Meeting 2010-09-12

First draft as of 2010-09-12

What we've done so far

We created a preliminary description of Entities (as planned) and derived a relational model from it.

Agenda

Review Use cases, approve and freeze them.

UseCases Review

rsfs_manuallyUpdateProposedSchedule

In the discussion of this use case we thought about allowing concurrent versions of a proposed schedule, e.g. users change the schedule which was automatically proposed by the scheduling algorithm (that proposed schedule is a global schedule visible to all users and acts as their common set of data they operate on) and their change creates a new version. At the end of the reviewing process an administrator would merge all proposed version. However, this will eventually end up in conflicting versions and would therefore be contradictory to the goal creating a working schedule, since the administrator would have to ask the affected lecturer. Long story short: We decided that users act on a common data base and their is one and only one version of a proposed schedule.

USER CHANGES A PROPOSED SCHEDULE

Actors	(Program)-Administrator, main lecturer, GUI
Scope	GUI
Iteration	2
Precondition	System is installed/running correctly and a proposed scheduling is present.
Postcondition	System is still running, user is still logged in and the previous schedule is not lost if the changes were invalid.
Postcondition on Success	The manually updated schedule is in the database as a proposal overriding the old one.
Basic course of events	(1) User logs in. (2) User opens schedule-panel and choses to edit the Schedule. (3) User alters the parts of the Schedule he wants to. (4) User tries to save data. (5) Scheduler validates and enters data into Database. (6) The GUI presents a message on success.
Alternative paths	(5a) Conflicts occur, a errormessage is presented
Open Questions	Q: What does it mean when proposed changes are valid? A: All constraints are satisfied Q: Should the programadministrator get a notification on his next login? A: yes, he should Q: Will there be proposals for corrections if conflicts occur? A: No proposal, better inform the user about conflicts that occur immediately implementation notes Q: 5a When will conflicts, if any, be detected? A: The scheduler detects conflicts at step 5a
Implementation notes	Q: 5a When will conflicts, if any, be detected? A: The scheduler detects conflicts at step 5a

iped_availableFacilities

ENTER DEPARTMENS

Actors	(Programm)-Administrator, GUI
Scope	GUI
Iteration	2
Precondition	System is installed/running correctly.
Postcondition	System is still running and user is still logged in.
Postcondition on Success	A new department is inside the database with the given data.
Basic course of events	(1) User logs in. (2) User opens department-panel and chooses to create a new department. (3) User enters requested data via form. (4) User tries to save data. (5) GUI validates data (format validation). (6) GUI enters data into Database, a success message is presented.
Alternative paths	(5a) GUI detects error in data and raises error message and requests data-input
Open Questions	none at this time.
Implementation notes	none.

iped_someConstraints

We identify two use-cases (at this time of development) regarding constraints on courses which affect scheduling. They are:

DEFINE A FEATURE

Actors	(Program-)Administrator, GUI
Scope	GUI
Iteration	2
Precondition	System is installed/running correctly
Postcondition	System is still running and user is still logged in and no side effects on database
Postcondition on Success	A new feature is defined which can be required by courses and provided by rooms
Basic course of events	(1) User logs in. (2) User opens feature-panel and chooses to create a new feature. (3) User enters data about feature via form. (4) User tries to save data. (5) GUI validates data. (6) GUI enters data into Database, a success message is presented.
Alternative paths	-
Open Questions	Q: how to implement point 3. of specification 'availability of the room' ? whats that in detail ? A: workaround with another table
Implementation notes	-

Assign a constraint to Course Elements or Course Element Instances

Actors	A: main lecturer B: (program-) administrator GUI
Scope	GUI
Iteration	2
Precondition	System is installed/running correctly and features are already created.
Postcondition	System is still running and user is still logged in and no side effects on database.
Postcondition on Success	The course element or course element instance requires this new feature.
Basic course of events	(1) User logs in. (2a) User opens feature-panel and chooses to assign a new feature to a course element instance. (2b) User opens feature-panel and chooses to assign a new feature to a course element. (3) User chooses feature and enters feature-specific data and defines if it's a hard constraint(obligatory) or if it's a soft constraint (optional) via form. (4) User tries to save data. (5) GUI validates data. (6) GUI enters data into Database, a success message is presented.
Alternative paths	(2a) User opens constraint-panel and chooses an existing constraint from a course element/course element instance. (3b) User chooses feature and edits feature-specific data and defines if it's a hard constraint(obligatory) or if it's a soft constraint (optional) via form.
Open Questions	-
Implementation notes	-

iped_availableLectureRooms

CREATE ROOMS (IN THE DATABASE)

Actors	(program-) administrator; GUI
Scope	GUI
Iteration	2
Precondition	System is installed/running correctly, departments available.
Postcondition	System is still running and user is still logged in and no side effects on database.
Postcondition on Success	A new Entity reflecting Room is created in the database.
Basic course of events	(1) User logs in. (2) User opens room-panel and chooses to create a new room. (3) User enters data about a room via form. (4) User tries to save data. (5) GUI validates data. (6) GUI enters data into Database, a success message is presented.
Alternative paths	-
Open Questions	-
Implementation notes	-

iped_limitedAvailabilityOfLecturers

A LECTURER ENTERS HIS TIMETABLE

Actors	lecturer; GUI, Database
Scope	GUI
Iteration	2
Precondition	System is installed/running correctly AND configured correctly (esp. number of days, no. of time slots etc.); lecturer created (in fact the user is marked as lecturer and his data are stored in the Database)
Postcondition	System is still running and user is still logged in and no side effects on database.
Postcondition on Success	The timetable has been created/altered. It is saved in the Database.
Basic course of events	(1) User logs in. (2) User opens personal panel and chooses to enter his timetable/availability-times. (3) User (de)activates time slots of the week when he/she is available via predefined timetable for availability in general (will be applied for whole semester). (4) User saves data. (5) GUI presents a success message.
Alternative paths	-
Open Questions	-
Implementation notes	-

iped_studentSelectionOfTheCourses

A STUDENT SELECTS A COURSE

Actors	Student; GUI, Database
Scope	GUI
Iteration	2
Precondition	System is installed/running correctly AND configured correctly (esp. number of days, no. of time slots etc.); student created/is studying at the university (in fact the User is marked as student).
Postcondition	System is still running and user is still logged in and no side effects on database.
Postcondition on Success	Courses the students attends are saved in the database.
Basic course of events	(1) User logs in. (2) User opens personal panel and chooses to add courses to his current semester. (3) User gets a list of all available course instances available for him/her this semester (4) User chooses one and can enroll in this course. (5) GUI saves course instance data in Database.
Alternative paths	-
Open Questions	-
Implementation notes	-

pofsc_webBasedInterface

DISPLAY SCHEDULING-DATA BY ROOM

Actors	any member of the university (student, lecturer, main lecturer, (program) administrator); GUI
Scope	GUI
Iteration	2
Precondition	System is installed/running correctly, scheduling has finished.
Postcondition	System is still running and user is still logged in and no side effects on database.
Postcondition on Success	(1) User logs in. (2) User opens room panel and chooses to view scheduling for rooms. (3) User gets a list of all available rooms and selects one. (4) User gets a timetable with all courses using the chosen room during the current week (if not specified).
Basic course of events	(3a) User also selects the time for which week the plan should be displayed. (4a) User gets timetable for the given week.
Alternative paths	-
Open Questions	-
Implementation notes	-

pofsc_presentScheduleForSelectedCourse

DISPLAYING COURSE INSTANCE DATA (INFORMATION ABOUT COURSES)

Actors	Student; GUI, Database
Scope	GUI
Iteration	2
Precondition	The user is logged in with student rights.
Postcondition	The user is still logged in with student rights.
Postcondition on Success	-
Basic course of events	(1) User opens course panel and chooses to view course instance details. (2) GUI presents all course instances of the current academic turn. (3) User selects a course instance. (4) GUI displays details about the selected course instance (including course element instances).
Alternative paths	-
Open Questions	-
Implementation notes	-

pofsc_scheduleForAStudent

STUDENT VIEWS HER/HIS SCHEDULE

Actors	Student; GUI, Database
Scope	GUI
Iteration	2
Precondition	The user is logged in with student rights. The student is enrolled in courses.
Postcondition	The user is still logged in with student rights.
Postcondition on Success	-
Basic course of events	(1) User opens course-panel and chooses to view his personal schedule. (2) GUI queries the database view of the proposed academic term. (3) Database filters the view by the students courses and delivers the data to the GUI. (4) GUI displays the students course schedule.
Alternative paths	-
Open Questions	-
Implementation notes	Q: Is it useful to organize the course schedules with a database view? A: Yes it is. Especially in PostgreSQL, which we use, is a view something like a stored (pre defined) query. The alternate is to create real entities in the database which could lead to update anomalies and other inconsistency as well as to heavy more usage of storage (since it is redundant information and students are the most objects in our database).

scala_definitionOfAcademicTerms

THE PROGRAM ADMINISTRATOR DEFINES THE ACADEMIC TERM ¶

Actors	administrator; GUI, Database
Scope	GUI
Iteration	2
Precondition	The user is logged in with program administrator rights. There is no academic term configured yet.
Postcondition	The user is still logged in with administrator rights.
Postcondition on Success	The academic term definition has been set up and the database is initialized.
Basic course of events	(1) User opens university panel and chooses to create a new academic term for the university. (2) User enters the time span for the academic term. (3) User tries to save the data. (4) The GUI validates the data and inserts the data into the database. (5) The GUI displays a confirmation message.
Alternative paths	(4a) The GUI falsifies the data. (5a) The GUI displays an error message.
Open Questions	Q: Should we consider semester breaks respectively period of course execution separately? A: No, but semester breaks (vaction, holidays) are scheduled manually.
Implementation notes	-

misc_importDataToTheDatabase

MISCELLANEOUS: IMPORT DATA TO THE DATABASE

Actors	Administrator; GUI, Database
Scope	GUI
Iteration	2
Precondition	The user is logged in. The scheduling shall not be frozen.
Postcondition	The user is still logged in.
Postcondition on Success	The import of a file was submitted to the database.
Basic course of events	(1) User opens the import/export-panel. (2) User selects a file to import. (3) GUI validates the file. (4) GUI queries the database. (5) Database changes its state. (6) GUI displays a confirmation message.
Alternative paths	-
Open Questions	(3a) The GUI falsifies the file. (6a) The GUI displays an error message.
Implementation notes	Q: Are there any other kind of imports besides the whole database data and its scheduling? A: The only imports allowed are the whole database state, when setting up the system or basic data for instance Courses, Lecturer, Rooms, Features, Course Types and Persons.

misc_ExportDataToOtherSystems

MISCELLANEOUS: EXPORT DATA TO OTHER SYSTEMS

Actors	User; GUI, Database
Scope	GUI
Iteration	2
Precondition	The user is logged in.
Postcondition	The user is still logged in.
Postcondition on Success	There is a file with the data of the database.
Basic course of events	(1) User opens the import/export-panel. (2) User chooses the data he wants to export. (3) User chooses a file type for the export. (4) GUI queries the database for the data. (5) GUI writes the received data to the matching file format. (6) The GUI displays a confirmation message.
Alternative paths	-
Open Questions	Is it part of the use case to specify the formats we want to export? If so, what should we support? Plain Text, SQL, XML -> (PDF, CSV)
Implementation notes	-

Further course of action

Julian improves his relational model based on our todays enlightment and sends and mails it to the other team members tonight. The others provide feedback using PDF Annotations until Monday, 2010-09-13 23:00, which is then incorporated into the model by Julian for our next meeting, so that the entities can be finally discussed, approved and freezed then.

Hagen develops an informal description as well as a sequence diagram in UML for the scheduling process until Monday, 2010-09-13 18:00 and mails it to the other team members. The resulting set of documents are intended as another, yet more complex use case. The artifacts created are to be discussed, approved and freezed in our next meeting.

André prepares the only missing use cases (regarding Login and Entities) for our next meeting.

Everybody reads and comments on the Guidelines which can be found in our wiki on the page “OurGuidelines”. They are to be approved by all of us so that they can be freezed in our next meeting.

Next meeting

The next meeting is on 2010-09-14 09:00 @ Hagen. Since we will not be complete than we meet two times on that day, another time 19:00 @ Hagen for setting up a common development environment. Julian will do his presentation then, introducing the already existing Makefile from our Subversion repository and discussing Guidelines and advices for working and living together happily.

The agenda on 2010-09-14 therefore is:

09:00	(1.0) Discuss remaining UseCases (including description of Scheduler). (1.1) Approce and freeze UseCases. (2.0) Discuss Description of Entities (2.1) Approve and freeze Description Entities. (3.0) Discuss Relational Model. (3.1) Approve and freeze Relational Model.
in between	recreation

19:00	**Julians presentation:**
	• Learn how to use make with the Makefile
	  • *Set up Environment on everybody’s Notebook*
	• Learn how to hack the Makefile
	  • *Hack the Makfile*
	• Learn how to build and deploy the project
	  • Introduction to AspectJ
	  • Introduction to JUnit
	  • *Writing a sample Class and JUnit-Test*