Coco Scheduling

(Automated Core course scheduling system)

Client: Computer Science Department

Contact: Tanja Breinig, Erich Reindel

A project, in partial fulfillment of the Software Engineering course, Winter Semester, 2015

Tutor: Rafaella Antonyan (Tutor)

Group Members

- Sotaya Yakubu (Project Lead)
- Emamurho Ughergbe
- Anastasiia Izycheva
- Kemi Oladipo
- Deepti Mittal
- Adekunle Seun
- Amanuel Ghirmay

27th November, 2015

Table of Contents

Project overview

The Computer Science department offers about 6 to 12 core courses each semester. To schedule this courses, a huge number of constraints have to be considered, e.g. availability of rooms, number of estimated participants, no overlap with other core courses or courses belonging to the same field of study and many more. In addition some time slots are favored (e.g. Tue, Thu 10-12) than others (Mo 8-10, Fr 4-6).

The aim of this project is to develop a web based system that is browser independent, having a user friendly and interactive graphical user interface to help schedule the core courses. The system is expected to serve as the link between the lecturers and the study coordinator.

The system should be able to provide lecturers options which fulfil as many requirements as possible while avoiding strong inconveniences. All constraints must be clearly identified and algorithms that enable the lecturer and the study coordinator to interact efficiently should be developed. These algorithms should therefore optimize the scheduling process. Also a database should be incorporated and the front ends should be coded.

Scope of the project

This project group is expected to develop a web-based system that is able to automate the scheduling of core courses in the computer science department. The system should be functional enough for the lectures to submit their preferences for core courses they are teaching in each semester. Considering the predefined constraints, the study coordinator should get a suggestion how time slots, venues and other applicable logistics should be.

Definitions

Clients definition: Representatives of the computer science department.

User definition: The users of this system are the lecturers and the study coordinator

Contract definition: The document containing the specification of the software to be developed, which is reviewed by the client and agrees to its content.

Types of Users

- Study Coordinator
- Lecturer

Requirements List

Must-Have-Features

(Note: Core Features that are absolutely necessary for the system to be usable).

Priority	Feature	Category
1	Provide an email-function which should be used for authentication by both Lectures and coordinators.	Functional
2	Lecturers must be able to submit their preferences for the core course scheduling. i.e their date preferences, and course they are offering.	Functional
3	The system should provide an automated scheduling of the core courses based on the lecturer's preferences and the various system constraints resulting in schedule suggestions.	Functional
4	The system must be a stand-alone system (i.e independent of the Computer Science online portals)	Functional
5	The system must be web-based and provide an easy to use calendar based user Interface.	Functional
6	A documentation on the usage of the system and different aspects of it during usage as help-function. And the system should provide a note to the lecturers view to notify them of what and what not to do while selecting their preferences.	Non-Functional
7	A documentation of the coding has to be delivered at the end of the project.	Non-Functional
8	The study coordinator has to be able to define for each CC to which field of study it belongs, e.g. computer graphics, formal methods, embedded systems. Also the study coordinator has to be able to block respective time slots for selected CCs.	Functional
9	Based upon the suggestions of the system the study coordinator can change one or more time slots by hand, block time slots and afterwards he can press an	Functional

	"execute button" and the systems re-calculates the new schedule.	
10	The coordinator should be able to modify constraints which were originally predefined.	Functional

May-Have-Features

(Note: This section of the features highlights the features to be implemented after the *must-have features* but are not the core features necessary for the system to be usable. This usually will come at the end of the lecture scope).

11	Automatically generated email for lecturers to submit the core course plans for the semester.	Functional

Must-not Have

(Note: This section highlights features that the customer needs, but we cannot deliver by the end of the lecture scope.)

12	Automated email reminder for lecturers	Functional

System Constraints

- Core courses must have two lecture slots in a week.
- There must be an interval of at least one working day between a core course lectures.
- The scheduling of a CC should be well-balanced, e.g. it should be avoided, that one lecturer has only preferred time slots whereas one other has only "bad" slots.
- There must not be an overlap of core courses and advanced courses belonging to the same field of study.
- There must not be an overlap of core courses.
- Courses should be assigned lecture halls with enough space as the estimated number of students taking the course.
- No two courses should take place in the same hall at the same time and date.
- No course should be scheduled on a blocked time period except if the lecturer preference explicitly wants that time slot.