**IF4150 Domain Specific Software Engineering**

**DSL CLASS SCHEDULING**



by:

Roselina Pradjanata 13515034

Erick Wijaya 13515057

Kezia Suhendra 13515063

Rachel Sidney Devianti 13515124

**INFORMATICS STUDY PROGRAM**

**SCHOOL OF ELECTRICAL ENGINEERING AND INFORMATICS**

**BANDUNG INSTITUTE OF TECHNOLOGY**

**BANDUNG**

**2018**

# **Table of Revisions**

|  |  |  |
| --- | --- | --- |
| **No** | **Revision** | **Page** |
| 1 | Add ontology for class scheduling. | 12 |
| Update grammar’s syntax to add axiom, create, update, delete, and display entity in the program. | 15 |
| Add descriptions for each command in grammar. | 17 |

# **Table of Contents**

[**Table of Revisions**](#_p6qatjidrqhw) **2**

[**Table of Contents**](#_5mriv9xbe8x2) **3**

[**Domain Analysis**](#_bko5sulcg9dk) **4**

[Description](#_es89dxanmus6) 4

[Context](#_es89dxanmus6) 4

[Domain Components](#_es89dxanmus6) 4

[Entity](#_xoao4tglx073) 4

[Function](#_v5usnbt9o2w) 5

[Behaviour](#_rb1ntp9s011v) 6

[Event](#_h3a3foyusx4t) 7

[Axiom](#_f198shvystzn) 8

[Stakeholders](#_uuq3t4xscz74) 8

[Domain Model](#_xzxsy41m63kn) 9

[Structural Model (Class Diagram)](#_q2uwmi3m05qj) 9

[Behavioral Model (Use Case Diagram)](#_xll2mjn8nuar) 10

[Ontology](#_fkndj1or0sdw) 12

[**Supported Use Cases**](#_t0p2uqrbl6j8) **14**

[**Language Grammar**](#_y7nvhzeoab97) **15**

[Syntax](#_9gn7ltvl9wh7) 15

[Description](#_96zy3tvbm6nu) 17

[Lecturer](#_8sdope9vdgrt) 17

[Course](#_l1hrlecz6fg1) 18

[Classroom](#_ps8654fbnrz3) 18

[Class](#_furd12qbeo93) 19

[Display](#_gsrlou68lsdq) 19

[Axiom](#_un501pqmunzr) 20

[Examples](#_1wvr12cvdz32) 22

# **Domain Analysis**

## **Description**

In a university, a class schedule is a list that shows the times in a week at which particular subjects are taught. Usually, lectures are held between Monday to Friday from 7 AM to 6 PM and the duration for a time slot is an hour. Lecturers that teach in university specialize in certain courses and they have preferred times to teach. Each course has facility needs, such as computers, chairs, white board, LCD, and etc. A course can have more than one classes, each class usually do not have two or more time slots on the same day.

## **Context**

Following are the context used in the development of this DSL for class scheduling:

* The scheduling is for classes in university.
* Total time duration of a class has to be equal with the number of credits.
* Longest duration of a class on a given day has to be no more than n (default = 2) hours.

## **Domain Components**

### **Entity**

|  |  |  |
| --- | --- | --- |
| **No.** | **Entity** | **Description** |
| 1. | Classroom | A room, typically in a school, in which a class of students is taught. |
| 2. | Facility | A place, amenity, or piece of equipment provided for a particular purpose. |
| **No.** | **Entity** | **Description** |
| 3. | Course | A series of lectures or lessons in a particular subject, typically leading to a qualification. |
| 4. | Class | A group of students who are taught together in a classroom. |
| 5. | Lecturer | A person who gives lectures, especially as a profession. |

### **Function**

|  |  |  |  |
| --- | --- | --- | --- |
| **No.** | **Entity** | **Function** | **Description** |
| 1. | Classroom | Add Classroom | Add a new classroom to be included in the class scheduling. |
| Get Classrooms | Get existing classrooms that have been defined before. |
| Get Facilities | Get facilities that exist in a classroom. |
| 2. | Facility | Add Facility | Add a new facility that can be used by courses. |
| Get Facilities | Get list of existing facilities. |
| 3. | Course | Add Course | Add a new course to be included in the class scheduling. |
| Get Courses | Get list of existing courses. |
| Get Classes | Get classes that exist for a certain course. |
| Get Lecturers | Get list of lecturers that teach this course. |
| **No.** | **Entity** | **Function** | **Description** |
| 4. | Class | Add Class | Add a new class for a specific course. |
| Get Time Slots | Get existing time slots of a specific class. |
| Get Classes | Get list of existing classes. |
| Get Course | Get parent course of a certain class. |
| Get Lecturer | Get lecturer of a specific class. |
| 5. | Lecturer | Add Lecturer | Add a new lecturer who can teach some courses. |
| Get Lecturers | Get list of existing lecturers. |
| Get Preferred Time | Get list of preferred times for a certain lecturer. |
| Get Classes | Get list of classes that are taught by a certain lecturer. |

### **Behaviour**

|  |  |
| --- | --- |
| **No.** | **Behaviour** |
| 1. | Lecturers often have to reschedule their time slot because of other matters. |
| 2. | Lecturers often have facility preferences while teaching. |
| 3. | Lecturers often have time preferences. |
| 4. | Available time slots often do not fit to lecturers’ preferences. |

### 

### **Event**

|  |  |
| --- | --- |
| **No.** | **Event** |
| 1. | Class gets scheduled. |
| 2. | Class gets rescheduled. |
| 3. | Lecturer asks to change schedule for a certain course. |
| 4. | Lecturer is added. |
| 5. | Lecturer’s preferred times are added. |
| 6. | Classroom is added. |
| 7. | Course is added. |
| 8. | Course’s facilities are added. |
| 9. | Course’s constraints are added (courses that are not allowed to happen at the same time with the course). |
| 10. | Class is added. |
| 11. | Class’ lecturer is added. |
| 12. | Lecturer is updated. |
| 13. | Lecturer’s preferred times are updated. |
| 14. | Classroom is updated. |
| 15. | Course is updated. |
| 16. | Course’s facilities are updated. |
| 17. | Course’s constraints are updated (courses that are not allowed to happen at the same time with the course). |
| 18. | Class is updated. |
| 19. | Class’ lecturer is updated. |
| 20. | Lecturer is deleted. |
| 21. | Lecturer’s preferred times are deleted. |
| **No.** | **Event** |
| 22. | Classroom is deleted. |
| 23. | Course is deleted. |
| 24. | Course’s facilities are deleted. |
| 25. | Course’s constraints are deleted (courses that are not allowed to happen at the same time with the course). |
| 26. | Class is deleted. |
| 27. | Class’ lecturer is deleted. |

### **Axiom**

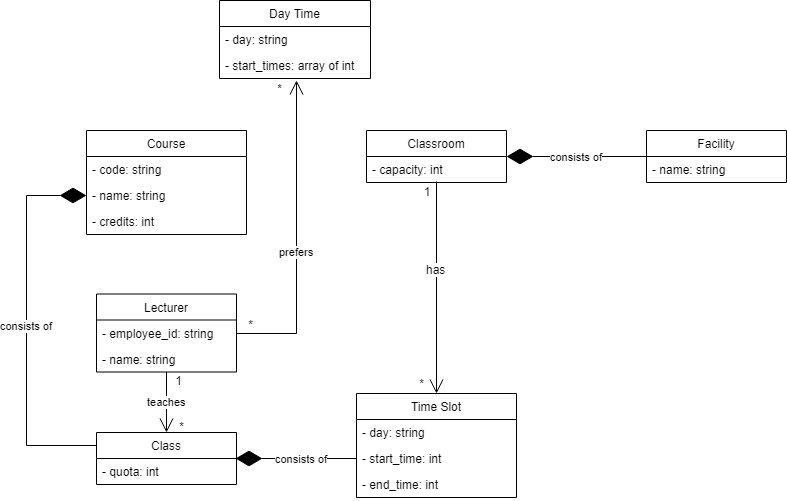
|  |  |
| --- | --- |
| **No.** | **Axiom** |
| 1. | The capacity of a classroom must be bigger than the maximum quota of the class that occupies it. |
| 2. | A classroom can only hold single class in specific time slot. |
| 3. | Courses that are mandatory and located in the same semester tend to not happen in the same time. |

### **Stakeholders**

|  |  |
| --- | --- |
| **No.** | **Stakeholders** |
| 1. | Students |
| 2. | Lecturers |
| 3. | Faculty Administrators |
| 4. | Technical Supports |

## **Domain Model**

### **Structural Model (Class Diagram)**

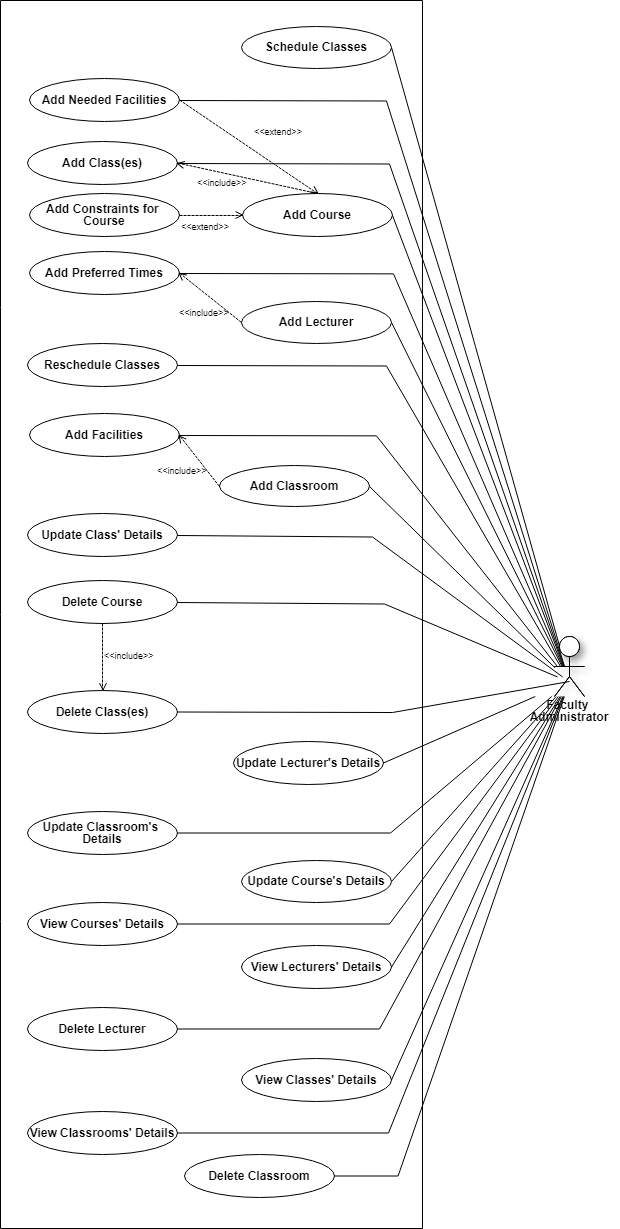


A course consists of one or more classes. A class consists of one or more time slots. Each classroom can have zero or more time slots. A lecturer can have preferred times to teach and teach zero or more classes. Other than that, a classroom can consist of facilities, such as LCD, chairs, white board, black board, and etc and have zero or more time slots.

Notes:

* Time Slot is a specific time when a specific class happen in a day.
* Day Time is lecturer’s preferred times to teach in a day.

### **Behavioral Model (Use Case Diagram)**

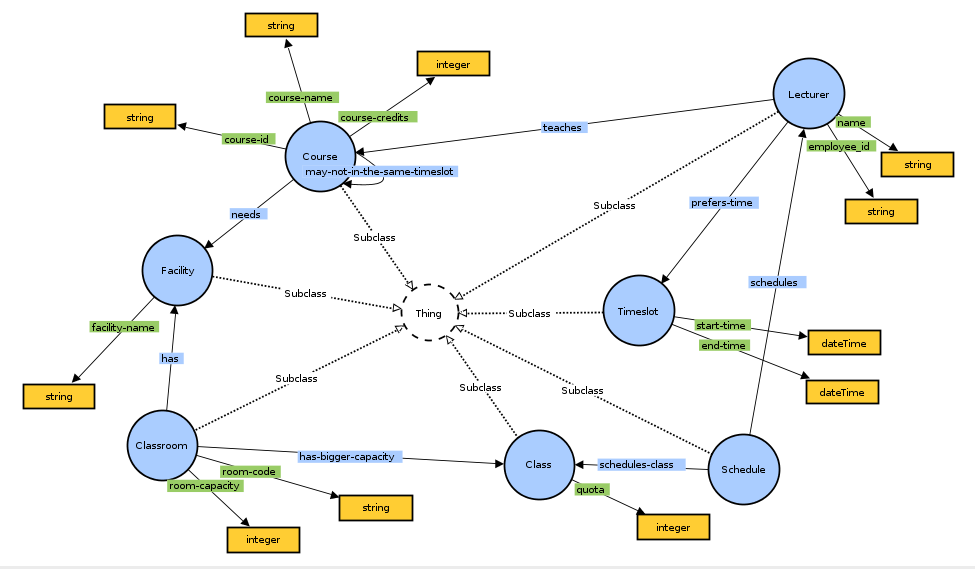


Faculty administrator (admin) has many use cases. The most important use cases are admin can schedule and reschedule classes. Admin can add and delete a lecturer, classroom, course, and class. Admin can update lecturer’s details, classroom’s details, course’s details, and class’ details. Admin can also view lecturers’ details, classrooms’ details, courses’ details, and classes’ details. When adding a course, admin has to add class(es) for the course. He/she can also add needed facilities and constraints for the course. However, when admin deletes a course, admin will also automatically delete its associated classes. Other than that, when admin add a lecturer, admin also has to add its preferred time(s) to teach. Same with classroom, when adding a new classroom admin has to define its facilities (chair, white board, and etc).

Although there are update, delete, view, and add use cases for classes, classrooms, lecturers, and courses, the developers only implement add and view use cases for each objects and also schedule and reschedule classes. If the admin wants to change the detail informations of one of the objects (update or delete), he/she can change it directly in the resource (data.txt) file.

### 

### **Ontology**



Ontology for class scheduling can be seen from the diagram above. In class scheduling, there are seven main objects such as class, course, classroom, facility, schedule, timeslot, and lecturer. Each object has its own attributes, except for schedule. Class has quota (positive integer), classroom has room capacity (positive integer) and room code (string), lecturer has employee id (string) and name (string), course has course id (string), course name (string), and course credits (positive integer), facility has facility name (string), and last but not least time slot has start time (date) and end time (date).

Every course has one or more classes, constraints, and also facility needs. A classroom has facilities inside it. A classroom that has been assigned for certain course, must have fulfilled the facility needs of the course. A classroom must have capacity that is bigger than the class that is held in it. Every lecturer has his/her preferred times to do a lecture, he/she will not do a lecture outside of his/her preferred times. Lecturer can also teaches one or more courses. Schedule contains classes and also lecturers that have been assigned to teach certain class(es).

# **Supported Use Cases**

Below are supported use cases from our DSL. The complete use case diagram can be viewed in [behavioral diagram](#_xll2mjn8nuar).

**Classroom Configuration:**

“Add Classroom” use case implements classroom configuration that has name, capacity, and also list of facilities that the room provides.

**Class Requirements:**

“Add Course” use case and “Add Class” use case implements class requirements which contain what facilities that classes for a certain course must have and what courses that must not happen in the same time with a certain course.

**Lecturer Availability:**

Lecturer availability is defined as “Add Preferred Times” use case which contains list of key-value pairs (day as key and hour as value) that define the time when the lecturer is available to give a lecture.

**Schedule Preferences and Constraints:**

* Lecturer can choose preferred time to teach.
* Classroom capacity has to be bigger than the maximum quota of the class that uses it.
* A class cannot have more than 2 time slot on the same day.
* Some courses have facility needs.
* Some courses cannot be held in the same time.

**Schedule Class:**

Schedule class is implemented as class that has been automatically scheduled by the program (“Schedule Class” use case). Schedule class consists of class assigned in valid classroom and in valid time slot.

# 

# **Language Grammar**

## **Syntax**

Below are the syntax of the grammar that the developers use to create objects for class scheduling.

|  |
| --- |
| */\* LEXER \*/*  lexer grammar ClassSchedulingLexer;  fragment A : [aA]; fragment B : [bB]; fragment C : [cC]; fragment D : [dD]; fragment E : [eE]; fragment F : [fF]; fragment G : [gG]; fragment H : [hH]; fragment I : [iI]; fragment J : [jJ]; fragment K : [kK]; fragment L : [lL]; fragment M : [mM]; fragment N : [nN]; fragment O : [oO]; fragment P : [pP]; fragment Q : [qQ]; fragment R : [rR]; fragment S : [sS]; fragment T : [tT]; fragment U : [uU]; fragment V : [vV]; fragment W : [wW]; fragment X : [xX]; fragment Y : [yY]; fragment Z : [zZ]; COLON : ':';  */\* Command \*/* COURSE : C O U R S E; FACILITY : F A C I L I T Y; CLASSROOM : C L A S S R O O M; CLASS : C L A S S; TIMESLOT : T I M E S L O T; LECTURER : L E C T U R E R; AXIOM : A X I O M;  DISPLAY : D I S P L A Y;  */\* Operator \*/* EQ : '='; OPEN\_PARENTHESIS : '('; |

|  |
| --- |
| */\* PARSER \*/*  parser grammar ClassSchedulingParser;  options {tokenVocab=ClassSchedulingLexer;}  key  : WORD;  value  : (WORD)+ | (map)+;  map  : key COLON value;  line  : (TAB)\* map;  createParam  : line+;  *// Course* createCourse  : COURSE COLON createParam;  *// Class* createClass  : CLASS COLON createParam;  *// Classroom* createClassroom  : CLASSROOM COLON createParam;  *// Lecturer* createLecturer  : LECTURER COLON createParam;  *// Display*  createDisplay  : DISPLAY COLON createParam;  *// Axiom*  createAxiom  : AXIOM COLON createParam;  file  : (createCourse | createClass | createClassroom | createLecturer | createDisplay  | createAxiom)\*; |

file function will parse the input objects into a course (createCourse) if the first word equals to “course”, a class (createClass) if the first word equals to “class”, a classroom (createClassroom) if the first word equals to “classroom”, a lecturer (createLecturer) if the first word equals to “lecturer”, an axiom (createAxiom) if the first word equals to “axiom”, or a display (createDisplay) if the first word equals to “display”. All of the words are case insensitive. After the name of the object, the grammars ask for a colon (:) input to differentiate between key and value. After a colon (:), the parser will parse the attributes of the object using createParam function. The format of the attributes is key-value pairs, the value of a key can be either words (terminal node) or maps that have other key-value pairs. For example, lecturer’s preferred\_times key will have value in maps format (the key is the day and the value is the preferred hours). Each word can contain numerics, alphanumerics, or alphabets. The program will handle the type for each attribute when parsing the test data using the walker.

## **Description**

Below are the examples of objects that will be parsed by the parser, including Lecturer, Class, Classroom, and Course.

### **Lecturer**

|  |  |
| --- | --- |
| Lecturer:  action: create  id: 123456  name: Yudistira Dwi Wardhana Asnar  teaches: IF4150 IF3150  preferred\_times:  monday: 7 8 9  tuesday: 13 14 15 | Create lecturer with associated id, name, and preferred times. This grammar can also be used to define courses that the lecturer teaches. |
| Lecturer:  action: update  id: 123456  name: Yudistira Dwi Wardhana Asnar  teaches: IF4150 IF3150  preferred\_times:  monday: 7 8 9  tuesday: 13 14 15 | Update lecturer with id `123456` and write the new name, course(s) taught, and preferred times for that lecturer. |
| Lecturer:  action: delete  id: 123456 | Delete lecturer with id `123456`. |

### **Course**

|  |  |
| --- | --- |
| Course:  action: create  id: IF4150  name: Rekayasa Perangkat Lunak Spesifik Domain  credits: 2  facilities: papan\_tulis lcd | Create course with associated id, name, credits, and facilities. |
| Course:  action: update  id: IF4150  name: Rekayasa Perangkat Lunak Spesifik Domain  credits: 2  facilities: papan\_tulis lcd | Update course with id `IF4150` and write the new name, credits, and facilities for that course. |
| Course:  action: delete  id: IF4150 | Delete course with id `IF4150`. |

### **Classroom**

|  |  |
| --- | --- |
| Classroom:  action: create  id: 7601  capacity: 90  facilities: papan\_tulis lcd pc microphone | Create classroom with associated id, capacity, and facilities. |
| Classroom:  action: update  id: 7601  capacity: 90  facilities: papan\_tulis lcd pc microphone | Update classroom with id `7601` and write the new capacity and facilities for that classroom. |
| Classroom:  action: delete  id: 7601 | Delete classroom with id `7601`. |

### **Class**

|  |  |
| --- | --- |
| Class:  action: create  course\_id: IF4150  name: K1  quota: 50  lecturer\_id: 123456 | Create class with associated course id, name, quota, and lecturer id. Lecturer id represents which lecturer will teach the class. |
| Class:  action: update  course\_id: IF4150  name: K1  quota: 50  lecturer\_id: 123456 | Update class with course id `IF4150` and name `K1` and write the new quota and lecturer who teaches the class. |
| Class:  action: delete  course\_id: IF4150  name: K1 | Delete class with course id `IF4150` and name `K1`. |

### **Display**

|  |  |
| --- | --- |
| Display:  entity: Course  filter:  credits: 2  facilities: papan\_tulis | Create a display of entity’s instances (courses) that fulfill the conditions that are mentioned in `filter` key, such as credits and facilities. |
| Display:  entity: Lecturer  filter:  teaches: IF4150 | Create a display of entity’s instances (lecturers) that fulfill the conditions that are mentioned in `filter` key, such as courses that they teach. |

### **Axiom**

|  |  |
| --- | --- |
| Axiom:  entity: Classroom Class  capacity: bigger | Create an axiom that involves entities that are mentioned in the `entity` key which are `Classroom` and `Class` and add the associated constraint(s). In this case, the constraint is that the capacity of the classroom has to be bigger than the capacity of the class. |
| Axiom:  entity: Course  entity\_id: IF4150  conflict: IF4071 IF4090 | Create an axiom that involves entity that is mentioned in the `entity` key which is `Course` and add the associated constraint(s). In this case, the constraint is that course with id IF4150 must not be held at the same time with course with id IF4071 and IF4090. |
| Axiom:  entity: Classroom Class  hold: 1  time: same | Create an axiom that involves entity that is mentioned in the `entity` key which is `Classroom` and `Class` and add the associated constraint(s). In this case, the constraint is that at the same time, a classroom must hold just one class. |

## 

## **Examples**

Below are the example of the scheduler output given objects contain in data.txt (we do not include the file because the content is too long but the format is similar to the example of objects mentioned in language grammar section above). 