UNIVERSITY OF TARTU

MTAT.03.083 – Systems Modelling

Regular Exam – 12 January 2015

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 your exam on paper or electronically. In the latter case, include all files in a zip file and submit it using the "Submit" button in the course Web page. Allowed file formats are: PDF, PNG, JPEG, PNML (Woped), and mdzip (MagicDraw).
- 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

Tasks 1, 2 and 3 can be submitted as a single UML diagram or as separate diagrams.

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.

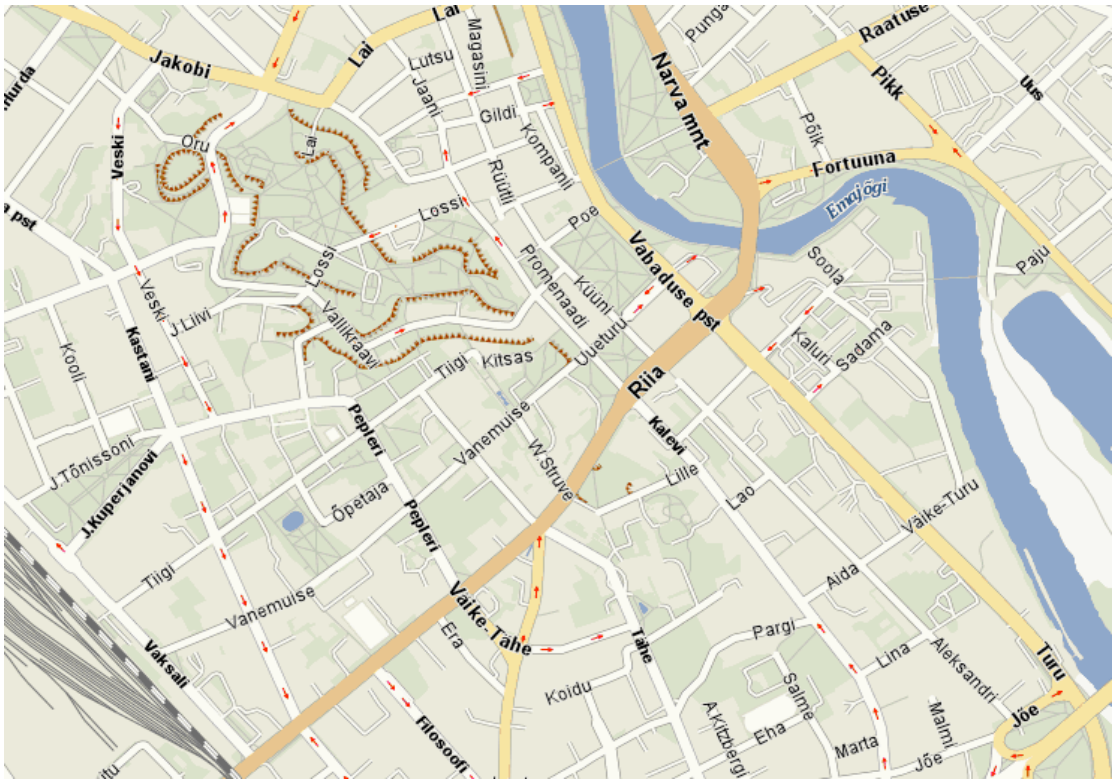

Figure 1 Street network of Tartu's city centre and surrounding areas

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*.[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)[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.

[1] You can submit your solution to this question as a separate domain model, or as an extension of the domain model of question 1.

[2] 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.

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. Automated Factory Line

We consider a segment of a factory with two conveyor belts, two machines, one robot and one buffer. Raw parts arrive through a first conveyor belt, called the *raw line*. The robot moves each part from the *raw line* into machine M1. Machine M1 immediately starts processing the raw part. Eventually, the machine will finish processing the part and the robot may then move the processed part into the buffer. Eventually, the robot moves the part from the buffer into machine M2. Machine M2 will start processing it. When machine M2 finishes processing the part, the robot will then move that part from machine M2 into a second conveyor belt (called the *finished line*).

Figure 2 depicts the factory line in the state where the raw line holds 2 parts, M1 and M2 hold one part each, the buffer holds two parts, and the finished line holds one part.

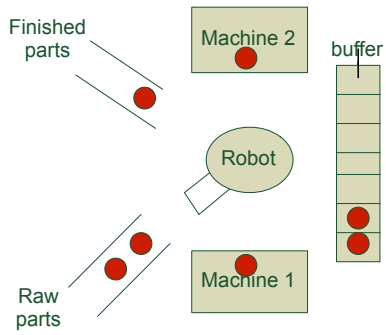

Figure 2: Factory Line Structure

M1 can hold at most one part at a time, and the same applies for M2. The robot can only move one part a time. When machine 2 is free, the robot can move a part directly from machine 1 into machine 2 instead of moving the part to the buffer. The conveyor belts can hold any number of parts but the buffer can hold at most 7 parts.

This segment of factory line (in its empty state) is modelled as a Petri net in Figure 3.

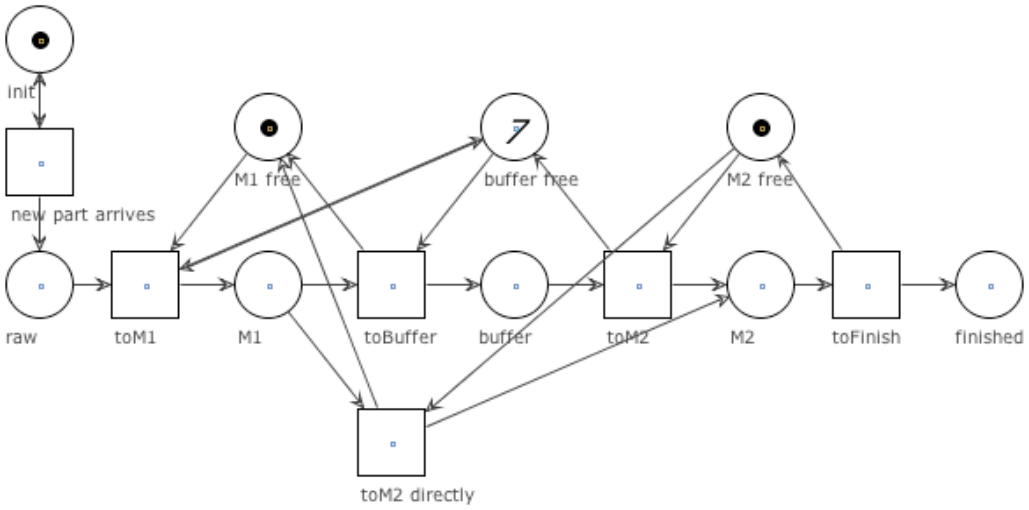

Figure 3: Petri net of the system in its initial (empty) state.

Task 5. [15 points]

Submit this task as a single PNML file exported from WoPeD. Both sub-tasks (a) and (b) should be done on the same Petri net.

a) Modify the above Petri net in order to capture the following requirement: A raw part can only be moved to machine M1, if there is at least one available slot either in buffer or in M2. This constraint is intended to avoid the situation where machine M1 finishes processing a part and the robot is not able to take away the processed part from M1. **[5 points]**

b) Modify the Petri net by adding a new machine M3 and a new buffer called "buffer2". When M2 finishes processing a part, the robot may move the processed part into buffer2. Eventually, the robot moves the part from buffer2 into machine M3. When machine M3 has processed a part, the robot moves that part from machine M3 into the *finished line*. M3 can hold at most one part at a time. When machine 3 is free, the robot can move a part directly from machine 2 into machine 3 instead of moving the part to buffer2. Buffer2 can hold at most 7 parts. A part can only be moved to machine M2, either directly from M1 or via the buffer, if there is at least one available slot either in buffer2 or in M3. **[10 points]**