Smart Space-Driven Sustainable Logistics: Ontology and Major Components

A. Smirnov, A. Kashevnik, N. Shilov
SPIIRAS
14 Line, St.Petersburg, 199178, Russia
{smir, alexey, nick}@iias.spb.su

H. Paloheimo, H. Waris, S. Balandin
Nokia Research Center
Itämerenkatu 11-133, 00180, Helsinki, Finland
{Harri.Paloheimo, Heikki.Waris, Sergey.Balandin}@nokia.com

Abstract

Today, ridesharing is attracting more and more attention due to its high potential in reducing the greenhouse gas emissions and decreasing the amount of traffic in the streets. The paper extends the earlier presented approach to building sustainable logistics system for ridesharing support based on the idea of smart spaces, where various electronic devices can seamlessly access all required information distributed in the multi-device system from any of the devices. In particular, the paper describes the ontology and major components for the ridesharing system. The Smart-M3 open source platform was chosen as a basis for the proposed approach.

Index Terms: ridesharing, sustainable logistics, ontology, behavior scenario, logic.

I. INTRODUCTION

Ridesharing (also known as carpooling, lift-sharing and covoiturage), is the shared use of a car by the driver and one or more passengers, usually for commuting. Dynamic ridesharing (also known as instant ridesharing, ad-hoc ridesharing, real-time ridesharing or dynamic carpooling), denotes a special implementation of a ridesharing service which enables the formation of carpools on very short notice. Typical for this type of carpooling is: arrangement of one-time trips instead of recurrent appointments for commuters; the usage of mobile phones for placing carpooling requests and offers through a data service, automatic and instant matching of rides through a network service.

In accordance with Global GHG Abatement Cost Curve v 2.0 [1] in the travelling sector the carbon emission can be significantly decreased via more efficient route planning, driving less, switching from car to rail, bus, cycle, etc. Additionally, the fact that the transport sector is 95% oil dependent makes it vulnerable to the expected rise in oil price during this decade [2], [3]. As a result, evolving of flexible, ecological and energy efficient logistics systems can be considered as one of the significant steps towards the knowledge-based Green ICT applications in low carbon economy. Above regard a growing number of initiatives has been developed in this area (e.g. Figure 1 represent activity of major vehicle manufacturers in the carsharing market). Carsharing assumes renting cars for one ride. This principle differs from the ridesharing, however, the figure clearly represents the growing demand and interest in this area, especially in Europe, where daily travel distances are relatively short.

Modern ICT make it possible to combine several ideas, which would result in a more flexible and efficient transportation systems. The main idea of the proposed approach is to develop models and methods that would enable configuration of resources for decision support in ad-hoc sustainable logistics. In the proposed approach the dynamic ridesharing for passengers as well as for cargo is considered. The paper extends the research results presented at AIS-IT'10 [4] presenting the developed ontology and major components of the dynamic ridesharing system.

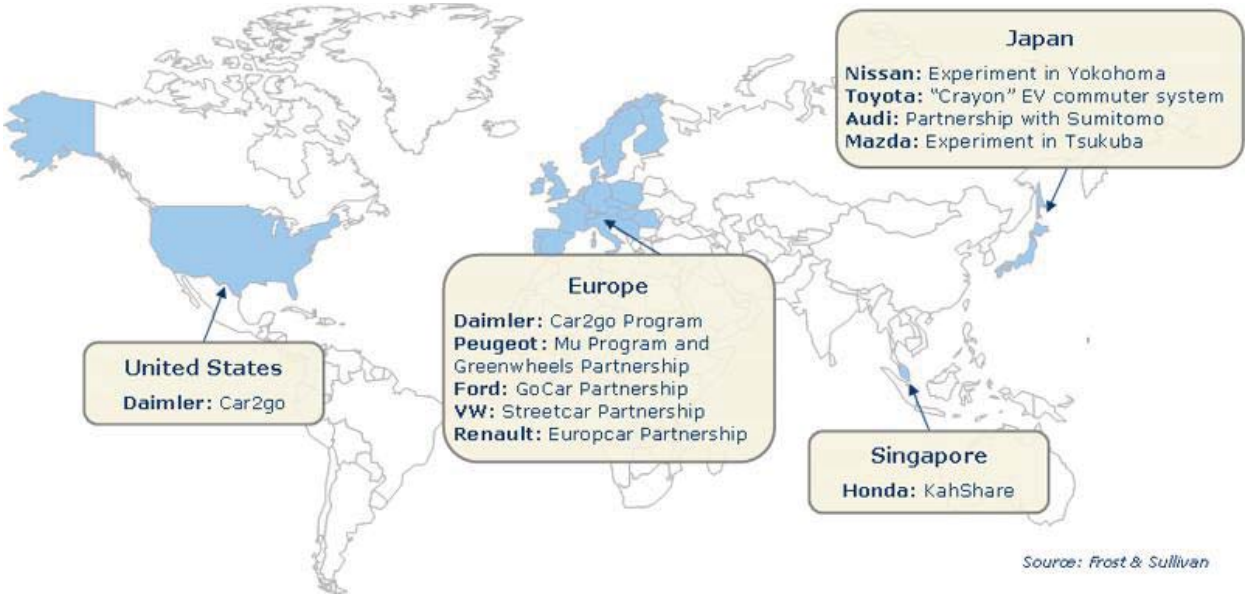

Figure 1. Major vehicle manufacturers in the carsharing market (World), 2009

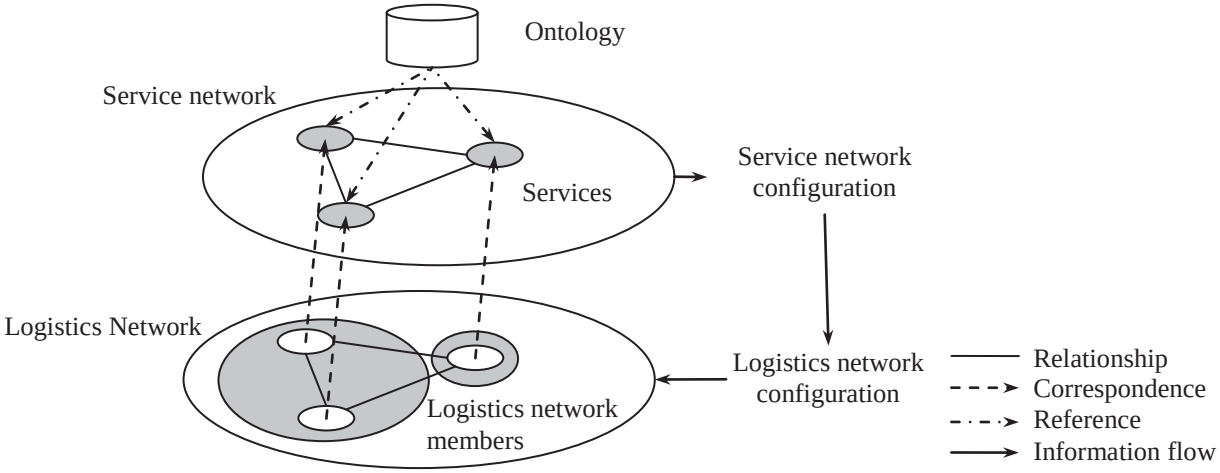

Figure 2. Generic scheme of the approach

The paper is structured as follows. The technological framework is described in sec. II. The ontology used for logistic operations is presented ion sec. III. Logic of the main components of the ridesharing system is presented in sec. IV. Major results are summarized in conclusion.

II. TECHNOLOGICAL FRAMEWORK

Figure 2 represents the generic scheme of the approach. The main idea of the approach is to represent the logistics system components by sets of services provided by them. This makes it possible to replace the configuration of the logistics system with that of distributed services. For the purpose of interoperability the services are represented by services using the common notation described by the ontology.

For the purpose of implementation the Smart-M3 platform developed by Nokia Research Center (Helsinki) is used [5]. The key idea in Smart-M3 is that devices and software entities can publish their embedded information for other devices and software entities through simple, shared information brokers – a "push"-based information sharing model rather than specific publish-subscribe. The understandability of information is based on the usage of the common ontology models and common data formats. Another key idea is that Smart-M3 is device,

domain, and vendor independent. It is free to use, open source solution available in BSD license. So, Smart-M3 refers to a piece of software technology, a number of software products encoding this software technology, a computing platform that the software products make available and any computing system that has been developed and deployed by using this platform [6].

Figure 3 represents the architecture of the prototype. Knowledge processors (KPs) represent participants of the system usually running on mobile devices of the users integrated into an ad-hoc network via communication technologies such as GSM or WiFi. Places are defined via coordinates or address/intersection. "Car" actually stands for any means of transportation, including family car, small car, or bicycle. The division of functionalities between the car KP and driver KP are yet to be defined. The broker produces possible matches between transportation service requesters (User1 KP, User2 KP) and providers (User3 KP, User4 KP). These matches are used then for direct negotiation between KPs concerning the transportation service.

III. Logistics Ontology

The logistics ontology consists of 3 parts: the driver's part, the passenger's part and the cargo item's part. The generic part of the ontology is shown in Figure 4.

A. Point

"Point" is an auxiliary class used for route definition in the ontology and has the properties shown in Figure 5. Usage of the class "point" can be illustrated via the following example. The driver ("user1") goes from point "A" (at 10:00) to "B" (at 11:00) without any passengers/cargo (Figure 6). In this case the driver will have two instances of the "point" class assigned (Figure 7).

If the driver goes through several points: starts from point "A" at 10:00, picks up a passenger at point "B: at 10:20, makes the passenger out at point "C" at 10:40, and finally arrives to "D" at 11:00 (Figure 8). In this case the driver will have four instances of the "point" class assigned (see Figure 9).

B. User

The class "user" describes all possible users of the system and has the properties shown in Figure 10. Property "Delay" defines the delay acceptable for the user when using the system. There can be several instances of the class "point" attached to an instance of the class "user". The subclasses of the class "user" inherit its properties.

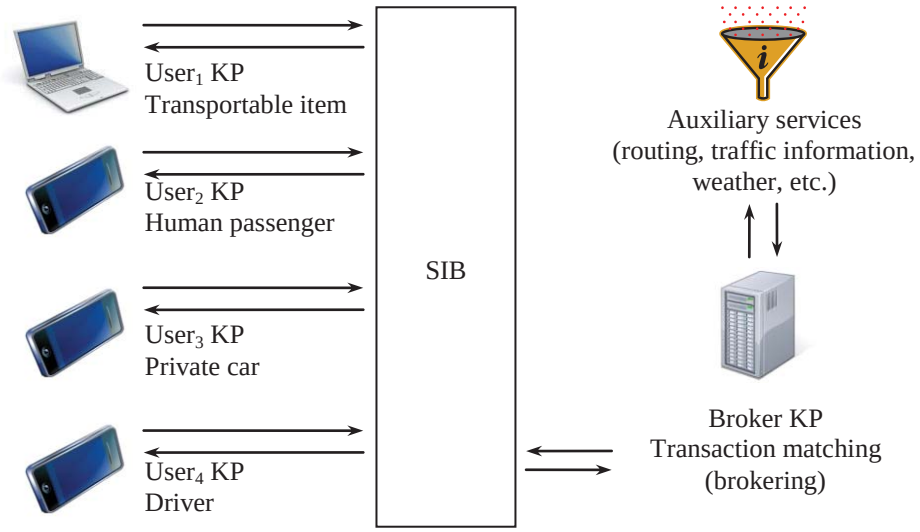

Figure 3. Architecture of the sustainable smart logistics prototype

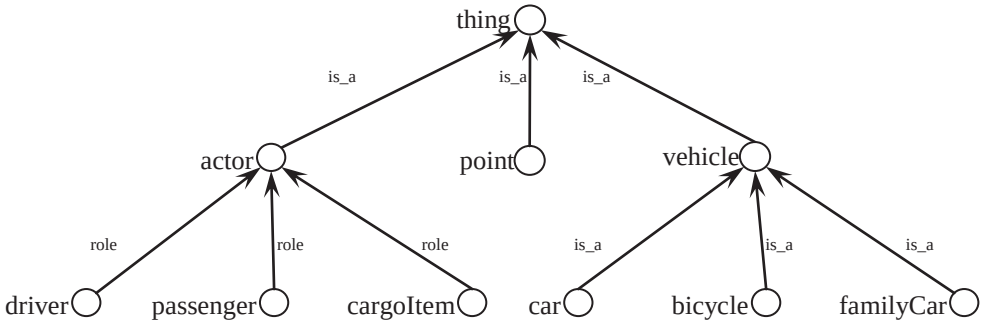

Figure 4. Generic part of the logistics ontology

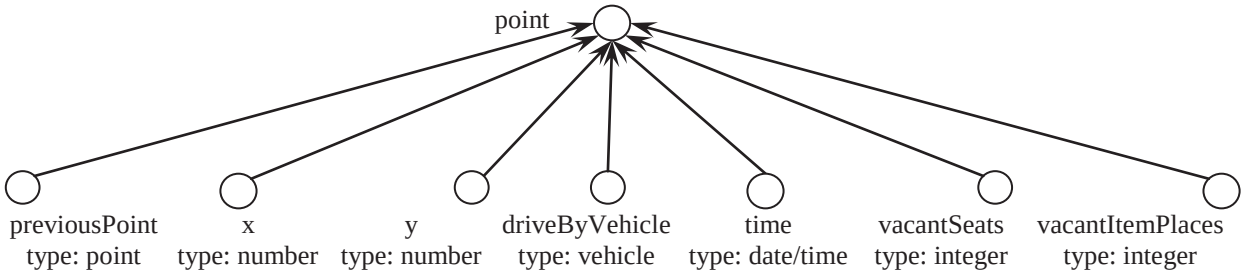

Figure 5. Properties of the class “point”

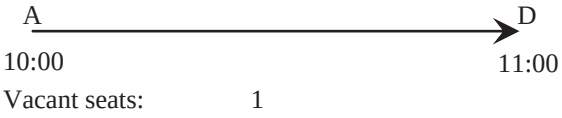

Figure 6. The driver goes from point “A” (“point1”) to point “B” (“point2”)

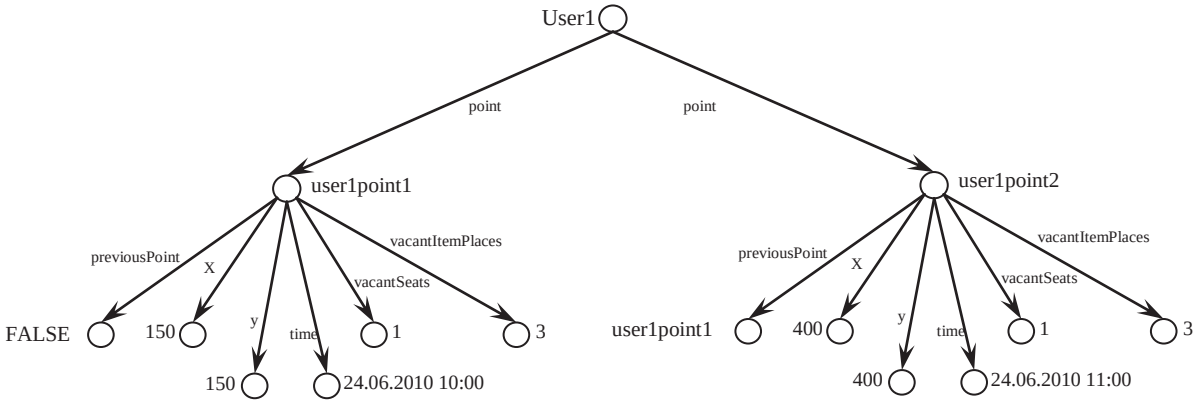

Figure 7. Example of the ontology for the driver going from point “A” (“point1”) to point “B” (“point2”)

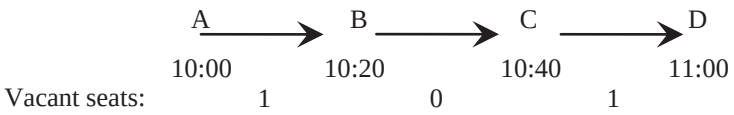

Figure 8. The driver goes through several points

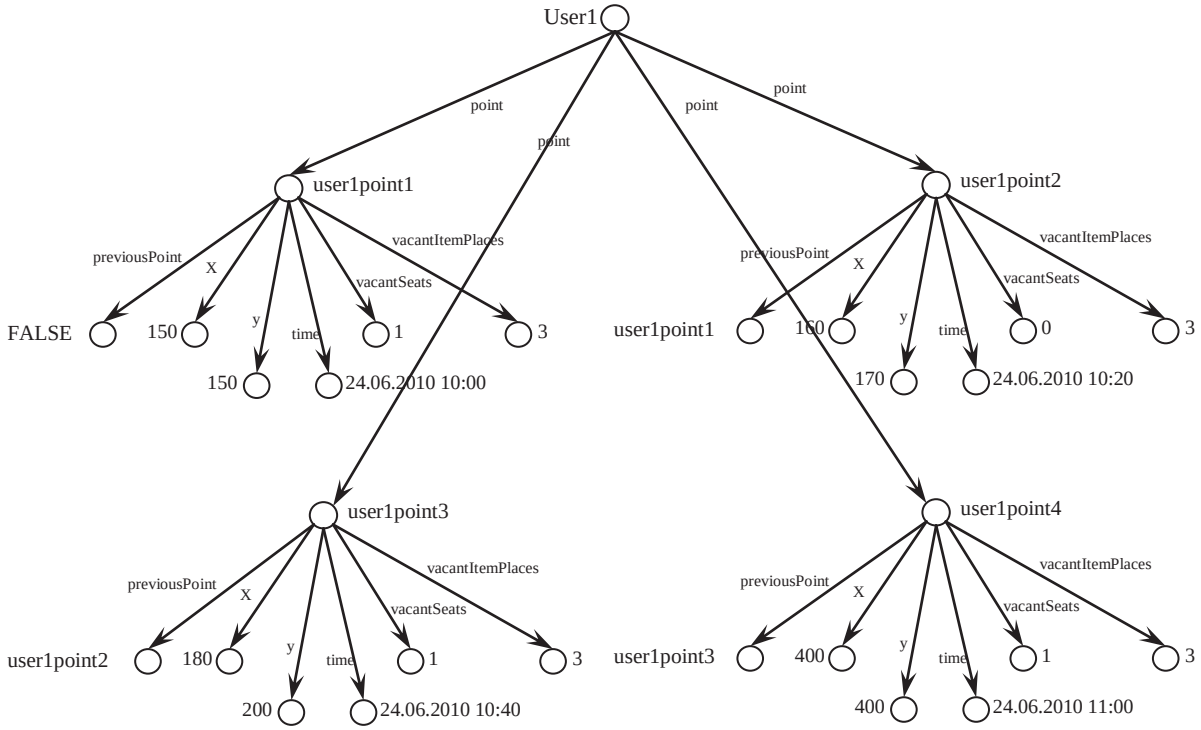

Figure 9. Example of the ontology for the driver going through several points

The subclass “driver” of the class “user” has the properties shown in Figure 11. Property “detour” defines the detour acceptable for the driver when using the system.

The subclass “passenger” of the class “user” has the property “detour” shown in Figure 12. it defines the detour acceptable for the driver when using the system.

The subclass “cargoItem” of the class “user” has the property “size” shown in Figure 13. “n” free item places are required to transport item of the size “n”.

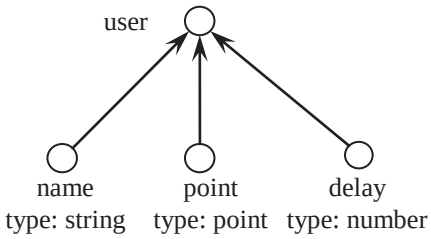

Figure 10. Properties of the class “user”

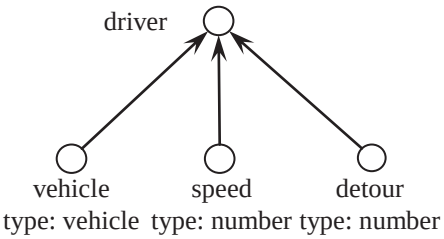

Figure 11. Properties of the class “driver”

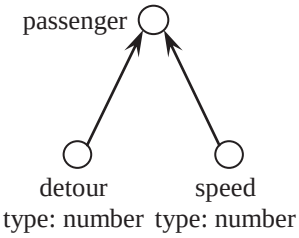

Figure 12. Property of the class “passenger”

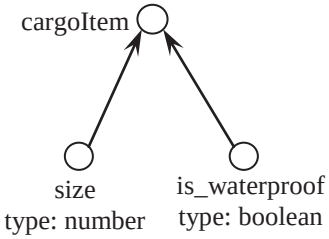

Figure 13. Property of the class “cargoitem”

IV. LOGIC IN COMPONENTS

A. Passenger and Item

These actors put a request to transport a passenger/item with certain characteristics from place A to place B between time points X and Y and receive corresponding offers. The scenario logic for the KP representing a cargo item or a passenger is the same. However, the "user" mentioned in the scenario is the passenger him/her-self for the "passenger KP" and a person sending the item for the "cargo item KP". The scenario logic is represented in Figure 14, its subscenarios are represented in Figure 15 and Figure 16. The waiting time assumes the time during which the passenger/item sender or driver (sec. B) can wait for the solutions. After this time they either make a choice or try to find different travel means.

B. Driver

This actor puts an Offer to transport persons and/or items and receives corresponding transport requests. The scenario logic is represented in Figure 17; its subscenarios are represented in Figure 18 and Figure 19.

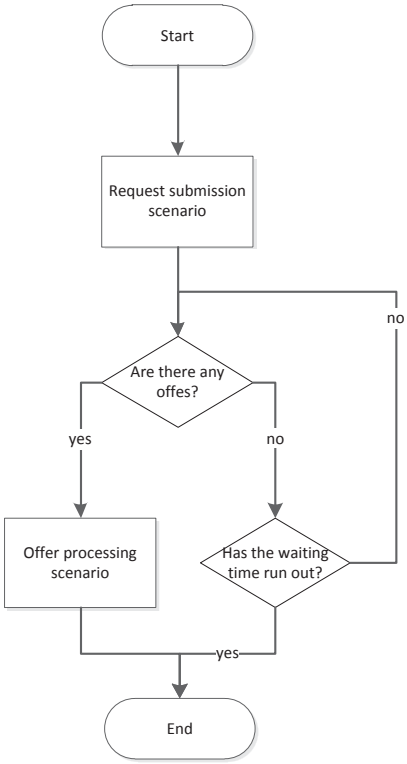

Figure 14. Scenario logic for passenger/item

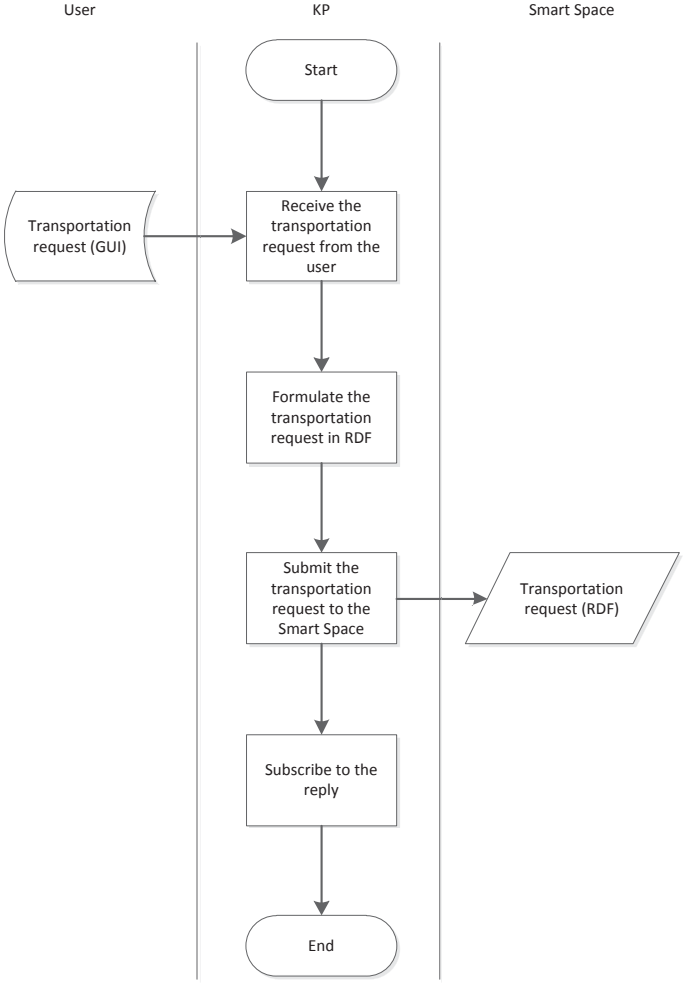

Figure 15. Request submission scenario logic for passenger/item

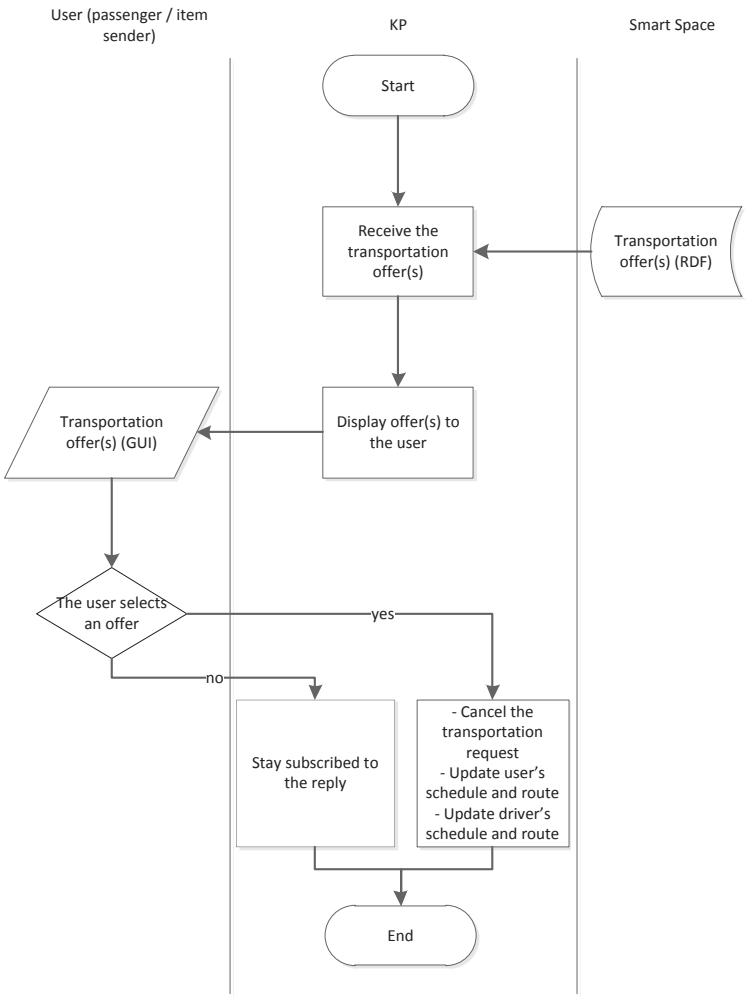

Figure 16. Offer processing scenario logic for passenger/item

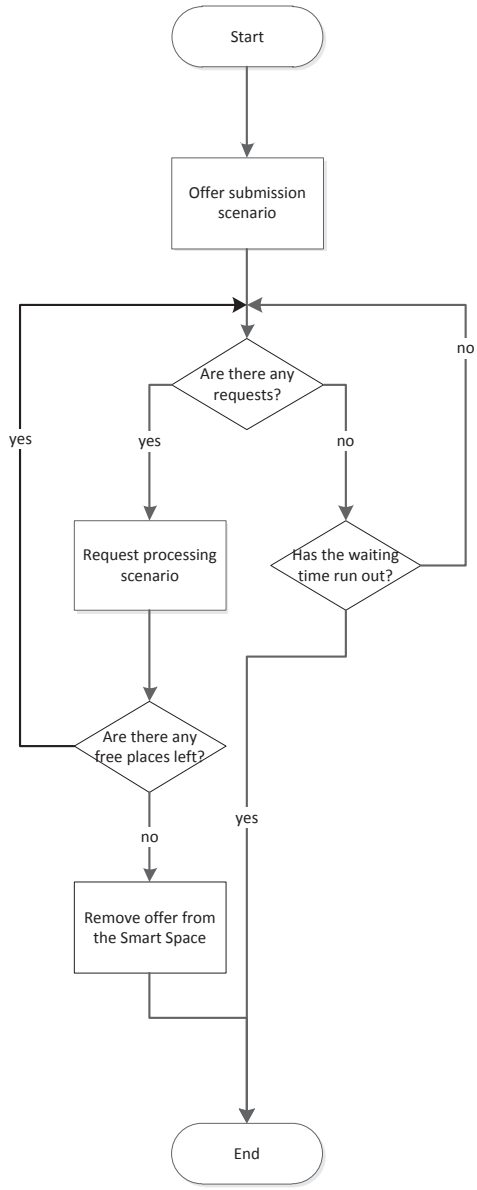

Figure 17. Scenario logic for driver

C. Broker

Broker process the transport requests and offers to find the matching ones. The execution logic for broker is represented in Figure 20.

The broker tries to find matching requests and offers. For each request in the smart space (i-cycle) it checks all available offers (j-cycle). However, in reality the amount of calculations can be significantly reduced since the complete search through all the offers and requests is needed only when the broker starts. After that it only needs to go through offers upon appearance of a new request and go through requests upon appearance of a new offer.

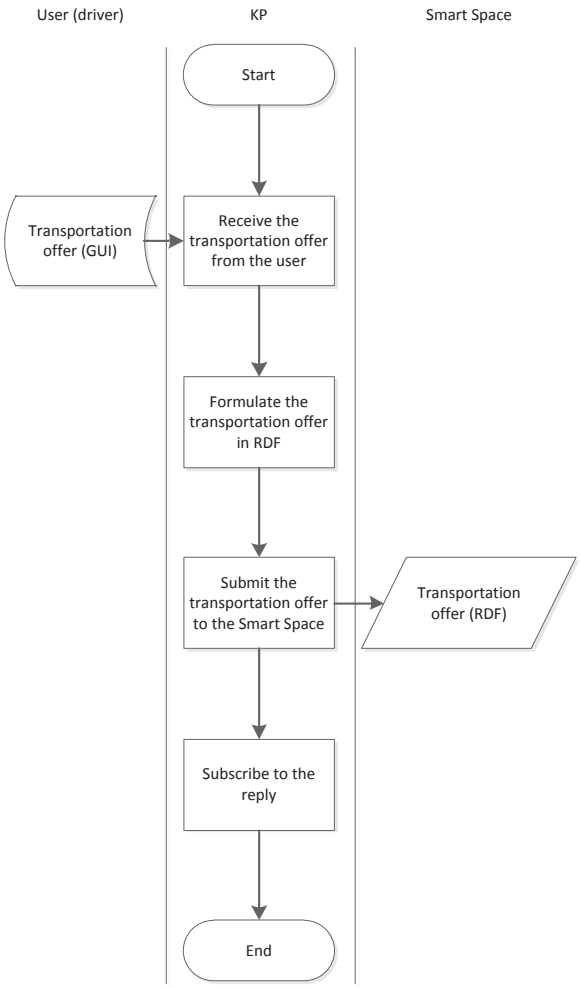

Figure 18. Offer submission scenario logic for driver

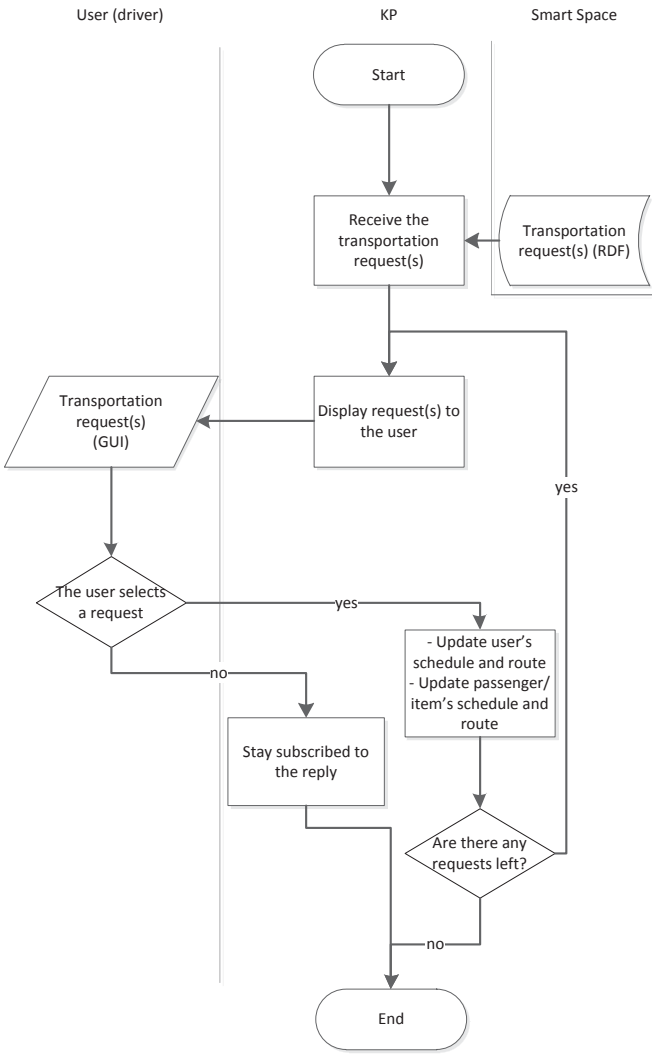

Figure 19. Request processing scenario logic for driver

First, the minimal distance between the passenger's start point and driver's route as well as the minimal distance between the passenger's end point and driver's route have to be found in order to estimate if the given driver can give a ride to the given passenger (whether they are not to far from one another): the sum of these distances should be smaller than the sum of the passenger's and driver's detours (boxes 3 and 4).

Then, it is checked if timewindows of the passenger [p.start_time, p.end_time+p.delay] and the driver [d.start_time, d.end_time+d.delay] intersect (box 5). If no, the joint trip is not possible.

Operations 3-5 are aimed to decrease the search space significantly.

After that the more precise estimation of the possibility of the joint travel and the joint travel's path and schedule is done (box 6). The following mathematical model describes the "Adjustment of the driver's and passenger's paths".

Input data:

- passenger's max detour (p.det);
- passenger's max delay (p.del);

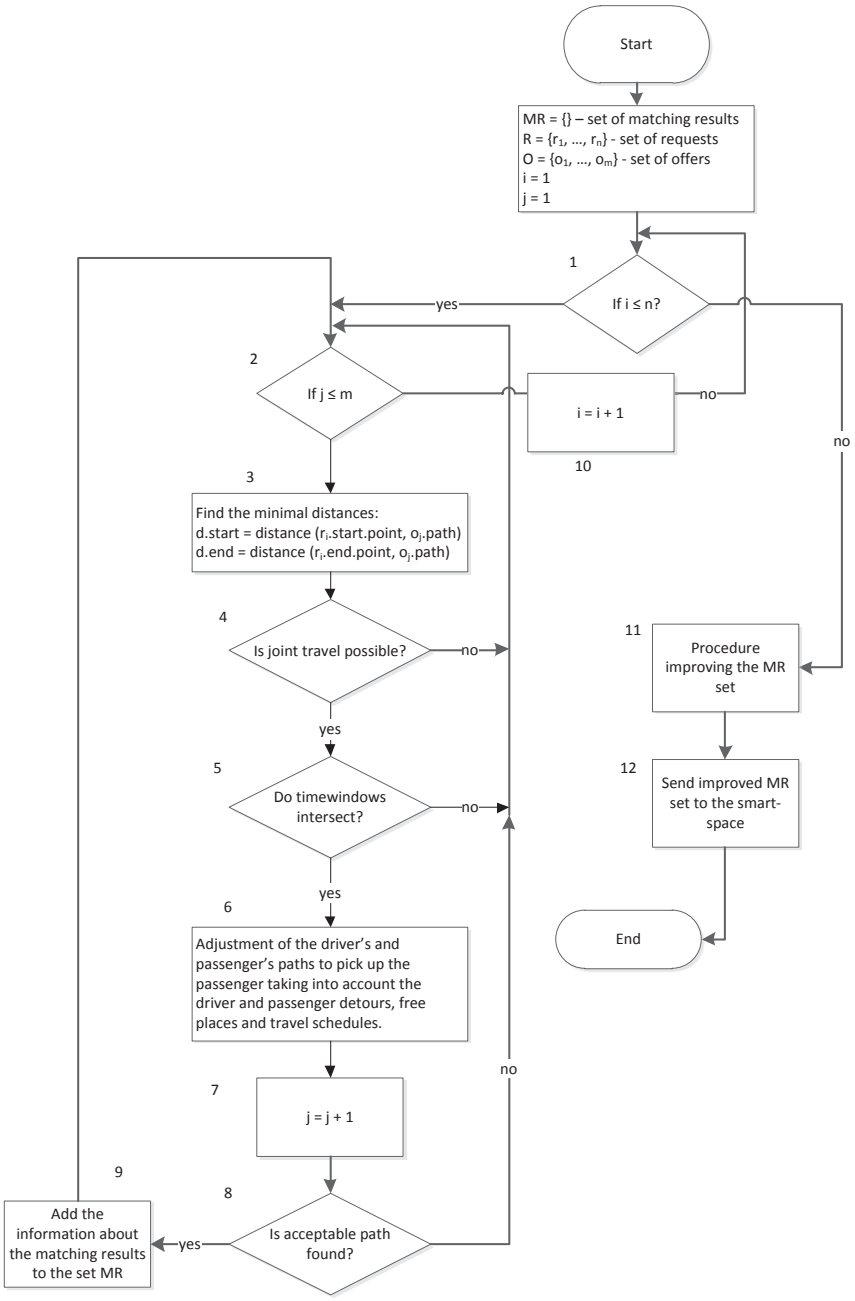

Figure 20. Execution logic for broker

- passenger's estimated speed (p.speed)
- driver's max detour (d.det);
- driver's max delay (d.del);
- driver's estimated speed (d.speed)
- passenger start and end locations: p.points = [$p.point_1$, $p.point_2$];
- driver's path, ordered array of points: d.point = [$d.point_1$, $d.point_2$, …, $d.point_n$];
- minimal distances (d_1, d_2) between the nearest points of driver and passenger calculated at the previous step ($p.point_1$ and $d.point_i$, $p.point_2$ and $d.point_j$);

Each point includes the following values: [x, y, time, vacant_seats];

ms.point = find_start_meeting_point(p.det, p.del, p.speed, d.det, d.del, d.speed, p.points, d.points)
me.point = find_end_meeting_point(p.det, p.del, p.speed, d.det, d.del, d.speed, p.points, d.points)

These functions search for the start and end meeting points (ms.point = [x, y, time]) of the passenger and driver, taking into account the following constraints:

$$distance(p.point, m.point) \leqslant p.det$$

$$distance(d.point, m.point) \leqslant p.det$$

$$\frac{distance(p.point, m.point)}{p.speed} \leqslant p.del$$

$$\frac{distance(d.point, m.point)}{d.speed} \leqslant d.del$$

These functions also check for vacant seats availability:

```
foreach d.point between (ms.point and me.point)
        if d.point->cacant_seats < 1 then
                return “The driver d can’t take the passenger p”
```

If an acceptable joint path is found the information about it is saved.

Let us consider the following case: passenger 1 can travel with drivers 1 and 2, passenger 2 can travel only with driver 1. In this case it would be better for the system for passenger 1 to travel with driver 2. For this purpose a procedure improving the set of matching requests and offers is proposed.

The solutions are presented to the clients (passengers, drivers, item senders) into a certain sequence in accordance with delays, detours. Besides, it is proposed to take into account the above mentioned factor. The procedure assigns weights to the solutions in accordance with the following logic. For passengers/items the drivers who can transport more passengers (have more matching requests) have a lower weight. For drivers the passengers/items which have more travel options have a lower weight. Since the clients will tend to select the topmost solutions the solutions with a higher weight will be presented followed by the solutions with a lower weight.

The screens represented in Figure 21 demonstrate the routes for the passenger as a pedestrian (left), for the driver (right), and the joint route when the driver gives a ride to the passenger (bottom).

V. Conclusion

The paper extends the proposed earlier approach to sustainable logistics based on the usage of Smart-M3 platform for supporting a ridesharing system. Presented ontologies and scenarios are aimed at being processed by handheld devices. The broker is aimed to be run on a server, however, it has to be responsive enough to be able to handle millions of queries daily. For this purpose the matching procedure within the broker has been split into several steps, where each step significantly reduces the search step at minimal computational expenses. The presented approach also has several heuristics which allow significant reduction of the approach complexity. These heuristics restrict the search space in computationally intensive parts of algorithms.

Acknowledgment

The work has been done under the joint project between Nokia and SPIIRAS. Some elements of the work have been carried out under grants of the Russian Foundation of the Basic Research

Figure 21. Prototype screenshots

(# 08-07-00264 and # 10-07-00368) and of the Presidium of the Russian Academy of Sciences (project # 213 of program # 14)

REFERENCES

[1] Global GHG Abatement Cost Curve v 2.0, 2009. URL: https://solutions.mckinsey.com/-ClimateDesk/default.aspx.

[2] UK Industry Task Force for Peak Oil and Energy Security, The oil crunch. Final report, 2010. URL: http://peakoiltaskforce.net/wp-content/uploads/2010/02/final-report-uk-itpoes_report_the-oil-crunch_feb20101.pdf.

[3] US Joint Forces Command, The Joint Operation Environment (JOE), 2010. URL: http://www.jfcom.mil/newslink/storyarchive/2010/JOE_2010_o.pdf.

[4] H. Paloheimo, H. Waris, S. Balandin, A. Smirnov, A. Kashevnik, N. Shilov, "Ad-Hoc Mobile Networks for Sustainable Smart Logistics," *Proceedings of the Congress on intelligent systems and information technologies "AIS-IT'10"*, Divnomorskoe, Russia, Sept. 2-9, Physmatlit, Vol. 4, pp. 38-44, 2010.

[5] Smart-M3 at Wikipedia, 2010. URL: http://en.wikipedia.org/wiki/Smart-M3.

[6] Smart-M3 at Sourceforge, 2010. URL: http://sourceforge.net/projects/smart-m3.