**Special Scheduler**

**for**

**Creekside Park Elementary**

**Bruce Templeton**

**CS 470**

**04/29/2010**

**Table of Contents**

1. Introduction 1

2. Project Overview 1

 2.1 Specials 1

3. Project Requirements 1

 3.1 Special Requirements 2

 3.2 I/O Requirements 2

 3.3 User Interface Requirements 2

 3.4 System Specifications 2

4. System Design 2

 4.1 User Interface Design 2

 4.2 Classes 3

5. Software Design Process 4

 5.1 Testing and Debugging 4

 5.2 Challenges 5

 5.3 Work Break Down 5

6. Results 6

7.Summary and Conclusion 6

**1. Introduction**

This project was designed for the Title One department at Creekside Park Elementary School. The Title One department is tasked each year with coming up a schedule for all the classrooms at Creekside Park. They have to schedule all the specials and make sure there are no conflicts. These specials are classes and activities that the students are assigned to on a daily or weekly basic.

There currently are only two full time members of the Title One department at Creekside Park. The Title One department has been creating these schedule by hand using white boards to write everything out and visually make sure there are no conflicts. In the past, it has taken them around two weeks to make up these schedules.

**2. Project Overview**

The goal of this project is to assist with the scheduling of these specials and decrease the time it takes to make the schedule. The project gives the operator the option to manually schedule specials, while showing existing conflicts, if any. The operator also has the ability to set parameters for the specials and then let the program find optimal placement in the schedule for the specials. The operator can also import specials from previous years to speed up the process even further.

*2.1 Specials*

A special is a scheduled time in which students are required to be performing certain tasks. These scheduled the students will either be in their homeroom or another classroom or designated area. Some examples of specials are PE, Lunch, Recess, Math, Library, Music, Art, Health, Reading, and Science.

**3. Project Requirements**

The requirements set out by the Title One department were fairly loose. They were not sure what they wanted as a user interface in the beginning. They wanted a program that would let them know when they would have a scheduling conflict. They were not very descriptive on what they wanted for a GUI. However, they were very descriptive on what a special was and how it should be able to be scheduled. Because of the uncertainties the project was designed on a prototype method. Several of the requirements listed below were added at later dates. There were also several requirements that they decided to get rid of and are no longer in the project.

*3.1 Special Requirements*

A special may be scheduled once a week, like Music, or it may be scheduled as a daily item at the same time each day, like Lunch. A special may be assigned to a single class, or the entire grade. A special may have a location assigned to it. For example, students must leave their class to go to Music class. There can only be one special at a time per classroom. For example, there cannot be two classes in the Library at the same time.

*3.2 I/O Requirement*

There were not many requirements placed on the input and output of data by the program. The Title One department wanted a way to save current schedules and be able to open them to edit if need be. They also needed a way to be able to print the schedules. They optionally wanted to be able to import specials from previous years.

*3.3 User Interface Requirements*

The user interface has gone through major changes. This was probably the one item that the Title One department was most unsure about. The Title One department has requested that there be a real time update on any scheduling conflicts as the special parameters are being entered. There must be at least three schedule display windows available.

*3.4 System Specifications*

The program must be able to run on Windows or OSX. The Title One department currently uses OSX, but the school district uses both Windows and OSX. Writing the program in Java will allow this to work. Currently, any Intel based Mac will run the program.

**4. System Design**

The project was written entirely in Java.

*4.1 User Interface Design*

The GUI is designed using Netbeans. The GUI allows the operator to create Classrooms and Specials. What is currently scheduled can be seen in any of the three schedule windows. The majority of the GUI centers around making sure that special parameters are entered correctly and the proper combination are entered. Classes and Specials can be deleted. Figure 1 below shows the basic GUI.



Figure 1. Main GUI

The Class Editor can be seen on the right. The class allows for creating, deleting, and editing classes. The lower left box is the Special Editor. All information about the Special is shown here. The operator is shown, in real time, all scheduling conflict as the parameters are edited. All special constraints are enforced in the user interface. For example, it is not possible to have the ending time before the start time. The interface will not allow it. The upper left box is the schedule viewer. The operator may have up to three of these displays open to compare schedules.

*4.2 Classes*

There are five main classes.

4.2.1 GUIMain

This class controls the UI, loading and saving of schedules, and scheduling. All user input and output runs through this class. All Classrooms and Specials are stored in arrays.

4.2.2 Schedule

This class holds all scheduling information. The schedule is a 2D array that holds the ID values of each special scheduled. This is the class that performs all the logic for seeing if slots are open or not.

4.2.3 Classroom

This class holds a Schedule. This class passes and converts all scheduling request between the Schedule and MainGUI classes.

4.2.4 Special

This class holds all the information about specials. This information is accessed from the MainGUI class.

4.2.5 ExcelXML

This class takes an array of classroom schedules and converts them all into an XML format that can be read by Excel. Each classroom is put into it’s own sheet. The GUIMain class sends the schedules to this class for the conversion process.

**5. Software Design Process**

Due to the lack of knowing what they truly wanted, I used the prototyping methodology. I started by developing a user interface to show the Title One department. Took input on it and gathered new requirements. Implemented and removed what they asked. I have repeated this numerous times.

*5.1 Testing and Debugging*

A lot of my time has been testing and debugging the new changes that have been periodically added. It seemed when I would add a new component or change and existing one it would break something that was working perfectly. For example, when I added the feature to show real time conflict updates, initially, nothing would schedule properly. It ended up being that I was modifying the wrong variable. Something simple, but still lost a few hours on it.

The majority of testing was done on the GUI. The GUI is a huge part of this program. Changes in one window affect the others. Several hours of the debugging were used with deleting a classroom that is no longer needed, but has specials scheduled to it. The Excel export was a new item requested last week after the presentation. The Title One department decided they wanted the Specials in the table print out color-coded. That was an item I was not sure how to do in Java and really didn’t have time to explore it since it was so late in the semester. They then wanted to know if I could make the schedules readable in Excel so they could color-code them and then print out. I added that capability and spent about half a day debugging it.

*5.2 Challenges*

The biggest challenge with this project has been working for a client who doesn’t know exactly what they want. The constant changes to requirement and requests have put me way behind. Certain items in the design have been delayed from this version. The automatic scheduler is still in need of work. The color-coding of specials in the table will be added later. Since the automatic scheduler was not a priority, it is not very robust and a very stripped down version of the initial design.

*5.3 Work Break Down*

Figure 2 shows the work breakdown of the project. Most everything is complete with the exception the AI scheduling.



Figure 2

Since the AI Scheduling was not a priority to the Title One department, it was not completed due to other priorities added last week.

**6. Results**

All mandatory requirements were completed on time. Most requested items are in. Adding color to the schedules will be implemented in the next few weeks, barring any new requests. The Automatic scheduler was not fully implemented, as it was a minor requirement by the client. The program has been well tested and debugged. I am satisfied with the way the program has turned out.

**7. Summary and Conclusion**

Having a client that is not sure what they truly want can lead to lots of changes. Last minute requests and requirements can throw you way behind schedule. Since I work in the same building as my clients, I have had constant feedback. This can be a good or bad thing. The client is getting what they want, but all the changes have put the project behind schedule.

This has definitely been a challenging project. I have learned quite a bit about using Netbeans to generate a GUI. I also have learned recently a good bit about XML code for Excel.