SCORE SCRIPT 2011-18-05

Mittwoch 19:00

Alles anders als alle Ander'n

SHOW SLIDE BLACKBOARD

THIS shows Blackboard, a well-known Software used for managing courses and especially seminars at our university.

SHOW SLIDE CAMPUS MANAGEMENT

THIS shows Campus Management, a software written and customized by SAP. It's used for booking courses and keeping records, such as grades.

SHOW SLIDE KVV

THIS is a piece of software used at our university for **actually** booking seminars (modules need to be booked using Campus Management as seen in the previous slide).

Dear Ladies and Gentlemen, dear Audience

SHOW SLIDE "SCETRIS"

MY name is <first speaker> and this is <second speaker>. We are part of Team SceTris and we participated in the SCORE contest 2011.

We are 5 Students and we chose to work on the project "myCourses".

What you've just seen is the variety of software used at our university to manage the booking of courses. As you can imagine it's very complex – we needed 2 years to actually get an account in each of the just mentioned systems!

We thought we can do better than that, thus myCourses is the task that we chose.

SHOW SLIDE MYCOURSES

The initial requirements for myCourses stated, that a web-based resource management application should be built, allowing for scheduling of courses, both **collaborative** (thus web-based) and **computer-aided** – that is, the system should be able to create timetables etc. itself. EDUCATIONAL BACKGROUND

Besides these functional requirements, a strong focus was on usability, since the software should be used by non-technical staff and the average student. Also the software should be easily customizable, extendable, scalable – the usual stuff you present when selling software.

SHOW SLIDE SCREENSHOT OF SCETRIS

THIS is a screenshot of our application. As you can see, we adapted the corporate design of our university – which can easily be revamped thanks to XSL templates doing the layout. It actually shows the detail view of a specific room in the admin perspective.

SHOW SLIDE SCETRIS IN THREE ACTS

In the following we will present HOW we created scetris, the methods we used, the problems we faced. Along we will discuss a few technical issues. Due to the little time we won't be able to discuss every aspect of our project, thus we will focus on a few aspects

which were seem very important to us and steered our project into those directions which eventually led to the software because of which we're now here.

I REALLY DONT LIKE THE SLIDE, SIMPLY BECAUSE OF THE FACT THAT THE REST IS GREY. I THINK THEY SHOULD BE HIGHLIGHTED, MAYBE BY POINTING AT THEM WITH AN ARROW AND IN A DIFFERENT COLOR THAN TODAY.

SHOW SLIDE WITH MYCOURSES ----?----> SCETRIS

In the very beginning we asked ourselves, how could we manage to build scetris? What kind of development model should we use? What technologies? Who sould do what?

While we thought initially that an agile development model would suit us best, we decided against such a process, since large parts of our project would collide with our semester, such that we would not be able to follow such a process truly.

SHOW SLIDE WITH SPIRALS

In the end we decided to go with an iterative process, specifically the spiral model. Since the actual project would last about half a year we decided to do 5 iterations, 4 to 6 weeks devoted to each. In the first iterations we would try to do most of the specification, such that in the later ones only minor design changes should be necessary. The later iterations should be devoted to testing and validation more and more.

GEDANKEN ANDRÉ: WIRKLICH SCHNELL DURCH DIE ITERATIONEN DURCH GEHEN

Development process

ANMERKDUNG: teambildung!? streichen / Iteration 0 nicht diskutieren; uninteressant.

Iteration 1

The real work began in June 2010 with Iteration 1. As just mentioned, we wanted to devote it mostly to our initial requirements analysis. Besides we evaluated possible strategies for solving the course scheduling problem and looked at different technologies for creating a web-based application.

Since the automatic scheduling seemed to be a rather performance demanding task we decided to implement that part in Java. As for the web-interface we decided to use PHP.

During the process of requirements analysis we created a prototype, for checking whether it would be feasible to experiment with new technologies and approximate the amount of work needed to implement an automatic scheduler.

SHOW SLIDE WITH SCREENSHOT OF PROTOTYPE SCHEDULER RUNNING AS CLI

Very early in the stage of developing we decided to use a genetic algorithm for solving the course scheduling problem. In the screenshot you can see the prototype scheduler running using a simple command line interface.

The team was very fascinated by this totally new concept of genetic algorithms (we will talk about this in detail tomorrow in our poster session, **don't miss!**) we got a bit carried away and even thought, that creating a fully automatic scheduling would be possible.

SHOW NEXT SLIDE

And than there was this. This is one artifact from our first iteration (made up for this presentation). It's a UML sequence diagram which illustrates the process of scheduling the courses, as described in the project statement of myCourses. Thanks to **THIS** we realized, that this should not be what our application should become eventually.

Thanks to that requirements analysis in the very beginning we recognized, that in certain circumstances it might be needed to stop the process, allow for human interaction, etc.

Iteration 2

After the first iteration, guess what, the second followed. We had written use cases, decided about the architecture of our application, that we would use a template engine, etc.

SHOW model.xml SLIDE

However, as we were going to implement the scheduling component in Java and the web interface in PHP that it might become difficult to interface these two compoents with each other.

At that time in our project, we planned to use certain functions from the scheduling algorithm in the web component. For example, there is a fitness function contained in the genetic algorithm which can be used to calculate the quality of a certain scheduling. When manually editing the scheduling, that fitness function should have checked whether the edit would result in a better scheduling or not and inform the user about this.

Thus we decided, to drop PHP for our web interface.

Another problem which arose earlier was, that we needed to create a database access layer twice. Once for the scheduler, written in Java, once for the PHP web interface component, written in PHP ... . To tackle this problem we already had written a tool for generating both PHP Classes as well as Java Classes, from which the layout of our database was derived. Thanks to this technology we could easily switch programming languages without any fuss.

In the end we dropped PHP.

I THINK 18 AND 19 IS TOO MUCH.

SHOW SLIDE WITH FANCY PIE

Here you can see the amount of auto-generated code within our codebase and how much the generator takes up in it. The complete auto-generation toolbox was completed within two iteration and consumed one MP (as opposed to horse-power).

Iteration 3