Distributed Programming II

A.Y. 2018/19

Design and implementation of a simple vehicle tracking service (max 2 students)

General Description of the Project

The aim of this project is to design and implement a RESTful web service that can track the presence of vehicles in an area with restricted access and, based on the tracking information, can provide permission to new vehicles to enter the area and suggestions about the path to their destinations. The service has to use a data model similar to the one used for the DP2 assignments, but more complex (for example, there will be the possibility to have nested places). When a new vehicle wants to enter the area, it has to ask the service for permission, specifying both the entry point (i.e. its current position) and the desired destination in the area. The service will check if it is possible to allow the entrance of the new vehicle, based on the area model (which can include a number of constraints, such as capacity of parking areas and of roads) and based on the current and expected future positions of the other vehicles in the area. If the service allows entrance, it responds by also indicating a suggested path that the new vehicle is expected to follow from the entry point to the destination, and a unique identifier assigned by the service to the new vehicle. While the vehicle is moving to its destination it will periodically send information about its current position to the service, which will update its tracking information. If the service realizes that a vehicle is not following the suggested path, it will compute an updated path that starts from the current vehicle position and will send it back to the vehicle in response to the position tracking message. The area model must be specified in an XML document, for which a schema has to be designed. The service must upload the area model from the XML file at startup and it must allow administrators to collect various kinds of information about the vehicles currently in the area and their expected routes.

A documentation of the XML formats and of the web service, a client for testing the service, and an ant script that automates compilation and running of the various programs and tests must also be provided as part of the solution.

Deadline: **The end of the February exam session**

Additional Specifications

1. The specifications about the data model given for the DP2 assignments are still valid. In addition, the following extra specifications must be considered:
    a. Places can be nested. There are basically two kinds of places: simple places (e.g. gates or road segments) and places that are containers of other places (e.g. a road, which is a collection of road segments, is also a place). The current position of a vehicle is always a simple place, but, as a simple place can be part of a larger place, the vehicle is also in the larger place (e.g. when a vehicle is in a road segment of a road, it is also in the road).
    b. A simple place is also characterized by the average time spent by vehicles in transit

in it.

   c. Vehicles can carry dangerous materials. There must be a catalog of dangerous materials and the catalog must include the information about which dangerous material must not stay close to which other dangerous material.

2. The data model must be specified by means of UML diagrams (extending the one provided with the assignments), and by means of XML schema. The semantics of the data must be documented in the data model documentation.
3. The documentation of the REST service for vehicle tracking must include all the information that is necessary to develop clients.
4. Both service and client must be implemented in Java. Additional software libraries not mentioned in these specifications can be used provided they are open source, and only after discussion with the teacher.
5. The REST service must be deployed in Tomcat.
6. The REST service implementation should be based on an external tool that can solve optimization problems with constraints. We suggest Microsoft's z3opt, which has a Java API, and for which we can provide support.
7. When a vehicle wants to enter the system, it must specify its position (an input gate) and it must ask permission to go to a destination, specified by the vehicle (the destination can be any place). Similarly, a vehicle parked in a place can ask permission to move to another place. The service grants permission to the vehicle if it can find a path to the destination that satisfies the constraints about maximum number of vehicles in each place and about dangerous materials (a vehicle carrying a dangerous material must not pass in the same place where vehicles carrying incompatible materials can pass). In order to satisfy the constraints, for simplicity, the service will simply make a reservation on the places that are on the path to the destination. As far as the vehicle reaches and then leaves the places on its path, the reservations are removed. Of course, the reservations cannot exceed the allowed number of vehicles in each place and cannot create conflicts regarding dangerous materials.
8. Whenever a vehicle enters a new place, it contacts the service and communicates its new position.
9. The service must provide a way to get the current status of places (what vehicles are in each place and what reservations have been made on the place)
10. The client of the REST service must be either an android mobile app or a web application. In any case, the client must be able to simulate the behavior a vehicle that enters the system and navigates to its destination, and get information about the status of places. The client should be user-friendly, possibly with a simple graphical interface.
11. A set of tests must be developed to test the service and to test the client. The tests for the service must be automated.
12. An ant script must be provided to build client and service, to deploy/undeploy/redeploy the service, and to run automated tests. Documentation about how to use the ant script must be provided.
13. A final report that documents the project must be produced with the following contents:

a. Data model documentation
b. REST service documentation
c. Client documentation
d. Instructions for building, running, and testing the service