user's account data and lets the user log in and view which classes he is in and which lessons he has access to as well as which ones have been done. It then provides the interface for the user to choose which class and which lessons to view or edit, at which time it opens the designated module. Any time the user finishes viewing or editing one lesson, control returns to the logic which gives the user the same choices again.

1.2.2 Logic GUI

The GUI for the logic consists of all the windows that pop up when the user is getting started or transitioning between classes or lessons. These windows include account login, course selection, teacher table of contents, and student table of contents.

1.2.3 Editor

The lesson editor reads from and writes to the lesson data to load and save lessons. It uses the information stored in the data representation module and keeps track of what objects are selected. Once loaded, the user can edit the slides by using the functions available in the data manipulation module. When this is done, the editor updates the modified object in the data representation module.

1.2.4 Editor GUI

The editor's GUI contains a canvas displaying all the elements of the current slide in the lesson being edited, and those elements can be selected, moved, created, and deleted. It also contains buttons and menus for all the options available to manipulate the objects as well as to move back and forth to different slides in the lesson. Exercises and tests can also be created with this editor by adding elements made for user input. If the user chooses, he can use the templates in the template data to create the lessons.

1.2.5 Viewer

The lesson viewer is for previewing or working on lessons, and for assigning exercises. It loads lesson data and reads it from the data representation module. It keeps track of where the student has worked through and what page he is on. It also utilizes the user input module to read students' answers for practice exercises and quizzes and tests and then stores it in the user's account data.

1.2.6 Grades Viewer

This viewer lets the teacher view students' grades and answers to tests and quizzes as well as any other assignments done through the lessons. Here the teacher can also see the student's progress.

1.2.7 Viewer GUI

This GUI provides the screen for the user to view and work on the lessons. It consists of a screen to see the lesson and enter answers, and controls to go back and forward or stop. For teachers, it also contains the grades viewer's interface, which consists of a spreadsheet-like chart and buttons to choose to view students' tests and assignments.

1.2.8 Data Manipulation

Data manipulation consists of all the functions available for editing the various objects on a slide. This includes changing the color, shape, and size of text, images, and shapes. It is told by the editor to manipulate an object in a certain way and returns the modified object to the editor.

1.2.9 Data Representation

Data representation keeps track of the lesson currently being edited. It contains various data structures to hold the elements to be displayed on each slide. These elements are modified by the editor when the user edits them.

1.2.10 User Input

The user input is used in the lesson viewer to read the answers entered by the student user and return it to the viewer so it can store it when all the answers have been entered.

1.2.11 Data

The data may be stored in a database or in data files. Account data stores all user identification data as well as the progress they have made and for teachers, other data such as assignments and grades that they want to store. Lesson data is the objects and layout of those objects in the slides of each lesson. Template data is the objects and layout of various suggested templates available for creating lesson slides.

2. External Dependencies

There are no major external dependencies for this project, just some libraries that will need to be included such as the Standard Template Library and QT.

3. Project Roles

3.1 Project Manager

The project manager must coordinate all the members of the group by monitoring their progress and scheduling meetings. He will keep morale up while making sure everyone hits their deadlines, and adjust deadlines if necessary. He will keep communication lines open between all parts of the project and make sure everyone is aware of what is going on. Also, when decisions need to be made, he will take input from all members of the group and if no consensus can be reached, make the final decision.

3.2 Architect

The architect is in charge of designing the program, along with the input of the coders. Throughout the process, he will know the design of each component as well as all interfaces very well. If interfaces need to be adjusted (though they shouldn't) he will make final decisions and keep everyone involved informed and on track.

3.3 UI Designer

Since this program needs to be very user-friendly, it must have a well-designed accessible GUI, and the UI designer will design the look and feel of the program as well as the flow of control and organization for account management. Since there are a few separate GUI's, the UI Designer will have to make sure that all the GUI's are consistent and efficient. Since this job is fairly front-heavy, with most of the designing being done towards the beginning, he may be able to help the Tester towards the end.

3.4 Coders

There will be about 4 coders, depending on how many people are in the project group. These are the only people that will do the actual coding of the program, and they will never have to worry about design or user interfaces, only about the implementation of each part. They will have to code and test their components as well as debug them when the tester finds bugs. The coding responsibilities will be broken down into GUI (2), Logic, and Editing/Viewing.

3.5 Documenter

The documenter will work closely with the program manager and architect to know every part of the program and all its features and integrate the documentation of each part. His basic responsibilities include writing a user manual, and documenting the design and each component. However, the explanations of the design and components will be primarily written by the architect and coders and then edited and compiled by the documenter. The documenter will also have to keep a log of all milestones, progress, and changes made throughout the development process.

3.6 Tester

Coders are responsible for thoroughly testing their own components, but the tester will test each component more rigorously and in connection with all other components. He will test every possible combination of every function to find bugs and will document them when they are found and fixed. Additionally, if time permits he should test the software on potential users both to find bugs and figure out what could be improved upon in later versions.

3.7 Data Manager

The data manager will be in charge of file I/O and data storage. All the account data, templates, and lesson data will be stored either in files or in a database, and the data manager must determine the formats of these data and be in charge of storing them. He must communicate with the coders to specify how they should read and write any data and lesson files.

4. Schedule

Keep in mind the actual dates of this schedule may not be exactly accurate due to conflicting assignment schedules on the course webpage which is often unclear.

2/24	Choose Project Group
2/28	Top Level Design selected
3/3	Top Level Design revisions
3/5	Top Level Design frozen
3/10 (week of)	Write interfaces of all parts of program
3/17 (week of)	Discuss and revise interfaces
3/21	Interfaces frozen
3/24 (week of)	Develop detailed designs and implementations and begin coding
3/31 (week of)	Project manager and architect combine, revise, and approve designs
4/4	Detailed designs due
4/7 (week of)	Continue coding and component testing
4/16	System integration
4/28	All functionality implemented, documentation draft complete
4/28 (week of)	Extensive testing and debugging
5/2	Demo
5/5 (week of)	Complete all documentation and last debugging
5/9	Handin

5. Assumptions Made About Specifications

One assumption made that wasn't stated explicitly in the specifications document is that all lessons files store all elements as separate objects instead of in one pixel map, even in viewing mode, so that they can be reedited later.