Praxis der Software-Entwicklung

Development of a Campus Routing System

SS 2013

Institute of Theoretical Informatics
Prof. Dr. Dorothea Wagner

1 Important Preliminary Remark

This is *your* project. This document is not a checklist of exercises that should be processed point by point to obtain the credit. It is just a list of suggestions and hints of what you can do and what we expect you to do. It is *your* decision what *your* software will look like.

2 Task

The general task of this project is the design and implementation of a routing system for the KIT campus. The system should be able to compute and to display a shortest path between a starting point and a destination. It should be possible to route from and to buildings and to rooms inside buildings. For the latter, the system should display a floor plan of the building. It should also be possible to search just for the destination, without computing a route from some other point.

On the administration side, your system should provide a graphical user interface (GUI) to modify the underlying map data. It should be possible to easily add/delete buildings with or without routing information of the interior. Moreover, it should be possible to add vertices and edges with information that are necessary to do the routing.

2.1 Routing

The routing starts with the user input. It should be easy for the user to specify a building, a room or a lecture hall as starting point or destination. Your system should use Dijkstras algorithm to compute a shortest path from the starting point to the destination.

Which path is the shortest of course depends on the metric you use. A reasonable default metric could be a constant pedestrian walking speed.

2.2 Map Data – Administration Tool

You should use your administration tool to create the map data. As a basis for the map of the exterior you can use the KIT campus plan (`http://www.kit.edu/study/3250.php`). On top of that, you have to add the graph for the routing. Apart from the graph itself you probably need additional data. For example the entrance of a building should be tagged with the name of that building so that the user can use this tag to route to the building. You should also create the data for the interior of at least two buildings.

Make sure that your administration tool is consistent with the routing system. That is, it should be possible to add and edit every piece of data that is or can be used by the routing application.

If you use the number of stairs of a staircase to estimate the time necessary to go up or down the staircase, then your administration tool should support adding this information.

2.3 Displaying the Map

The simplest way to display the map is using an image file. On top of this image, you can draw the computed path (and perhaps additional information you want to display). If the computed route starts or ends inside a building, you should also display the floor plan of this building together with the computed route inside the building.

As the whole map might be too large to fit on the screen, you should add the possibility to pan and zoom.

2.4 Suggestions for Cool Features

The following list contains some suggestions for possible features. We encourage you to think about features not contained in this list.

- It might be interesting to compute more than just one route, based on different metrics. Some people use stairs, some others rather use the elevator. And what about using the bridge instead of crossing the “Adenauerring”?
- When routing to a building (or lecture hall) with more than one entrance, it would be nice to route to the nearest entrance possible and not to the main entrance.
- It would be nice to have a Java applet of the routing system.
- The shortest path from the “Mensa” to the lecture hall “HSaF” probably uses the gap in the computer-science building (50.34). It would be nice if it would not look like going straight through the building.
- It could be nice to be able to temporarily close some roads that are under construction.
- Is it possible to go through a building, if this is shorter than going around? Where do you display the interior of a building in such a case?
- Is it possible to use the tram for parts of the route?
- . . .

3 Working Process

3.1 Operating Principles

In this course you should use software design and quality assurance methods in practice, use and improve implementation skills, and collaborate in a team distributing the work among its members. Accordingly, the organization of this course is guided by software engineering principals, as you can also see in the schedule in Section 3.4.

It is not the aim to somehow implement a system that somehow works, but to go step by step through the phases of software engineering as prescribed by the schedule. You do not only have to hand in a working system at the end but all documents corresponding to each phase:

- Functional specifications document (dt. Pflichtenheft; about 20 pages)
- Software design (dt. Entwurf; about 40 pages)
- Implementation report (about 20 pages)
- Test report (about 20 pages)

At the end of the course you have to present your finished and working system. Moreover, a colloquium takes place at the end of each phase. The particular appointments will be arranged during the semester.

The schedule in Section 3.4 is not just a suggestion to help you developing your system, we expect you to follow it. To this end, in each phase a *team leader* is determined who is responsible for a correct execution of this phase and for submitting the documents in time.

3.2 Documentation

Creating a documentation is a major part of your project. Without a complete and extensive documentation of your work it is not possible to successfully pass this course.

We expect you to document your planning and the execution of your plans precisely, completely, and consistently. Concerning the content and the form of the documents we explicitly refer to the software engineering lecture.

You must set up and use a version management tool for both, the source code **and** the documentation. We recommend to use Subversion or GIT.

3.3 Recommended Tools

The recommended programming language for this course is Java (version 1.6). Using a different language is only in exceptional cases and after agreement allowed.

The JUnit Testing Framework[1] is the de facto standard tool for unit tests when using java. We strongly recommend to use this tool. Basic literature on how to handle JUnit can be found on the web or in the library. The tool CodeCover[2] helps you computing your code coverage. Both tools are integrated as plug-ins into the development environment Eclipse[3] that we recommend to use as IDE. For the software design it is recommended to use an UML-tool like for example ArgoUML[4].

3.4 Schedule

- **First group meeting:** 24th of April, 2013
- **Delivery of the functional specifications document:** (29.04–19.05). Shortly after this you are required to explain your document in the first colloquium (date for this colloquium by arrangement).

[1]`http://junit.org/`
[2]`http://codecover.org/`
[3]`http://www.eclipse.org/`
[4]`http://argouml.tigris.org/`

- **Delivery of the software design document:** (20.05–16.06). Shortly after this date you have to defend your software design in the second colloquium (date for this colloquium by arrangement).
- **Implementation:** (17.06–14.07)
- **Delivery of the implementation report:** (17.06–14.07), shortly after this date there will be a colloquium where you need to explain your implementation report (date for this colloquium by arrangement).
- **Break (exams):** from 15th to 28th of July
- **Delivery of the test report:** (29.07–18.08)
- **Final presentation:** (09.09–15.09)

Please note that this schedule is only preliminary. It is possible that we change some of the dates slightly. If this is the case we will make an announcement early enough in advance.

3.5 Contact Information

- **Thomas Bläsius**, Institute of Theoretical Informatics, Prof. Dr. Dorothea Wagner. Email: `thomas.blaesius@kit.edu`, tel.: 0721 608-44322, office: room 316, building 50.34.
- **Tamara Mchedlidze**, Institute of Theoretical Informatics, Prof. Dr. Dorothea Wagner. Email: `mched@iti.uka.de`, tel.: 0721 608-44245, office: room 307, building 50.34.

4 Grading

4.1 Minimum Capability Characteristics

Your system should consist of two major parts, namely the administration tool to create the map data and the routing application for the user. These two parts of your system should have at least the following features.

Administration Tool The administration tool should support the following operations.

- Exchanging the picture of the map.
- Adding, deleting and moving vertices and edges.
- Modifying properties of edges, such as the travel time (although there should be a default value for this, depending on the length).
- Modifying properties of vertices, such as room number or name of the lecture room.
- Adding the floor plan of a building.

Routing The routing application should have the following features.

- User-friendly and self-explaining GUI, that makes it easy to input starting point and destination.
- Fast computation of a shortest route ($<$ 1 second).
- Nice visualization of the computed route (indoor and outdoor).

4.2 Reference System

Your software must be able to run on a Linux PC. For testing purposes there are Linux PCs (with Suse 11.3) available in room 304. These PCs are equipped with an Intel E7200@2.66 Ghz processor and 4GB of RAM.

4.3 Grading

The grade for your project depends on the following criteria.

- Quality of all your delivered documents.
- Quality of all colloquia.
- Meeting all of the minimum capability characteristics (see Section 4.1).
- Useful extensions to the software that surpass the minimum capability characteristics.
- Consistency between the two parts of your system.
- Robustness of your software.

This list has *no* specific order. In particular the order of this list *does not* reflect any weighting of the criteria for the final grade.

The grade for the team-work in this practical course is a grade for the so called *soft skills*. The most important aspects for these skills are the presentations you give, how well you work as a team and the performance of the team leader during each phase.

For passing the practical course it is required that all documents mentioned in Section 3.1 are delivered *on time*. For a schedule see Section 3.4.

Each phase of your project is graded. The grades of each phase will be combined for the final grade of your project. See Table 1 for a detailed overview on how much the grade for each phase influences the final grade for the project.

Phase	**Weight**
Functional specifications document (dt. Pflichtenheft)	10%
Software design (dt. Entwurf)	30%
Implementation report	30%
Test report	20%
Final presentation	10%

Table 1: Weighting of the grade for each phase.