Meeting 2010-09-08

In between Iterations 1 and 2.

Evaluation Iteration 1

Where are we now?

We have two applications, a Java-application named ju-scheduler and a a PHP-application named an-roles-prototype. *ju-scheduler* should be a working implementation of a genetic course scheduling algorithm, *an-roles-prototype* should provide a rudimentary user administration and communicate with *ju-scheduler*.

ju-scheduler actually is a straight-forward implementation of a genetic algorithm, lacking a cross-over function. It supports reading and writing data from an XML-file as well as to and from a PostgresQL-database. Error-handling is not available.

an-roles-prototype features a possibility for logging in, but does not check permissions. One can enter entities which are used by *ju-scheduler*. It also features an option to start the scheduling process.

What can we do better?

The prototype applications were not documented at all, making the code unmaintainable and hard to understand. If we were to build on the prototype code we'd be seriously slowed down in our development process. Functions and methods did not have a formal specification and no extensive testing had been done, therefore we do not now about the correctness of the implementation.

We should improve that! We need:

- specification of what we're going to implement,
- documentation, documentation, documentation,
- and automated tests.

Thoughts on our current scheduler prototype

A genetic algorithm implementation always consists of a fitness function calculating a score (called fitness) for a possible solution, a crossover function mixing two possible solutions and hopefully creating a better one and a mutation function which is used to randomly (or with some kind of intelligence) change a possible solution so that new possible solutions arise. Our current implementation lacks a crossover function. A possible solution is called *individual* or *genome* and is currently implemented as a matrix of all possible timeslots in each available room. That implementation is very memory consuming and it is very hard to implement a reasonable crossover function for it. Therefor we think that in a future implementation the genome should focus on the courses, not the timeslots.

Course of action for the next iterations

We went through the project specification again and compiled a document from it, featuring key-words separated by concern for what our project goals should be. It can be found on wiki page *ScoreSpecification*. It features information on functional requirements, non-functional requirements (e.g. usability) and the intended development process.

Project plan

We will set up a project plan and document it. Therefor we create weekly goals to achieve. The project plan is documented in our project wiki using the Trac feature *Roadmap*.

Shorten iterations

Since the submissions of our report is due 2011-01-15 we think our iterations are too long (currently they span 6 weeks).

(Iteration 2) For our current iteration we set a duration of 4 weeks, including KW 36 to 39. It is due 2010-10-03. We will do the evaluation of our next iteration on that day, Sunday 2010-10-03.

(Iteration 3) The third iteration is set to start after the second iteration ends and is due 2010-10-31. We will probably do the evaluation of the third iteration on that day too, since it's a sunday, but no date has been fixed yet.

Presentations

Herr Schweppe recommended to us doing little presentations for ourselves to inform the whole team about certain topics – there are specialists for different kinds of things in our team (the scheduler, web design, development with subversion, ...) which everybody should be aware of. The following presentations will be set up:

Development of AspectJ using GNU Makefiles and the subversion control system

This one is about the already existing Makefile in the trunk-folder in our current subversion repository and how to do development in SVN without interfering with the other team-members. Additionally a short introduction to AspectJ and how our project benefits from it will be given. Julian will do it in our next meeting in which everybody is attendant.

Corporate Design and Human Interface

André is going to prepare some mockups regarding the look and behavior of the human interface of our application.

Database-Access using Hibernate

Since Konrad currently is the most experienced team-member regarding hibernate he will inform us about creating DAOs using Hibernate and maybe the *Java Persistence API* which allows for declarative programming using Annotations.

Goals for the next iteration (iteration 2)

Write use cases – all of them

Our current progress of the requirements analysis has not much matured. There is especially an insufficent number of use cases which do not even cover the project specification. Therefor one goal for Iteration 2 will be to create these use cases as fast as possible. The target is a set of documents covering the project specification. These use cases will specify the desired functionality, i.e. the intended product that will be our final deliverable. Once the use cases are written it is to be decided which one of them will be implemented within the next iteration.

Resulting artifacts: A set of documents.

Informal description of the entities the scheduler deals with

We need a description of each entity that is used in the scheduler. The description should be written in informal language, containing no code or formal specification at all. However, the resulting document should act as a reference on what the entities are and what they do, e.g. “a room has a number, is located in a building and has different features which can be present in a certain quantity, such as a beamer or 200 seats.”.

Resulting artifact: A single document.

Architecture and design of the scheduler

Specify the intended design of the scheduler and the processing model of the algorithm. This includes, but is not limited to, a description of objects and relationships the scheduler deals with as well as description of the algorithm, both using UML.

Resulting artifacts: A UML class diagram and a UML sequence diagram.

Clean implementation of the scheduler

Since the current implementation of the scheduler is rather a proof of concept and acts only as a prototype we need an implementation of the scheduler which is well documented, scalable and easily extendable. It is to be implemented according to the specified design and description of entities (see our other goals).

Unit tests are very important to ensure the scheduler complies with our specification.

Result: Well documented source code with unit tests.

Visual design of the web interface

This one will be done in a presentation by André (see above under “Presentations”).

Resulting artifact: Mockups

Implement a working and extendable web interface

The implementation should feature login and logout, roles (in a generic way, i.e. roles are user-definable), user administration, CRUD of Entities used by the Scheduler, presentation of schedules, start scheduling.

It is crucial to the web interface that it is easily extendable, since most changes occur in the user interface and most functionality may be added here.

Result: Well documented source code.