IRSTI 14.01.85

M. Zhuniskhanov[1], R. Suliyev[2]
[1,2]Suleyman Demirel University, Kaskelen, Kazakhstan

IMPLEMENTING COURSE SCHEDULE GENERATION APPLICATION FOR UNIVERSITY

Abstract. This work is about researching and implementing Course Schedule Generation Application for university. She compares other similar applications in this area and explores important points. Through UML diagrams, important concepts and development progress are explained. Screenshots of using the application will also be provided. Dataset, configuration and rules explained. In the end, we will talk about the importance of such an application and facilitate planning for the university.

Keywords: course, scheduling, university, rules, implement.

***

Аннотация. Эта работа посвящена исследованию и внедрению приложения для составления расписания курсов для университета. Она сравнивает другие подобные приложения в этой области и исследует важные моменты. С помощью диаграмм UML объясняются важные концепции и ход разработки. Также будут предоставлены скриншоты использования приложения. Набор данных, конфигурация и правила объяснены. В конце мы поговорим о важности такого приложения и облегчение планирование для университета.

Ключевые слова: курс, планирование, университет, правила, осуществлять.

***

Аңдатпа. Бұл жұмыс университетің курстар кестесін жасауды зерттеу және қосымша енгізу үшін арналған. Ол басқа да осы саладағы осындай бағдарламаларды салыстырады, маңызды жерлерлерін зерттейді. UML диаграммалар көмегімен маңызды тұжырымдамалар түсіндіріледі және әзірлеу барысы айқындалады. Сондай-ақ, қосымшалар пайдалану скриншоттары ұсынылатын болады. Деректер жиынтығы, орнату ережесі түсіндеріледі. Соңында біз қосымшалар маңыздылығы мен университет үшін жоспарлау жеңілдетілетіндігі туралы әңгімелейді.

Түйін сөздер: курс, жоспарлау, университет, ереже.

Introduction

A university course timetabling refers to the allocation of courses offered to fixed timeslots and rooms with respect to the number of lecturers available at that time [1]. Optimal Course schedule makes staff and students feel satisfactory and saved time for administrators. The application is designed to keep records of course scheduling to compare and choose most suitable

The application uses open source systems. Also it is user friendly and easily configurable.

Aim

Aim is to create application using optaplanner and drools rule engine, which allows set constraints in flexible manner. It gives options to use different settings to select algorithms, in a selected time or condition stop program execution. Such application provides necessary information for user to select best options, so that he can iterate the cycle changing settings and executing to obtain optimal result. Optimal result motivates staff and students to focus only on teaching and studying courses.

Technologies currently in use

For now, almost every university develop their own system or adopt already developed system from another universities. One of the examples of university timetabling software is the UTTS system [2] developed at the National University of Singapore. There are also open source libraries for further development and integration. Unitime [3] and Optaplanner [4] are some examples of them. They are no only focused on course scheduling also they solve other metaheuristic problems.

Keeping in mind the software integration tools and the dataset and infrastructure also impacts the decision which software to choose. In this project we choose optaplanner as base system for further integration.

Project development

In UML diagram you can see an architecture and logic of project. According to sequential UML diagram user have option to configure settings. First activity is main page. Main page has fields to be filled, the execution time and unimproved count of steps, they affect the stop condition of program. Also some soft constraint can be added to program like some lecturers can not teach in certain time. After user can press start button to start course scheduling execution under the hood. User entered data saved to database.

Second activity is maintained by program it collects data from database, the settings applied to program and data rearrangement occurs using algorithms. On every step rules check the satisfaction over constraints. After time timeout or unimproved step count exceeds limit program stop and final condition saved to database. UML diagram shown in Figure 1.

In last step user can view result if it not satisfactory then restarts course scheduling till optimal result obtained.

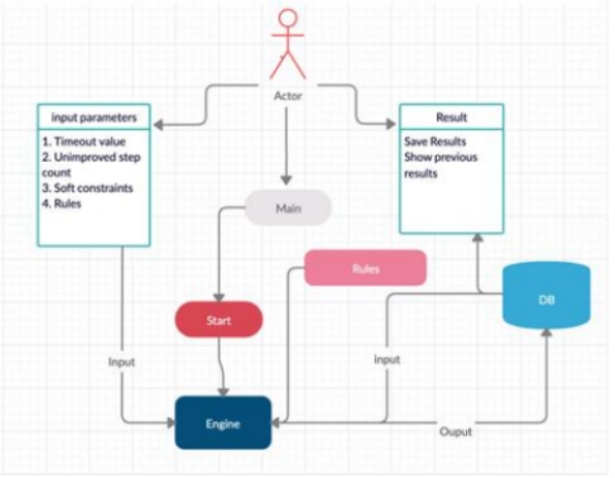

Figure 1. Use-case UML diagram of project

Use-case diagram shows how engine works when user presses start. Engine has algorithms to efficiently find solution and rules to structure data in needed manner.

Intermediate result can be viewed by clicking result button. Also constraint satisfaction score also shown. If its satisfactory enough then course generation can be interrupted.

In detail program uses three types of data, the main data are the courses themselves and its related data as teachers, classrooms, time periods. Their bunch form the result. You can see a bunch of data in figure 2.

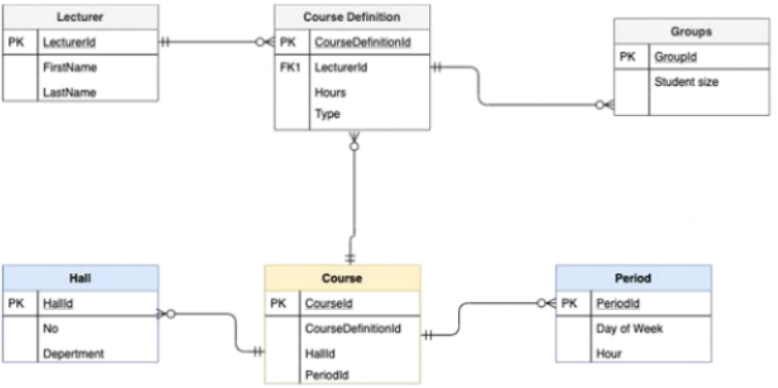

Figure 2. Course related objects

Settings can also be attributed to one type of data. In the java version of optaplanner it is possible to make settings through an xml file. the file contains a setting for selecting an algorithm for the first data filling in order to facilitate the work of the main algorithm for finding the optimal solution. The second item is the settings for the main algorithm, along with it there are different settings

for the action. Actions are when the classrooms are replaced at every step or the time the course begins to check for compliance with the rules. An example setup is shown in figure 3.

The third kind of data is the rule. From the rules describe the required location and the duration of the courses. The optimal result, she meets all the rules. Rules can be hard and soft constraints. Hard constraints are those without which it is impossible to conduct some kind of course. And soft constraint are requirements that it is desirable to observe, but if they interfere with hard constraints then they can be overdone.

```
<?xml version="1.0" encoding="UTF-8" ?>
<solver>
    <environmentMode>REPRODUCIBLE</environmentMode>
    <scanAnnotatedClasses />
    <scoreDirectorFactory>
        <ksessionName>scheduleKsession</ksessionName>
    </scoreDirectorFactory>
    <termination>
        <terminationCompositionStyle>OR</terminationCompositionStyle>
        <bestScoreLimit>0hard/0soft</bestScoreLimit>
        <minutesSpentLimit>5</minutesSpentLimit>
    </termination>
    <constructionHeuristic>
        <constructionHeuristicType>FIRST_FIT_DECREASING</constructionHeuristicType>
    </constructionHeuristic>
    <localSearch>
        <unionMoveSelector>
            <changeMoveSelector/>
            <swapMoveSelector>
                <filterClass>kz.sdu.schedule.solver.DifferentCourseSwapMoveFilter</filterClass>
            </swapMoveSelector>
        </unionMoveSelector>
        <acceptor>
            <lateAcceptanceSize>600</lateAcceptanceSize>
        </acceptor>
        <forager>
            <acceptedCountLimit>4</acceptedCountLimit>
        </forager>
    </localSearch>
</solver>
```

Figure 3. optaplanner course scheduling configuration

In our program, we use the drools rules program, which is quite autonomous and works more efficiently than the rules written in java code. An example of the rules is shown in figure 4.

```
import org.optaplanner.core.api.score.buildin.hardsoft.HardSoftScoreHolder

import kz.sdu.schedule.planning.Schedule
import kz.sdu.schedule.planning.Course
import kz.sdu.schedule.planning.Date
import kz.sdu.schedule.planning.Lesson
import kz.sdu.schedule.planning.Room
import kz.sdu.schedule.solver.CourseConflict

global HardSoftScoreHolder scoreHolder;

rule "roomOccupancy"
    when
        Lesson($leftId : id, date != null, $weekDay : date.weekDay, $hour : date.hour, room != null, $room : room)
        not Lesson(date.weekDay == $weekDay, date.hour == $hour, room == $room, id < $leftId)
        Lesson(date.weekDay == $weekDay, date.hour == $hour, room == $room, id > $leftId, $rightId : id)
    then
        scoreHolder.addHardConstraintMatch(kcontext, -1);
end

rule "lessonsWithSameTeacherInSameTime"
    when
        Lesson($id: id, $roomId : room!.id, $teacherId: course!.teacher!.id)
        Lesson(id > $id, room!=null, room.id == $roomId, course.teacher.id == $teacherId)
    then
        scoreHolder.addHardConstraintMatch(kcontext, -2);
end

rule "conflictingLessonsDifferentCourseInSamePeriod"
    when
        $courseConflict : CourseConflict($leftCourse : leftCourse, $rightCourse : rightCourse)
        $leftLesson : Lesson(course == $leftCourse, $weekDay: date!.weekDay, $hour: date!.hour, date != null)
        $rightLesson : Lesson(course == $rightCourse, date!.weekDay == $weekDay, date!.hour == $hour, this != $leftLesson)
    then
        scoreHolder.addHardConstraintMatch(kcontext, - $courseConflict.getConflictCount());
end
```

Figure 4. Drools rules

After launch, the system produces a result that can be used for the school year. For a good result, it is advisable to run a system with different algorithms and with a period of not less than an hour. An example result obtained for our data is given below.

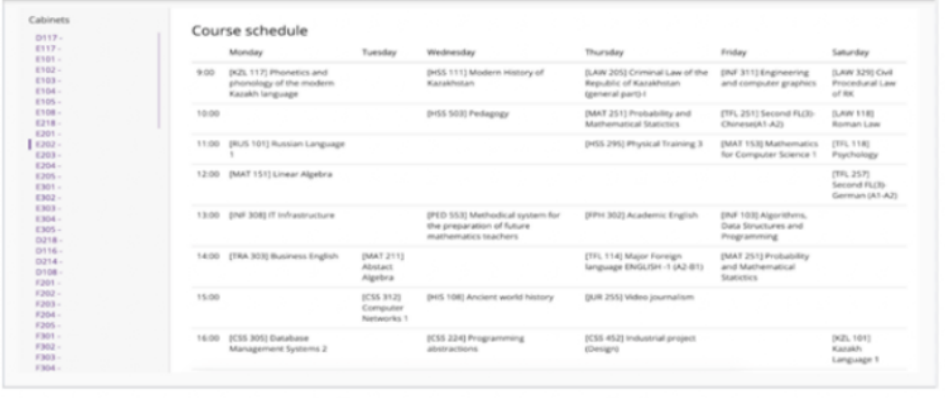

Figure 5. Course scheduling result

Conclusion

The goal of this paper was to develop an application for course scheduling for universities. The paper itself provides the project overview. It starts with some background information about application development and presents features of course scheduling application from the point of view of developer.

Then it moves on the description of the design of the application, providing some short justification for application design decision taken during development. Then the paper describes overall architecture of the application, covers some of the technically challenging or otherwise interesting features that have been implemented.

I hope this application will be useful for people who responsible for course scheduling in universities.

References

1 Wan Muhamad, Wan Zuki Azman & Adnan, Farah & Yahya, Zainor Ridzuan & Junoh, Ahmad & Zakaria, Mohd. (2018). Solving university course timetabling problems using FET software. AIP Conference Proceedings. 2013. 020052. 10.1063/1.5054251. p.1.

2 Andrew L, Ang J. Chin, Ho W. Kit, and Oon W. Chong, A Campus-Wide University Examination Timetabling Application, American Association for Artificial Intelligence, 2-3, 2000. p.1.

3 Comprehensive University Timetabling System. URL: https://www.unitime.org/

4 What is OptaPlanner? URL: https://www.optaplanner.org/