CS3716 ASSIGNMENT 3

System Architecture

Monday November 3, 2014

Kayla Blain - 201105103
Christopher Doyle - 200833176
Instructor: Adrien Fiech

A “feature” is defined as anything that your system **needs** to do. Requirements, for system function, can also be considered as features depending on which stakeholder you’re talking to. Some features are however, “architecturally significant design” which is defined as anything that is really important to your system. You can decide this by asking 3 questions at each feature: 1. Is it part of the essence of your system, 2. What does it mean and 3. How do I do it… by answering these three questions we’ve decided on the features that are architecturally significant to our program. Therefore, we have two features lists, one that includes all of the system features and one that separates the key features.

Group Assignment Generator

Features

1. Form for the instructor to initialize the assignment
   - A form where the instructor can initialize the assignment is the most important feature of the system because without the assignment, the other features would be unnecessary.
2. Instructor specifies the maximum number of group members
   - The maximum number of group members is the second most important feature, because you can’t have the computer generate groups when it doesn’t know how to divide the class up.
3. Form for the students to add themselves to the class, by entering their name and student number.
   - This is important because you cannot generate groups without a class list.
4. Generation of groups
   - Our system now has the basic, most important knowledge, to be able to create groups.
5. Allowing the students to input their personal and academic schedules.
   - Finding times for people to hold group meetings may be important for some assignments/projects so having the students add their schedules will allow for more specialized group selection.
6. Allowing the instructor to modify the groups
   - The instructor may want to modify the groups that are randomly generated by the computer. Having the instructor be able to modify the groups will allow him/her not to have to continually ask the computer to randomly generate groups until it produces one that the instructor is satisfied with.
7. A) The maximum number of people which a student is allowed to choose to work with or choose not to work with,
   - A student may have a classmate with whom they’re friends with, or know they work well with (from prior experience), also, the student may be in class with someone they don’t get along with/don’t work well with. Having the professor allow a maximum number of preferred partners may allow for better group selection/better assignment marks and happier students.

   B) A self-evaluation, for the student to fill out, based on skills relevant to the assignment
   - Some assignments may require different skill sets such as Java programming, C/C++ programming, O-O design principles, etc… and delegating groups based on balanced skill set may be necessary, especially in large scale projects.

- Both A) and B) are of the same priority because they aren't necessary for the main goal of the program however, they are both attributes that may be beneficial to group specification but neither takes any real precedence over the other.

8. Allowing the program access to the MUN database to access students class schedule, grades, etc…
   - This minimizes error by allowing to double check that the student belongs to the class, the student entered their class schedule correctly and that the student's grades reflect their self-evaluation of skills.
9. Professor specify the deadline for student input.
   - Specifying a deadline for student input may give those students who take the initiative to fill out their self-evaluation/partners an advantage over those students who don't make the deadline.

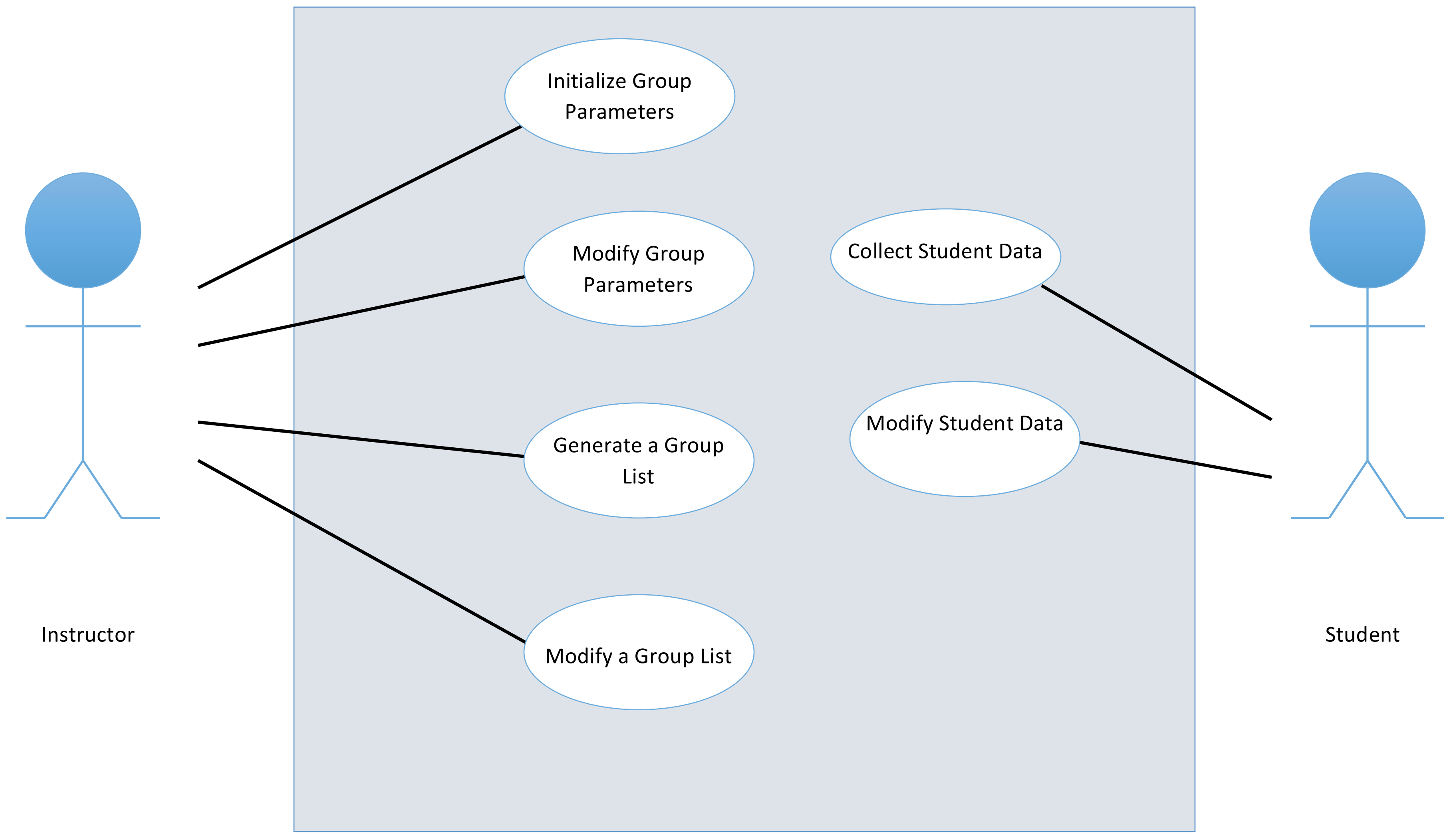
Group Assignment Generator

Key Features

1. Form for the instructor to initialize the assignment
2. Instructor specifies the maximum number of group members
3. Form for the students to add themselves to the class, by entering their name and student number.
4. Generation of groups

The form for the instructor to initialize the assignment in necessary to allow groups to be created in the first place. The groups can range in size from two members to half the class so it is necessary for the instructor to be able to decide on the number of students per group. The form of the students to enter their basic information helps create a useable class list to delegate members for the groups, which the system needs to be able to generate. These four features are essential to basic system function.

Use Case Diagram



Traceability Matrix:

	UC1	UC2	UC3	UC4	UC5	UC6
F1	✓		✓		✓	✓
F2	✓		✓		✓	✓
F3		✓		✓	✓	✓
F4	✓		✓		✓	✓
F5		✓		✓	✓	✓
F6						✓
F7 A	✓	✓	✓	✓	✓	✓
F7 B	✓	✓	✓	✓	✓	✓
F8		✓		✓	✓	✓
F9	✓	✓	✓	✓	✓	✓

Functionality Modules:

1) Instructor Module:

   The instructor would be directed to this section of the system in order to initialize the assignment and set the maximum group size. These are the two essential pieces of information. Without initializing the assignment, the system would not know that it is needed and will remain 'asleep'. It must know that an assignment has begun and that it is required to generate group lists. This parameter is basically the 'on button' of the system. Secondly, the system must know the maximum and minimum size of groups because without this information it might generate groups of uneven size.

   For more refined groups, the instructor can also set pairings of students that he may wish to work together as well as pairings of students that he may wish don't work together. These are optional parts of the Instructor section are not essential to the functionality of the system.

2) Student Module:

   The students are directed to this section of the system are given an opportunity to enter their personal information. The students' names are the most essential part of this section so that when the system generates the groups it is easy to identify which students belong to what groups based on their names.

   Optional information such as student number, preferred partners, and non-preferred partners can also be entered by the students. However, this information is not essential in generating group lists.

3) ClassList Module:

   The class list module is important because it serves as a 'container' for storing all the students. It is important to have this list of students so that as the system populates the individual groups, it knows which students remain. This ensures that no groups are generated that have the multiple copies of the same student. From the class list the system can also gather the size of the class and can begin to process how it will group the students based on the previously entered maxGroupSize and minGroupSize from the Instructor module.

   Again, optional information such as the university name and the assignment name/number can be entered in this classlist section for completeness, but it is also not essential to the functionality of the system.

4) Group Generator Module:

   This module will be responsible for using all the necessary and provided information, and generating the final result – the groups! Since the essence of our system is to break down the class into groups for assignments, this module is very necessary in order for the system to function. This module uses the information provided by the above modules.

Key Decisions:

Our system is organized in a particular fashion, which we felt would be ideal in accomplishing our tasks. We decide it would be good to have a class GroupList which would delegate all the necessary tasks to other classes. We made this decision because we are trying to apply Object-Oriented principles that would enable the system to become reusable and flexible for future use. As you can see in the domain model, the GroupList class has only one parameter, groupList, and its only purpose is to store the list of the groups. These groups are gathered from the Group class where they are stored but are generated by the GroupGenerator class. This design decision was based on keeping information where it is required and no where else – information experts!

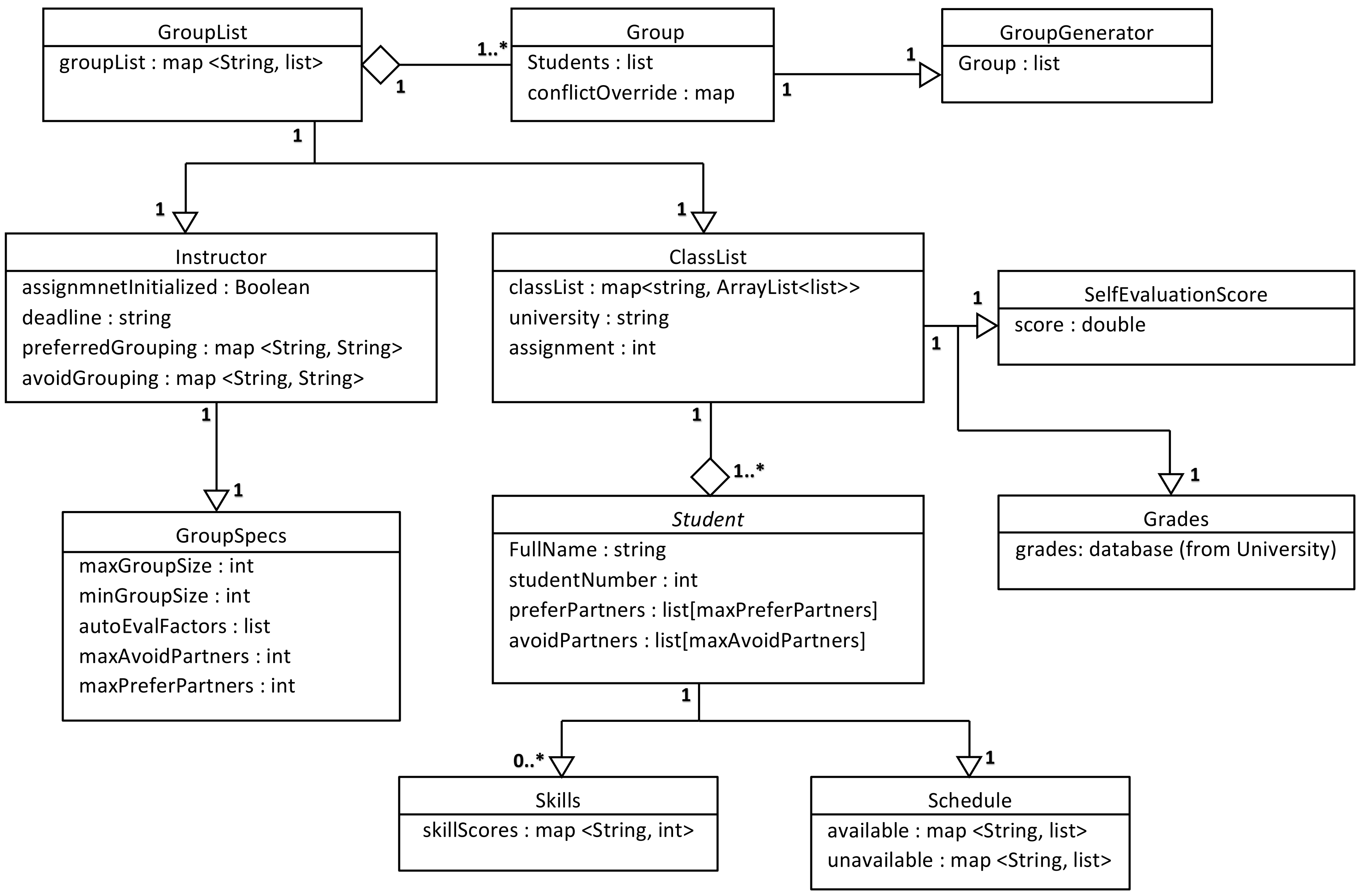
Next we incorporated inheritance into our system. There are just a few key pieces of information (explained in functionality modules) that are required in order for the system to generate the most basic groups. However, there is a lot of information required for the system to generate the type of groups that the instructor may specify. We thought it would be best to categorize all this information and use inheritance to trace the information back to classes that may require it later in the program. We have issued inheritance between the GroupList class and both the Instructor class and the ClassList class. This way the GroupList class can obtain the necessary group constraints outlined by the instructor as well as a list of the students in the class.

From there we have more inheritance from Instructor to GroupSpecs. This decisions was based on the instructor not needing to set all these parameters upon the initial start of the system. Also they are not essential to the functionality. They are additional options which the instructor may or may not wish to use.

The ClassList class inherits from the Student class because it required the different student objects in order to populate its list of students in the class. We debated making ClassList an abstract class but decide it against it because not all the classes that extend it have common fields. The Student class has been made into an abstract class for this reason.

We have essentially used delegation and inheritance within our classes in order to make our simple flexible and maintainable in the future. The decisions we made were based upon the individual needs that may arise from the instructor, and trying to create an object-oriented system that will serve its purpose for a long time.

Updated Domain Model


GroupList
groupList : map <String, list>
Group
Students : list
conflictOverride : map
GroupGenerator
Group : list
Instructor
assignmnetInitialized : Boolean
deadline : string
preferredGrouping : map <String, String>
avoidGrouping : map <String, String>
ClassList
classList : map<string, ArrayList<list>>
university : string
assignment : int
SelfEvaluationScore
score : double
GroupSpecs
maxGroupSize : int
minGroupSize : int
autoEvalFactors : list
maxAvoidPartners : int
maxPreferPartners : int
Student
FullName : string
studentNumber : int
preferPartners : list[maxPreferPartners]
avoidPartners : list[maxAvoidPartners]
Grades
grades: database (from University)
Skills
skillScores : map <String, int>
Schedule
available : map <String, list>
unavailable : map <String, list>
1..*
1
0..*