TimeTabling System Information

Michal Lange

August 2019

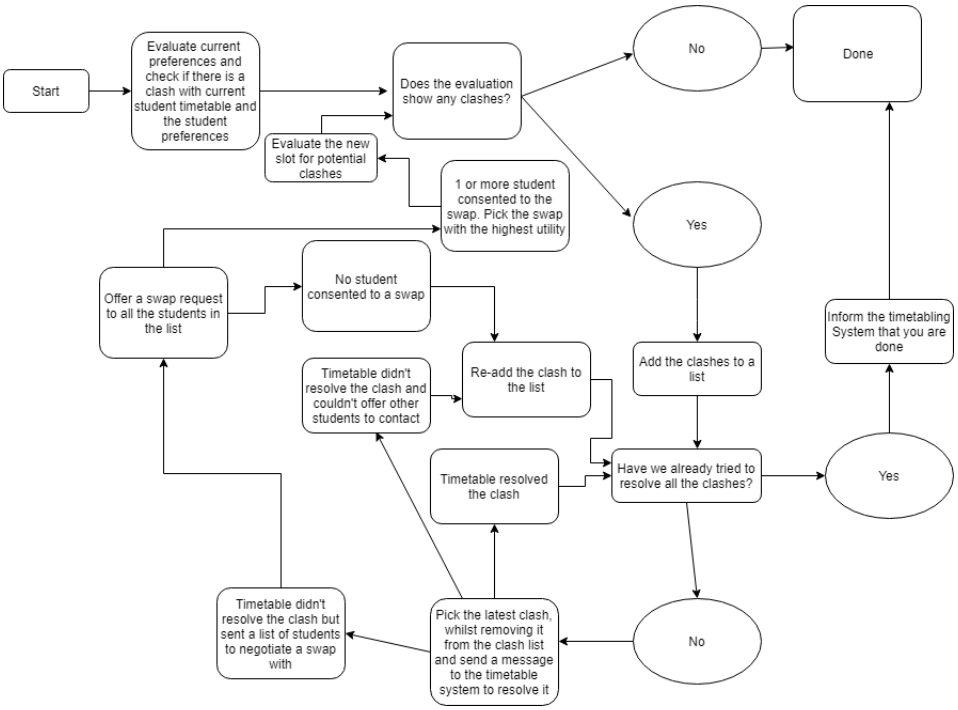
1 Introduction

The aim of this paper is to give a brief summary of the current state and functionality of the Time Tabling System.

2 Brief Description

The project aimed to create an optimal timetabling system, which took into consideration the preferences of students being allocated to different time slots. In real life, students would be aware of their preferences and would negotiate with their teachers/colleagues for slot swaps. It is possible to emulate this type of negotiation through the use of agent programming, which allows to create autonomous and self-interested agents representing each of the students.

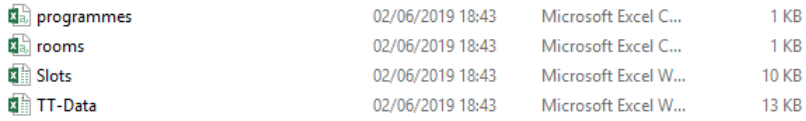
A survey is read in to the program which represents the preferences of the students. Currently each student who has completed the survey can choose up to 5 slots which they deem unacceptable and any number of slots that they deem awkward. Following that agents are instantiated, one for the timetabling system responsible for student swaps and then any number of additional agents equal to the number of students. Below is representation of how a given agent functions:



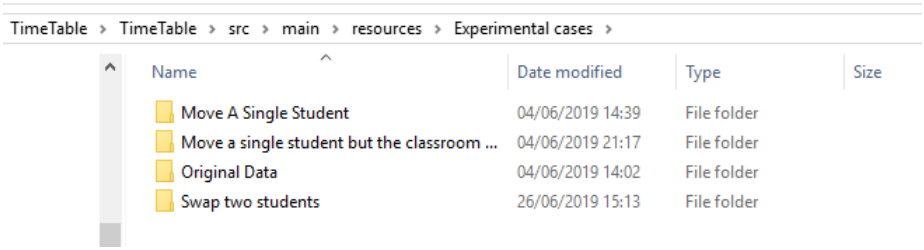
There are other functions that a student can perform, namely receiving swap requests from other students and evaluating them, this will be discussed in detail in further sections.

3 How to run tests?

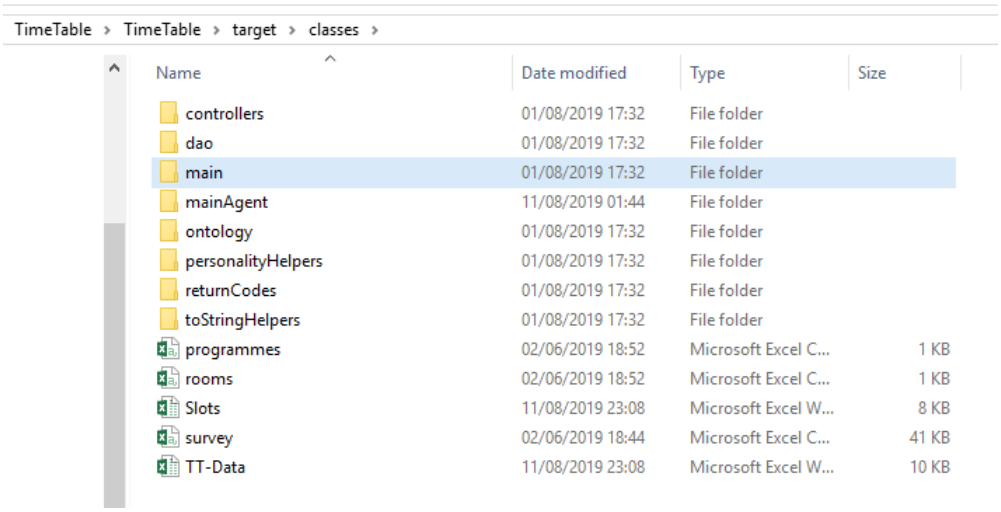
This is quick subsection about running different use cases and tests. For this you will need 4 excel files necessary for running the simulation:



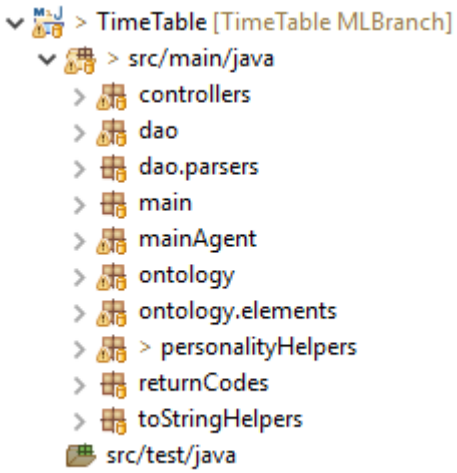
and preferably store them in the Experimental cases sub-folder:



to run the program on a specific use case move the 4 excel files into the following folder:



4 Understanding the project structure when you pull it from github



1. controllers - Contains classes/functions pertaining timetable manipulation
2. dao - responsible primarily for reading/querying excel data
3. dao.parsers - parsers for dao
4. main - an old/deprecated version of main, used for testing purposes
5. mainAgent - Contains the necessary files for the agent program - future work should be primarily done in this package

6. ontology elements - all the elements that make up the environment (Student, Event, Slots etc...)

7. personality helpers - responsible for reading in and generating personalities for the students

5 Understanding how utility is calculated

When a student receives a request from another student for a swap, that student has to give consent by assessing the utility of the swap, the formula is as follows:

Formula for student receiving an offer: CurrentEvent(Awkward) = +1
CurrentEvent(Unacceptable) = +2
CurrentEvent(Neutral) = 0
IncomingEvent(Awkward) = -1
IncomingEvent(Unacceptable) = -2
IncomingEvent(Neutral) = 0
IncomingEvent(AwkwardForOtherStudent) = 0
IncomingEvent(UnacceptableForOtherStudent) = +1
CurrentEvent + IncomingEvent + IncomingEventForOtherStudent greater or equal 0 Then Accept

Example 1: Harry asks John to swap events with him. John knows that the event that Harry is leaving is important to him (+1) and the event that Harry wants is awkward for John (+1) and he feels neutral about the event he is going to receive (+0). The calculated utility is +2 hence he gives consent to the swap.

Example 2: Harry asks John for another swap. The event is just awkward for Harry (+0), and the event John would leave is awkward (+1), however the event that John would receive is unacceptable for him (-2). The final utility calculated is -1 and John does not give consent for the swap.