UNIVERSITY OF TARTU

MTAT.03.083 – Systems Modelling

Regular Exam – 9 January 2018

Notes:

- The exam is open-book and open-laptop. Web browsing is allowed.
- You are not allowed to communicate with anyone during the exam in any way (except with the lecturer).
- You may submit part 1 of your exam on paper, or electronically (in the form of a pdf file).
- You may submit part 2 of your exam on paper, or electronically (in this case, please submit: (1) The "lamp.sct" file, and (2) a picture showing the statechart with your solution. There is no need to submit the whole project. The "sct" file and the picture of the statechart are sufficient).
- Include all files of your submission in a zip file and submit it using the "Submit" button in the course Web page.
- If you find that there is not enough information in the text below and you need to make additional assumptions, please write down your assumptions.

Part 1. Street Networks and Itineraries

Task 1. [10 points] Write a domain model (UML class diagram) of a *street network*. A street network is a sketchy representation of the set of streets of a given geographic area. For example, a street network of an area of Tartu is depicted in Figure 1.

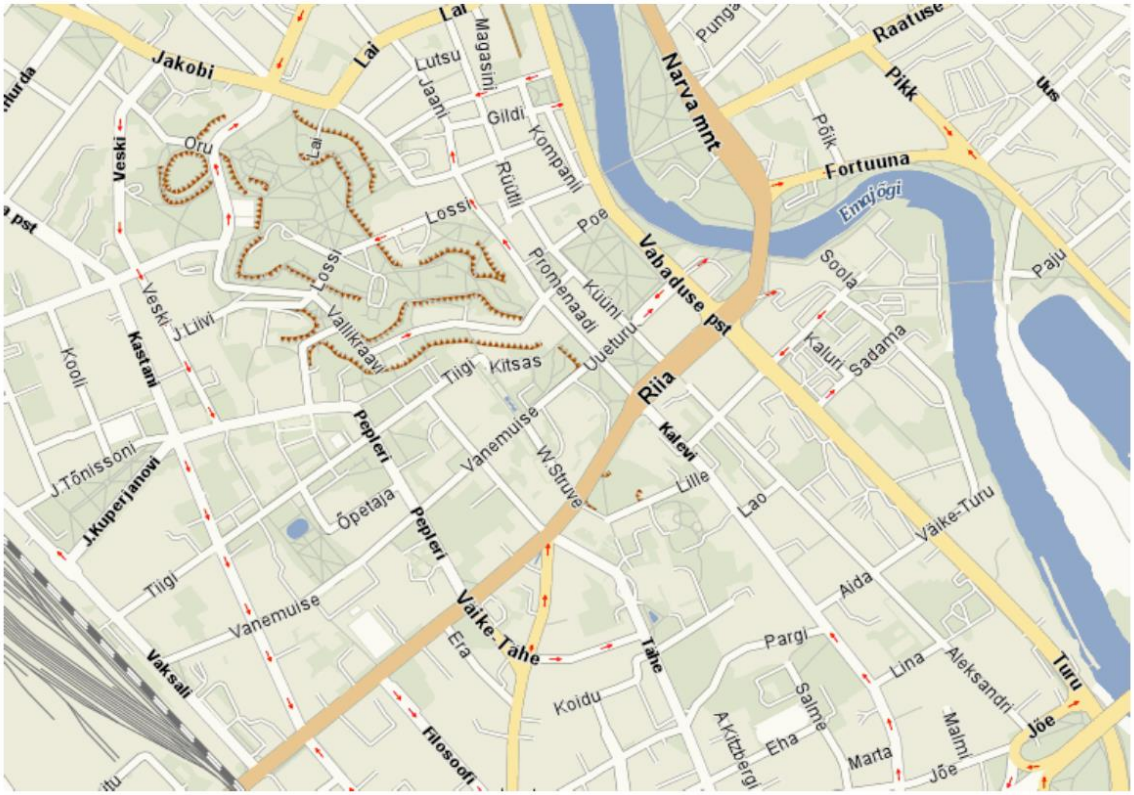


Figure 1 Street network of Tartu's city centre and surrounding areas

A street network consists of *street segments* and *junctions*. A street segment is a fragment of a street between two junctions. By default, traffic in a street segment can go in both directions. However, there are some street segments where vehicles can only go in one direction (one-way segments).

A junction is a point at the intersection between two or more street segments. The geographic location of a junction is determined by a GPS coordinate. Junctions can be of several types depending on the mechanism that controls vehicles and pedestrians crossing the junction. In this respect, we will consider three types of junctions:

• plain (no mechanism to control crossings)
• roundabout
• traffic light

The crossing of a junction is also controlled by ``give way'' signs. The *end of a street segment* at a given junction may (or may not) have a ``give way'' sign. Sometimes a street segment has a ``give way'' sign at one of its ends, but not at its other end.

Task 2. [5 points] Extend the previous domain model to capture the notion of an itinerary (you can submit your solution to this question as a separate domain model, or as an extension of the domain model of question 1.). An itinerary describes how to get from a given street segment to another street segment in a street network. In combination with the street network, an itinerary contains sufficient information to determine what to do at each junction (e.g. which street segment to take next) in order to ultimately reach the destination. For example, given the network in Figure 1, a possible itinerary from the train station at the end of J. Kuperjanovi street to the only segment of Filosoofi street is to first take two segments along Vaksali street (from the junction with J. Kuperjanovi to the one with Tiigi and from there to the junction with Vanemuise), then take one street segment along Vanemuise street, then one street segment along Kastani street, and then take the Filosoofi street segment, where the itinerary ends.

Task 3. [10 points] Extend the domain model(s) (you can submit your solution to this question as a separate domain model, or as an extension of the domain model of question 1 or question 2) of the previous questions in order to capture the average travel time required to cross street segments and junctions of a street network at different time points during a day (from 00:00 to 23:59). The domain model should capture enough information to determine, for any given time point in the day, how much time it takes to cross a street segment by vehicle from one of its ends to the other. The travel time required to cross a street segment can be different depending on the direction, meaning that at a given time point during the day, crossing a street segment by vehicle from junction 1 to junction 2 may take more time than crossing the same street segment from junction 2 to junction 1. The travel time required to cross a street segment varies depending on the time of the day (e.g. it can take 60 seconds when starting at 14:05 and 80 seconds when starting at 16:10). The function that assigns a time point in the day to the crossing time of a street segment (for a given direction) is a step function, meaning that for certain intervals of time (e.g., between 14:00 and 14:30 it is 60 seconds), the average street crossing time from one end to the other is constant and then it suddenly changes value and remains again constant for a subsequent interval of time (e.g. at 14:31, it goes up to 80 seconds and stays like that until for example 16:30).

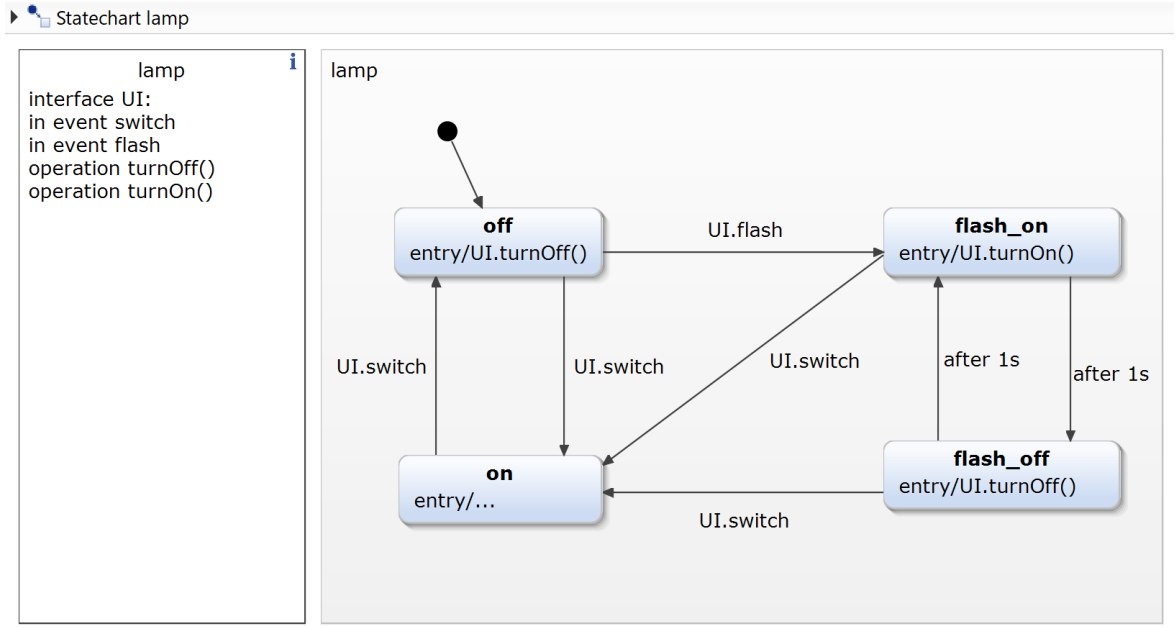
The extended domain model should also capture the amount of time required to cross a junction at any given point of time in the day. The average time required to cross a junction depends on the origin and destination street segment. For example at the junction connecting Tähe street and Riia street, at a given time of the day, it may take 20 seconds to cross from one of the street segments in

Tähe to one of the street segments in Riia, and 15 seconds to do so the other way around. The time to cross a junction from the end of a street segment to another varies depending on the time of the day, in a stepwise manner as for street segments.

Task 4. [10 points] Write a sequence diagram showing how objects of the classes defined in the above domain models (and possibly other classes as required) can interact in order to calculate the average time it takes to drive by vehicle across an itinerary, starting at a given time of the day. Assume that the vehicle will entirely cross the first and the last segments in the itinerary.

Part 2. Statechart of a Flashing Lamp

The starting point for this question is the statechart (you find the Yakindu project, which contains an application that displays a lamp here: https://www.dropbox.com/s/5m49ig2zkqjoo01/lamp.zip?dl=0):



The lamp can be switched on and off using the "switch" button. While the lamp is off, pushing the "flash" button makes the lamp flash. The flashing can be stopped by clicking on the "switch" button. The lamp then comes back to a state where it is on but not flashing. The controller of the lamp is implemented as a statechart found inside the Yakindu project. (Technical note: Once unzipped, the project can be imported into Yakindu by using the option: *Import --> General --> Existing project into workspace*. When running the Java app (Main file), you might need to increase the size of the window to see both buttons.)

Task 5. [15 points] Extend the statechart in order to implement the following feature: When the lamp is flashing and the button "flash" is pressed a second time, then the lamp starts flashing twice faster than normally (moving between states every half a second). Pushing flash a third time makes the lamp flash normally again (moving between states every second), pushing a fourth time makes the lamp flash faster again, and so on. The flashing can be stopped by pushing the switch button, in which case the lamp comes back to the state where it is on but not flashing. The flashing can be stopped both when the lamp is flashing fast or flashing normally.

The solution will be graded both in terms of correctness as well as simplicity and modularity of the solution.