

What does the project do?

UniTime is a university timetabling system that has four major components: course timetabling, examination timetabling, student scheduling and event management. Each of these components focuses on solving an optimization and a constraint satisfaction problem.








Pre-requirements:

UniTime needs all the data it will use while running to be in a database that it can access, then after it's up and running all the data from that database will be available on the system to be used in scheduling.

Course Timetabling:

Course timetabling means assigning times and rooms to classes, which is a difficult process given that there are many constraints to take into account while producing a timetable. For course timetabling UniTime needs data entered about the courses, the instructors, the rooms, the relationship between courses and classes (because one course can have many classes) and the curricula. All of the following are data and constraints on it that are considered when building the timetable for a university:

- The course offerings (which are the available courses)
- The scheduling subparts of a course (like lectures and/or lab)
- The Number of classes in a scheduling subpart (a class is part of a scheduling subpart of the course, and it actually needs to have a time and a place assigned to it and it has actual students attending it, e.g.: course A has 2 subparts a Lecture and a Lab the Lab subpart has 4 classes [lab 1, lab 2, lab 3, lab 4])
- Date pattern of a class (that is the weeks in which a class will be held so it can be all the weeks of the semester or a subset of that)
- Time pattern of a class (that is all the possible times within a week that the class can be held in, every time slot is given a rating on a scale from required to prohibited, both required and prohibited are considered strict constraints so they must be fulfilled by the system)
- Any room preferences for that class (so you can add constraints on the type of room that the class could be held in)

 Required	 Strongly Preferred	 Preferred	 Neutral	 Discouraged	 Strongly Discouraged	 Prohibited
--	--	---	---	---	--	--

- All Available rooms and information about:
 - The building in which the room is in, the type of the room and the group that the room belong to if any (like classroom, computer lab, chemistry lab)
 - What department/departments does that room belong to
 - At what time slots is the room available and for which department is it available (could be to everyone)
 - The travel time between this room and all the other rooms, which is important because a student can't have two consecutive classes in rooms that are too far apart for that student to travel from that room to the next
 - The features available in the room like audio recording, chalkboards or tables and chairs
- Distribution preferences which are constraints added on the relationship between two or more classes e.g.: classes can or can't be back-to-back, they must be held in the same room or on the same day.
- All instructors and information about what departments they teach in, when are they available to teach (again every time slot during the week is rated using the rating system mentioned above), and preferences or requirements made by instructors (can be for a time or a room or their distribution)
- Information about the students is crucial to produce a feasible timetable in this system such information can be obtained from more than one source:
 - Firstly, the Curricula, which is the courses taken by students in a certain major during a certain semester that specifies the percentage of students taking each of these courses.
 - Secondly the last year's enrolments
 - Lastly pre-registrations, which is more precise than last year's enrolments, but you can't guarantee that all students will apply.

The solver module:

After UniTime is equipped with all this data it needs to be loaded into its solver module. The solver will give warnings if there is a problem with loading any of the input data. Now the solver can find a solution using one of its two main configuration modes:

1. Check: which is a mode for finding a complete feasible solution, it's fast but not optimized. The word complete means that it respects all the hard constraint put by the data above an example on hard constraints are the size of rooms required, the room availability, the required or prohibited preferences applied in any area and of course that no instructor/room can have two classes at the

same time. A complete solution is found if 100% of variables are assigned otherwise check for the unresolved conflicts and make an appropriate change to the data then reload it or check the suggestions page.

2. Default: which is a mode for finding an optimized solution, meaning it respects all the hard constraints and also tries to accomplish as many soft constraints(objectives) as possible but it's not guaranteed like the Time/room/distribution preferences that aren't required or prohibited, student conflicts (a student having two classes at the same time or consecutively with too big of a traveling distance) and the traveling distance between back-to-back classes for instructors. The produced timetable can be modified manually. This process will take longer and has a timeout of 30 minutes.

The solver has also more options that work for both configurations like

- The source of the student course demands which can come from several sources:
 - Last Like Student Course Demands
 - Weighted Last Like Student Course Demands
 - Projected Student Course Demands
 - Curricula Course Demands
 - Curricula Last Like Course Demands
 - Student Course Requests
 - Enrolled Student Course Demands
- what action to take upon finishing generating the timetable
- whether the committed student conflicts should be loaded as well into the solver to be considered.

Every solution produced by the solver is given a rating on some properties that helps in deciding on if the solution is acceptable or not, what percentage of every type of soft constraint does it fulfill. The two most important metrics here are:

- Assigned variables percentage because it indicates how many of classes were assigned a time and a place (for a feasible timetable this must be 100%)
- Student conflicts which indicated the number of conflicts that exist with the current timetable.

The last step after reaching a desirable timetable would be to save it and it can be viewed from Courses -> Course Timetabling -> Timetable Grid. The timetable can be published as PDF or XLS

Examination Timetabling:

This is the process of assigning exams to time periods and locations, UniTime is used to produce such exam schedules by using data about:

- All examinations and info about
 - What class or course does that exam belong to know which students are taking it
 - The length of the exam, seating type and size (how many students)
 - Any requirements or constraints on an individual exam like room/distribution/period preferences
- All rooms available (with all the information discussed earlier)

For this problem the hard constraints specified in any of the preferences are satisfied along with some other hard constraints that are applied by default like:

- No two exams are in the same period and room.
- Examination must fit the period's time and room (or rooms) capacity.
- Room must be available to be used for examination.

The soft constraints are the same as the ones mentioned above with the addition that a student having two exams on the same day is also considered a conflict and the examination solver will try to minimize those conflicts as much as possible while it's optimizing the solution. Any unsolved conflicts will be shown in the reports section under examination timetabling where you can apply filters on the unresolved conflicts to see only a specific type of conflicts or conflicts for a specific subject.

Student Scheduling:

Sometimes students can't take a combination of courses because it will cause a conflict in their schedule. The Course Timetabling module tries to minimize these conflicts as much as possible, but it can't guarantee they won't happen. Student scheduling is the enrollment of students into classes in a way that maximizes the ability for students to get the courses they need. The first part in this process is for students to submit their course requests in which they specify what courses will they take and organize them by priority, and they can even add having a specific time of they day to be free from classes as one of their priorities. These course requests could be selected as the source of the student course demands considered by the solver in the course timetabling component while it's optimizing a feasible schedule produced from the requirements and preferences of the university departments. After the student fills their course requests, they can generate an initial schedule on which they

have the option to modify from the available options for every class as long as it doesn't create a conflict, or they could simply go back to their request and add more restrictions. When the student is happy with their schedule, they can submit it, so they are enrolled in those classes.

Event Management:

Event management is the management of the remaining classroom spaces. An event here means either a class, an examination or other event. Those other events are what mainly need to be managed by the Event Manager. Users of UniTime can use this module to:

- Search for a room with specific feature, type or group
- See all events that were requested regardless of if they were approved or not.
- Add an event to be approved.
- The availability of all rooms or a specific kind of room in any day in the semester

To add a new event request to the system the user must provide the event's name, sponsor, type, expected attendance, contact info of the person to contact about that event, requested services (like catering, video recording, unlock service), and an expiration date of that request after which approving it will be too late.

There are three types of events that the user must choose on from:

1. Special Events: which anybody can request.
2. Course-Related Events: which can't be entered by anyone and will only be approved if approving such an event won't cause conflicts to students in that course.
3. Not-Available Events: can only be added by event managers and they indicate that the room isn't available for events.

Any room or department in the system can define event status, meaning it can add constraints on whether anyone can request an event or not, who can request an event, the default break time after events and what type of authentication is required for event approval.

There are three types of users that can work with event management:

1. Event Managers:
 - a. Can setup event statuses, notes, and room availabilities (for rooms in their department, because every event manager belongs to a department)
 - b. Create events on behalf of other users.
 - c. They approve/reject events.
 - d. Can delegate roles to other departmental users.
2. Instructors:
 - a. Can see their classes and/or exams, including enrolments.
 - b. May cancel or re-schedule individual class meetings (when allowed)
 - c. Can request special and course-related events.
3. Students (and other authenticated users):
 - a. Can see their schedules.
 - b. Can request special events.

The lifecycle of an event in the system:

- Pending which means that the requested space is blocked, but not for classes or exams.
- Approved which means that space is blocked, and event cannot be deleted (only cancelled by either the manager or the owner)
- Rejected / Cancelled which means that space is released.
 - If an event expires it's cancelled
 - A manger can cancel an event.

What Technique was used to obtain that description?

Opportunistic Approach was used to gain understanding of the system as it was the logical choice for an unfamiliar large system with complex code where top-down and bottom-down approaches cannot be used. Thus, the hybrid approach was the recommended choice as Letvosky suggests.

Steps to form mental model:

1- top-down approach was used to gain an overview of the system and its functionalities by reading the documentation and testing the application firsthand to familiarize with the system and its major components Course Timetabling, Examination Timetabling, Student Scheduling and Event management.

2- Moving to lower-level and applying bottom-up approach to verify hypotheses in incremental manner. Here, we mostly used cross-referencing to map the high-level components to its respective class(es).

For example, in solver package `InstructorSchedulingSolver.java` as the name proposes is used to generate instructor's schedule free of conflicts while students schedule may be not conflict-free, instructors must be conflict-free as per the requirements. This is also considered a good-quality beacon as the class name is meaningful and gives a cue to what's being computed.