Whiteboard over Web
Fortgeschrittenenpraktikum im Somersemester 2007

Jens Schönborn, Andreas Grüner, and Christian Motika

Lehrstuhl Softwaretechnologie
{jes,ang,cmot}@informatik.uni-kiel.de

1 General

This programming exercise deals with the implementation of a distributed graphical whiteboard application. It enables one to exchange ideas with other people in the web by means of a shared canvas. The tool shall be developed in groups (number of groups depends on number of participants).

Section 4 deals with the separate programming modules. The evaluation method for this programming exercise can be found in Section 6, and the documentation requirements can be found in Section 5. In Section 2, the schedule for the exercise is given and some advice on some topics regarding this course is given in Section 7. In section 3 some existing programs which show parts of the required functionality are presented.

2 Schedule

Number	Date	Name	Meeting required
1.	5.4.	General organization	X
2.	13.4.	Arrangement of groups and assignment of tasks within the groups	Mail
3.	19.4.	Presentation of specification draft	X
4.	24.4.	Final specification and working plan	Mail
5.	26.4.	Presentation of final specification and working plan	X
6.	1.6.	Progress report including documentation	Mail
7.	7.6.	Presentation of progress report	X
8.	3.7.	Terminated feature report	Mail
9.	12.7.	Code deadline, final report including documentation	Mail
10.	19.7.	Final presentation	X

Participation in required meetings is mandatory. If common sense is agreed, different dates can be arranged, but only during the first 10 days. Thereafter the deadlines are strict. Mails have to be sent until 14:00 on that day.

2.1 Organization

It is mandatory in order to get a Schein that you register in the “Studierendendatenbank” and enlist to the course *Vertiefende Übung ”Programming in the many” - Fortgeschrittenenpraktikum* number *080115* for 8 SWS and number *080111* for 4 SWS. Further

information can be found here:
`http://www.informatik.uni-kiel.de/ifi/studium-lehre/studidb/`.
Subscribe to the mailinglist
`https://lists.informatik.uni-kiel.de/mailman/listinfo/p-softtech`.

We encourage the participants to contact the organizers in case of questions of any kind, or ideas concerning the project (e.g., about additional features).

– Every Tuesday from 10:00 to 11:00 at room 1210 Christian is available especially for technical questions (Java, programming environment, etc.).
– For other questions every group may request a meeting (though not the whole group has to attend it) by mail with Jens or Andreas.

3 Examples

Laeqed-Latex equation. Laeqed is a Latex equation editor specifically targeted at producing PNG images of math equations for use on web pages.
`http://thrysoee.dk/laeqed/`

Jarnal-Notetaking Software. Jarnal supports collaborative writing; a feature that allows multiple authors to edit a single file in real time over a network. The english wikipedia
`http://en.wikipedia.org/wiki/Comparison_of_notetaking_software`
provides an overview of some other, probably not all support collaborative writing, Notetaking Software.
`http://www.dklevine.com/general/software/tc1000/jarnal.htm`

iText-a Free Java-PDF library. iText is an Open Source library for creating and manipulating PDF.
`http://www.lowagie.com/iText/`

4 Modules

In this section we provide you with a more or less disjoint set of modules containing tasks that shall be assigned self dependent to the participants. Each module scores an individual number of SWS-points so that one may have to take over several modules to obtain enough total SWS to get a Schein over either 4 or 8 SWS at the end. We encourage the students to make further suggestions.

4.1 Coordinator

A person responsible for delivering the common parts like the working plan. He has to write the common documentations and give respectively organize the common presentations. Here common means the non single module dependent issues. The coordinator is also responsible for managing the communication within the group and check the source code for proper comments. He has to write and maintain a *make*, or *ant*, or . . . file for the project.

Value: 4 SWS

4.2 Data structure implementation

The common data structure has to be implemented with efficient algorithms. Further, a file representation of this structure (including save-to-file and load-from-file operations) must be specified and implemented.

Value: 4-6 SWS

4.3 Graphical user interface

Implementation of graphical input facilities for the whiteboard. Modifications should be possible at any time. Furthermore, this module provides the typical user interface (load, save, exit, undo/redo, . . .), for instance, in form of a menu bar. Implementation of a facility to use pdf or postscript-files as a background of the whiteboard and the function to generate PDF output.

Value: 10-16 SWS

4.4 Latex commands

Functionality for displaying output of latex code (esp. math) shall be provided. The users shall be enabled to enter his own macros resp. provide macro files. In case this yields missing commands at other clients the corresponding macros resp. macro files shall automatically be shared. As fallback solution the output shall be shared as a graphic.

Value: 4 SWS

4.5 Network

Implementation of a network protocol which ensures that every participant has the same view of the whiteboard. This includes that every draw activity of one participant has to be broadcast to the other ones. A chat function has to implemented.

Value: 8 SWS

Optional: Users shall be able to send arbitrary files to each other.

Value: 2 SWS

5 Documentation requirement

Writing full length project reports is not part of this exercise. However, four of the deadlines include a short written report (2-3 pages each, the terminated feature reporrt can be shorter) about the group's progress.

– Final Specification and working plan should include:

    - detailed responsibility assignment (preferably including comments on group organization)
    - schedule (e.g. using Gantt Chart) including prognosis of work that should be finished until progress report
    - detailed interface description
- Progress report should include:
    - revised schedule, showing previous as well as remaining work
    - progress description of individual modules incl. outlook on remaining work
    - revised interface description (if changed since working plan)
    - unexpected problems that might have occurred and how they have been solved
- Terminated feature report should include:
    - a list of of the contained features for every group (the feature list may not be modified afterwards)
    - a short description of every feature
- Final report should include:
    - finished, partly finished or unfinished tasks[1]; with the following information
        - time of the beginning of the work,
        - if available, time of reaching an testable stage,
        - if available, time of the end of the work,
        - if available, the reason why the task was not finished
        - persons involved in this task (with an estimation of the expenditure of time)
    - Which points of the final specification/progress report/terminated feature report where amended?
    - updated interface description if it has changed
    - Noticeable problems, which arose during the project: Which tasks were more laborious than expected, which less? Which problems emerged and how could they be solved or not solved? Which special design decisions were made? Did the programming language prove suitably? What shall be done differently next time? Etc.

5.1 Presentation

Here are a few tips about presenting you probably already know, but experience shows you can never say it often enough:

- be prepared – do not just hope everything will work, make sure it will (as good as you can)
- know your work – learning the speech by heart is not enough, you have to know what you are talking about
- stay focussed – don't get carried away by unnecessary details
- know your audience – do not waste time for telling things everyone already knows, but do not leave important things out the audience probably does not know
- rehearse your presentation before you give it – not only will it save you a lot of embarrassment, without rehearsing it you will not know what time it takes to give it
- rehearse your presentation before you give it – we mean it!

[1] This also includes, e.g. administrative tasks, not only the programming tasks.

6 Evaluation

The evaluation is made at the end of the course depending on the delivered code, documentation, and presentations. The final code is most important. We also expect interaction between the participants (including organizers) in order to maximize the success of the resulting software project. **Deadlines are strict** (once they have been agreed upon). We expect reasonable comments in your code explaining non-trivial parts of code. We recommend to make use of
javadoc (`http://java.sun.com/j2se/javadoc/`).

7 Advice

7.1 About groupwork

To learn something about successful groupwork regarding software projects, the following essay is recommended, not only for this project but for all future group assignments:

`http://www.csc.calpoly.edu/~sludi/SEmanual/TableOfContents.html`

7.2 Tool recommendations

The implementation has to be made in *JAVA SE 5.0* We recommend to use a programming environment like
eclipse (`http://www.eclipse.org/`).
We recommend to use *lint4j* for code optimization.

7.3 Subversion

- Access to the Subversion repository
  In order to get an account for the projects repository an account for the RBG-Rechnernetz is required. The corresponding user name should be included when enlisting in the list of participants on April 5th. If you were prevented to do so for any reason, please send a mail with the missing information to `jes@informatik.uni-kiel.de` as soon as possible. The passwords will be handed out on Wednesday resp. Thursday 11th resp. 12th April. With your password and username you will be able to visit
  `https://snert.informatik.uni-kiel.de`. Attention: please accept the ssl certificate although it is issued by an unknown authority.
- There is a good book ”Version Control with Subversion” at
  `http://svnbook.red-bean.com/`
- Initial Checkout
  Checking out a repository creates a copy of projects repository on your local machine in directory wow. Attention: please accept the ssl certificate although it is issued by an unknown authority. This working copy contains the latest revision of the repository that you specify on the command line:

  ```
  svn checkout https://snert.informatik.uni-kiel.de/svn/wow
  ```

  After that, entering subversion command line client commands within the working directory does not require the url anymore.

- The typical work cycle using Subversion
  - Update your working copy with `svn update`
  - Change, add, delete, copy and move files with `svn add <file>`, `svn delete <file>`, `svn copy <file>` and `svn move <file>`
  - Examine your changes with `svn status`, solve conflicts and merge others' changes into your working copy `svn update`
  - Commit your changes with short comment
    `svn commit -m "<your comment>"`