UWB CSS Scheduling Simulation

## CSS 458

## Spring 2015

## Christopher Tran Jason Dailey Kevin Parker

# Program Description

This simulation program is designed to use historic data from the time schedule to predict future schedules and professors necessary to fulfil the course schedule for the CSS program at UWB.

# Program Use

## Reading Historic Data

While the simulation is not required to read historic data, the simulation would be ineffective unless data is provided.

### Time Schedule

Simulation data is expected to come from the UW Bothell time Schedule. Example files have been provided in the simulation.

### TSV Format of one quarter

Course Rows

1. Course number with CSS Prefix
2. Course Name

Course Instance Row (Every course should have one or more Course Instance Row)

1. Blank
2. SLN Number
3. Section Code
4. Credits
5. Days formatted as abbreviations(M,T,W,TH,F,S,SU) and are comma delimited if more than one day is identified
6. Time of the course in military(24H) formatted as Start-End
7. Location of the course to be taught
8. The instructor of the course
9. Enrollment information formatted as RegisteredStudents/StudentCapacity

Example Course Entry

CSS301 TECHNICAL WRITING

12453 A 5 M,W 1745-1945 CC1 041 "ANDERSON,LAURIE JOY" 16/30

12454 B 5 T,Th 1315-1515 UW1 031 "MAYER,ROBERT" 29/30

12455 C 5 M,W 1530-1730 CC1 011 "ANDERSON,LAURIE JOY" 16/30

## Starting Schedule

The file for the beginning schedule is expected to be within a CSV with the following format

1. Course
   1. Course number
2. Time
   1. Start-End
   2. Military time(24H)
3. Day
   1. Abbreviated Day Names
   2. M T W TH F S SU
   3. Delimited with forward slash /
4. Cap
   1. Capacity of the course
5. Quarter
   1. Quarter of the course
   2. autumn, winter, spring, summer
6. Expertise
   1. Category of the course
   2. Programming, Teaching, Hardware, Writing, Cybersecurity, etc

## Professors

The file for the instructors is expected to be within a CSV with the following format

1. Name
   1. Name of the instructor
2. FullTime
   1. Y if the faculty member is full-time
   2. N if the faculty member is part-time
3. Classes
   1. Sum of the allocation of classes that the instructor is capable of teaching
   2. 1 for a regular course
   3. 0.5 for a Teaching course
4. StudentsAdvised
   1. Number of students the faculty is advising for capstone
5. Expertise1/Expertise2/Expertise3
   1. List of expertise the instructor is capable of teaching delimited with forward slashes /

## Simulation Execution

Maindriver.py contains all necessary example code for execution that will be described.

### Historic Data Reading

Reading of historic data is assumed to have TSV files separated by quarter. To add historic data into the course history, use the readCourses() method. This method takes three parameters. First, is the location of the file, second is the year as an integer, third is the quarter as a string. The quarter should be all lowercase using the names “autumn”, “winter”, “spring”, “summer”.

Example function calls for a year

# Create Course History and read all data files  
history = CourseHistory()  
# read in year 2010  
history.readCourses(".\\TimeSchedules\\2010-01-Fall-CSS.tsv", 2010, "autumn")  
history.readCourses(".\\TimeSchedules\\2010-02-Spring-CSS.tsv", 2010, "spring")  
history.readCourses(".\\TimeSchedules\\2010-03-Winter-CSS.tsv", 2010, "winter")  
history.readCourses(".\\TimeSchedules\\2010-04-Summer-CSS.tsv", 2010, "summer")

### Importing Starting Schedule and Professors

The provided courses and faculty files are expected to be given in .csv files with columns as defined above. There is class defined (dataImport.py) that provides methods to read the csv files and return them as lists of course or faculty objects. For the courses, the call importCourses = dataImport.importCourses() is made. For the faculty, the call importProfessors = dataImport.importFaculty() is made. This also assumes the files are at the same directory level and are titled courses.csv and faculty.csv respectively.

If there is ever a need to add additional entries (new courses or new faculty) into the starting schedule or professors, all that is required is to add additional rows in either file, filling out the expected data columns.

### Simulation

Within maindriver, modify the simStart variable to the beginning of the year to begin the simulation.

Within maindriver, modify the simiterations variable to run the simulation the number of years desired

In our example we, our simStart variable is 2015, and simiterations is 5. This sets our simulation to run and give us schedules for 2016 – 2020.

After your variables are set, run the maindriver.py file. The driver is composed to print data for each year in the range of simulated years. For each of these years, the program will print out the professors with their assigned courses for the given year, print out the remaining unassigned courses, print out the annual schedule, and then print out the annual enrollment history. The annual enrollment history will show all history data that was imported at the beginning of the simulation, as well as our calculated enrollment, capacity, and occupancy for each simulated year. This will show based on our assumptions, how the growth of the program is expected to go.