## Title

MyCourses – A Course Scheduling System

## Abstract

The program MyCourses provides as optimal as possible a plan for scheduling courses. Every university is faced each year with the same problem: How to schedule a large number of courses in an optimal way while fulfilling a number of constraints, such as available lecture rooms, limited availability of lecturers, students' selections of the courses, and similar. MyCourses should be implemented as an interactive program that (i) enables entering data, such as courses, the faculty members, the available facilities and some constraints related to the course scheduling, (ii) calculates and proposes a scheduling for courses, (iii) makes it possible to manually update the proposed schedule, but keeping track of the consistent scheduling, and (iv) provides a presentation of scheduled courses.

## 1. Introduction

Course scheduling is a tedious and error-prone task when done manually or semi-manually. For this reason a program that can automatically produce course scheduling with given requirements and constraints is very important. The goal of this project is to develop a course scheduler, MyCourses.

MyCourses will make it possible to enter data and requirements in a simple way using a web-based interface, calculate and propose a schedule, enable manual updates, and finally present the schedule for the selected courses.

Since different people (students, lecturers, program planners, etc.) will use the program its user-friendliness is crucial. An efficient automatic scheduling is also important, but even more important is a possibility to manually re-schedule or pre-schedule some courses or course elements (like lectures, labs, etc.).

The project includes requirements solicitation, requirements specification, design and the implementation. The program should be implemented as a distributed web-based, application, and data should be stored in a database. Since the program is aimed for universities, it is expected that FLOSS (Free/Libre and Open Source Software) will be used.

## 2. Application domain and scenarios

We describe here MyCourses by presenting several scenarios.

1. Entering programs and courses
   A *program administrator* who is responsible for management of the programs at the university defines programs. She identifies the program, its running period (starting and ending year), and a program manager who will have the overall responsibility for the program. Typically when defining a new program, the same program from the previous year would be copied, with some data changed afterwards.

   A *program manager,* when enabled, can enter all details about the program: Which courses it includes, which of these courses are obligatory and which optional, etc. The courses may already exist, and in that case she creates a new course instance with a given period of its execution, who is the *main lecturer* ("examiner"), and some additional general information about the course. If the course is new, then the program manager creates it first, and then creates a new instance of it.

The *main lecturer* defines the details about the course instance he is assigned for: Which are other people from the teaching staff involved in the course, which are the course elements (lectures, tutorials, labs, projects…), the way of possible course execution (a number of lectures and other elements per week, preferred days, expected number of students, and similar). He may wish to (smart) copy all data from a previous instance of the course.

2. *Entering resources*
   An administrator, or a program administrator enters data about different resources: The available lecture rooms and laboratories in which the courses (or particular elements) will take place, and some other elements such as data about number of available places, or availability of the room is entered.

3. *Scheduling*
   Several users can run a (semi)automatic scheduling process that provides a schedule proposal for a course or a set of courses (e.g. the entire program, or selected courses): the days and time, and places should be scheduled. The scheduler is not necessarily an automatic solver – but it allows some manual predefinition of the schedule, and manual changes after the proposal is created. The main lecturer can run the scheduler for his course, and mark it as a course schedule proposal. The scheduler shows if some conflicts occur. The program manager can verify the scheduling in combination with other courses in the program. The program manager can modify the scheduling if necessary and then freeze it (i.e. make it official).

4. *Presentation*
   Different users can use the program to present data. Examples of presentations: Availability and utilization of the facilities; Schedule of a particular course; Schedule for a faculty member; A schedule for a student (after she defines in which courses is she enrolled), and similar.

## Project goals (requirements)

The scenarios listed above basically define the requirements of the projects. It is expected that the project members will use these scenarios to a) specify and refine the requirements and b) build a prototype, which will be used for a communication with the customer through several iterations.

It is important that the program is configurable and flexible – i.e. that it can be configured with different rules of scheduling (for example definition of semesters, periods of the course execution, and similar)– which can be quite different at different universities. At the same time it is important that the program is user-friendly and that it is easy to import/export data to other systems which the universities can have.

The user-part (client) should be web-based, while the server can use a particular technology, preferably FLOSS.

## Intended output of the process (process focus)

We recommend using an iterative development process. However the participating teams are free to choose the development process. In any case they must provide evidence that will prove their adherence to the chosen methodology. Required evidence includes, but is not limited to a standard set of documents for a particular methodology (for example: project plan, requirements, architecture and design, test plan document, etc.). User instructions are also required, and should be a part of the final product (on-line help for system users).

The project deliverables, beside required project documentation, should include the following artefacts:

- The project documentation
- The application ready for the installation
- If possible a running application (on a server) that can directly be configured and used.
- The source code

We also encourage that the following requirements are also met:

- UML should be used consistently throughout the project documentation
- Automatic testing tools should be used

## Tools and standards

We encourage project teams to use FLOSS components in this project as much as possible.

## Interaction between stakeholders and developing teams

The contact persons for the MyCourses project are Ivica Crnkovic (ivica.crnkovic@mdh.se) and Thomas Leveque (thomas.leveque@mdh.se). They will be available for questions regarding project throughout the year with the exception of vacation days. Please note that questions will probably not be answered on a daily basis (rather once or twice a week), so it is advised that you collect all your weekly questions in one mail.