Verification and Validation

Web Interface

The meausre of all things in means of testing the web interfaces are the use cases we have written. With a total of 70 use cases we sought to cover most of the user scenarios. In principal we went through the use cases to test the web interface. As a matter of fact the perspective of the developer is very different to the perspective of the user. For this reason we let family members and friends go through some of these use cases. However this approach was done very seldom and mostly at the end of the development process as we were in need of the time for finalizing the application.
We benefited from these external tests for discovering problems with the web interface regarding the usability, for instance when a user did not know what to do next.

Scheduler

In order to test the correctness of *MyCourses* we applied JUnit. Additonally we used Cobertura to check the code and branch coverage of each unit test. Since total code or even branch coverage was a too time intense task, we decided to concentrate the unit tests on only the most important parts of *MyCourses*. This encompass unit testing for the scheduler and the complete data access layer where the tasks are most critical.

In order to examine the overall performance of the scheduler a parametrized test data generator was implemented to run the scheduler with different test data. It was useful the scheduler was developed as a standalone application. This way the benchmark could be implemented decoupled from the rest of the application.
The results of the benchmark are shown below. The percent of constraints applies to the possible user-defined constraint. For instance 25% hard constraints means:

- every fourth course gets a time slot preference
- every fourth course gets a room preference
- every fourth lecturer gets a time slot preference
- every fourth room gets a time slot preference
- . . .

The results show with less constraints an optimal scheduler solution is reached in sufficient time or becomes at least very close to be optimal. With more constraints it seems to become impossible to find a solution, even with far more time then 3 hours as shown in the diagrams.

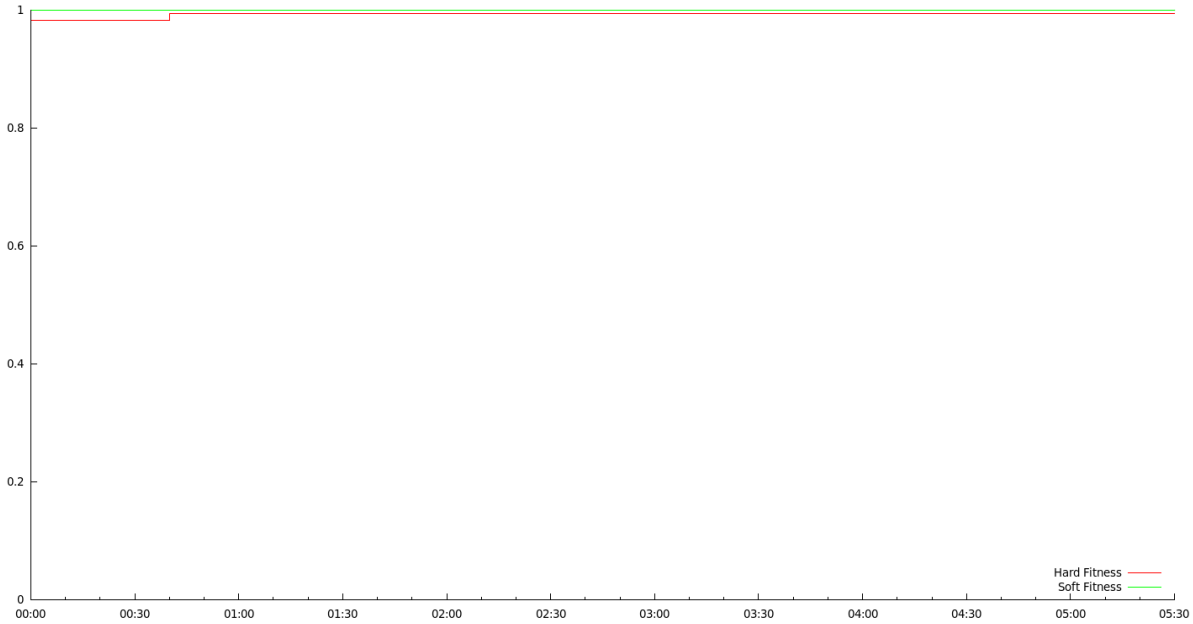

Figure 1: Hard constraints: 0% Soft constraints: 0%

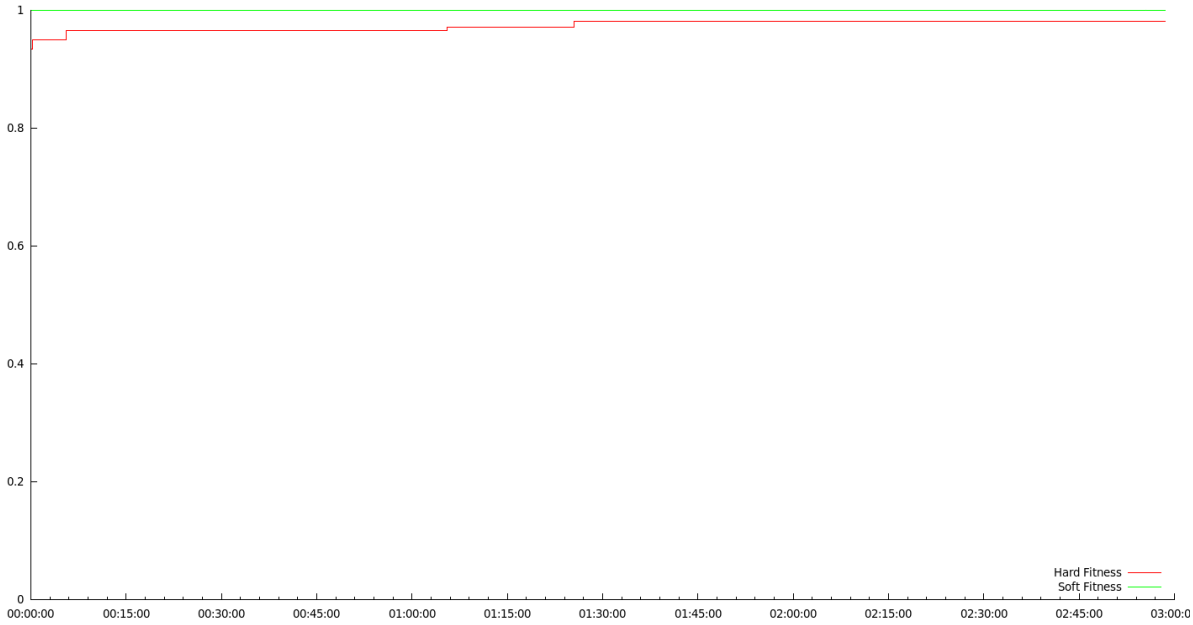

Figure 2: Hard constraints: 15% Soft constraints: 10%

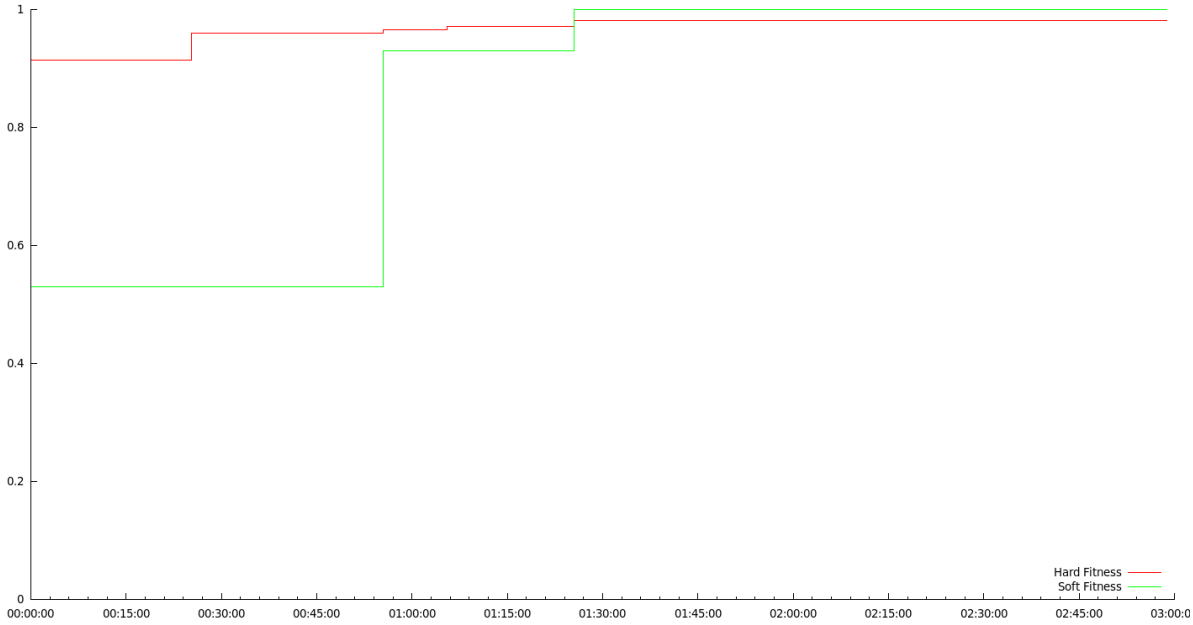

Figure 3: Hard constraints: 25% Soft constraints: 15%

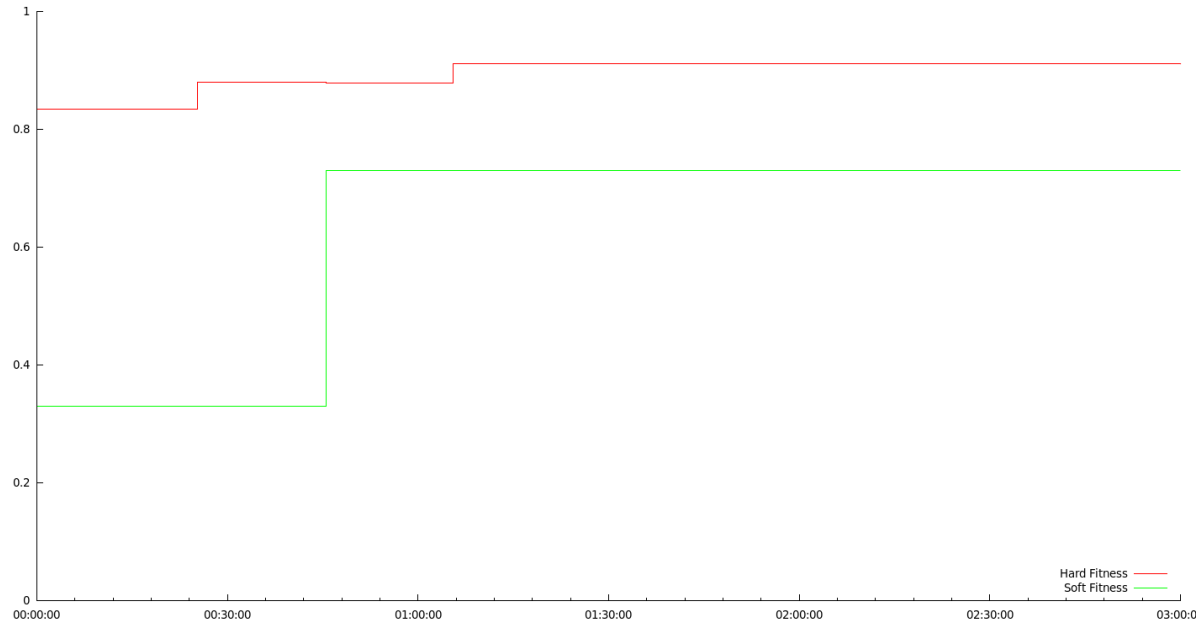

Figure 4: Hard constraints: 50% Soft constraints: 25%

Profiling

We used a profiler to inspect the Java Virtual Machine while executing the program. This lead to a step-by-step process of picking up the slowest component display in the profiler and trying to improve its performance. In some cases this procedure gave us huge performance boosts. The following screenshot shows a snapshot of the method time measurement with focus on the scheduling methods.

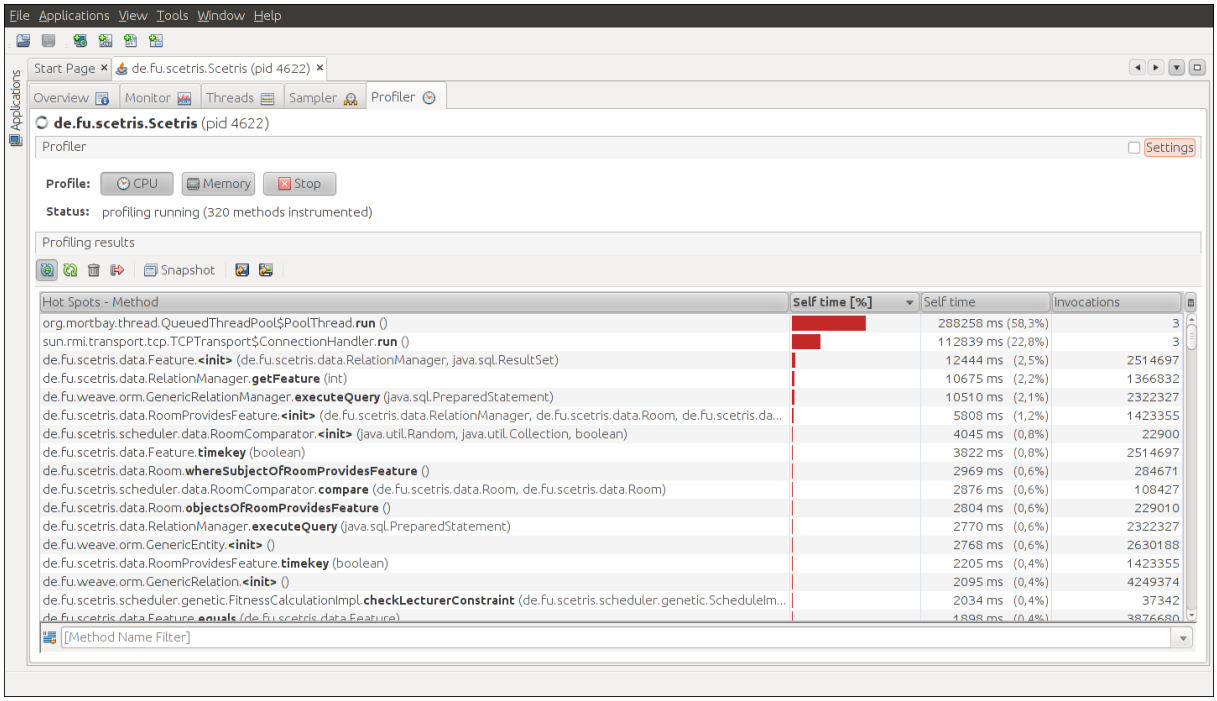

Figure 5: Measurement of the scheduling methods

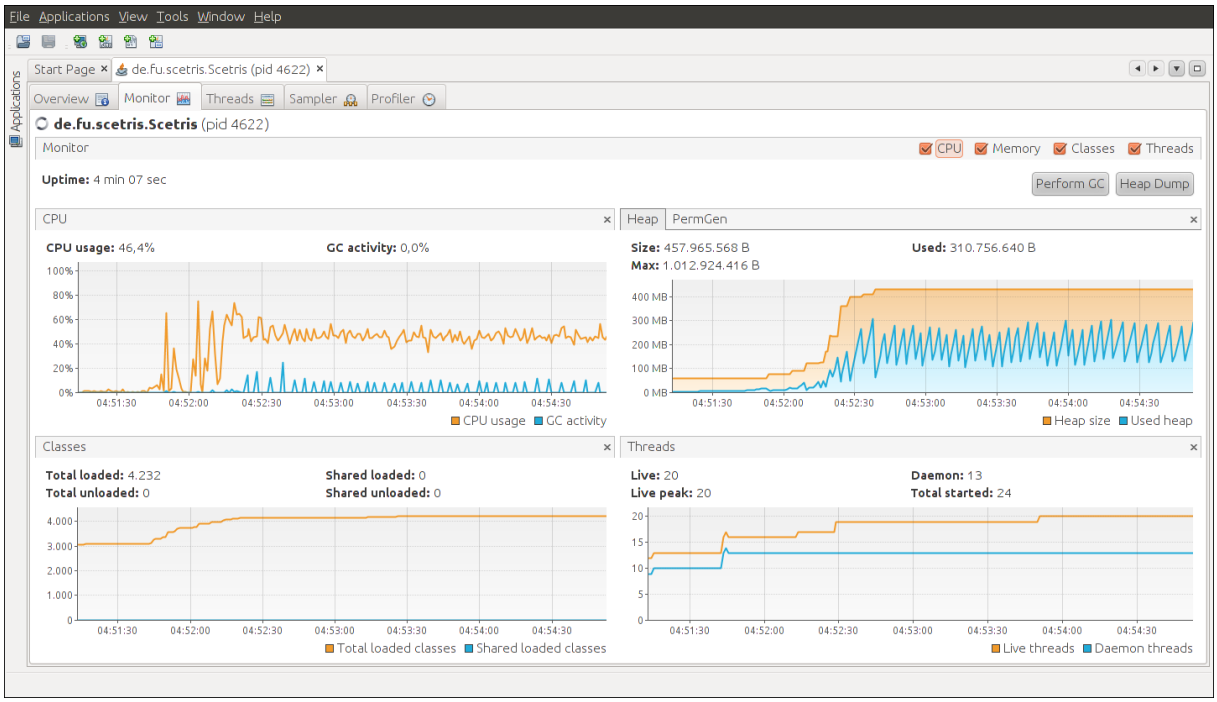

Figure 6: Inspecting the memory allocation