УДК 004.8

А.М. Кашевник, Н.Н. Тесля

АРХИТЕКТУРА ЛОГИСТИЧЕСКОЙ СИСТЕМЫ ПОИСКА ПОПУТЧИКОВ ДЛЯ ВОДИТЕЛЕЙ

Кашевник А.М., Тесля Н.Н. **Архитектура логистической системы поиска попутчиков для водителей.**

Аннотация. В статье рассматривается применение концепции интеллектуальных пространств для разработки логистической системы поиска попутчиков для водителей. Система базируется на онтологии информационного пространства, образованного мобильными устройствами участников дорожного движения и информационными каналами связи между этими устройствами. В статье представлено детальное описание архитектуры логистической системы поиска попутчиков для водителей. В описании архитектуры особое внимание уделено алгоритмам поиска совпадающих путей и точек встречи. Ввиду достаточно большой размерности задачи для ее решения предлагается использование эвристик, позволяющих снизить размерность задачи, что дает возможность существенно сократить время работы системы. Для демонстрации возможностей системы был разработан прототип, базирующийся на предложенной архитектуре, алгоритмах и онтологии системы.

Ключевые слова: онтология, интеллектуальные пространства, транспортная логистика, водители, пассажиры, карпул.

Kashevnik A.M, Teslya N.N. **Architecture of logistics system finding fellow-travellers for drivers.**

Abstract. The paper describes an architecture of the logistics system based on application of the smart space idea to finding fellow-travelers for drivers. The ontology formed by mobile devices of the system participants and interconnections between them are presented. The paper also describes algorithms for finding appropriate fellow-travelers for drivers as well as definition of the pick-up and drop-off points meeting requirements of both drivers and passengers. Due to the rather large dimension of the problem, the usage of heuristics significantly reducing the dimension of the task is proposed. To demonstrate the possibilities of the architecture and its underlying components, the software prototype of this system has been developed and described.

Keywords: ontology, smart space, transport logistic, drivers, passenger, carpool

1. Введение. Многие страны рано или поздно сталкиваются с транспортной проблемой, суть которой заключается в перегруженности существующей сети дорог и сети пассажироперевозок. К тому же, существующая сеть маршрутов общественного транспорта не всегда полностью покрывает населенный пункт и многим пассажирам приходится либо длительное время добираться до остановок, либо ощутимо долго ждать общественного транспорта. Основные способы решения проблемы — развитие транспортной инфраструктуры города, увеличение числа маршрутов общественного транспорта. Все эти решения обладают довольно существенными недостатками. Они требуют

огромного вложения финансов, увеличивают выбросы CO_2 в атмосферу, нанося вред окружающей среде, долго окупаются, повышается загруженность дорог, что может повлечь за собой частые заторы и аварии, возникают проблемы с поиском свободных парковочных мест. Однако, существует решение, лишенное данных недостатков — развитие системы карпула. Карпул (англ. *carpool*, так же известный как *ride-sharing*, *lift-sharing* и *covoiturage*) — это совместные поездки на автомобиле [1]. Преимущества данной системы очевидны: на дорогах уменьшается число транспортных средств, благодаря чему уменьшается число заторов и вероятность аварий, существенно снижается выброс парниковых газов и уменьшается количество припаркованных автомобилей на улицах города. Для участников карпула становится очевидным выигрыш в стоимости и удобстве поездки: используется только один автомобиль, следовательно, снижаются общие расходы на топливо, ремонт и обслуживание. К тому же, возникает возможность дополнительного общения людей [2].

В данной работе предлагается архитектура системы для поиска попутчиков, основанная на использовании интеллектуальных пространств [16]. Каждый пассажир и водитель имеют мобильное устройство, которое с соответствующим программным обеспечением является частью этого пространства.

2. Существующие системы карпула. Идеи карпула существуют еще с 1970 года, однако повсеместное использование и рост популярности стали возможны только с развитием сети Internet и мобильных средств связи, благодаря которым пассажиры и водители, ранее не знакомые, смогли быстро находить друг друга и договариваться о деталях поездки. В наши дни существуют следующие схемы карпула [1]:

- Поиск через форумы и сообщества. Такими ресурсами являются, например, «Давай со мной» [3], «Довезу!ру» [4], eRideShare.com [5], PickupPal [6], Zimride [7], RideshareOnline [8], rideshare.511.org [9], CarJungle [10]. На сайте размещается объявление о поездке, которое содержит в себе информацию о человеке, подающем это объявление и маршрут, с указанием начальной и конечной точек. Поиск попутчиков осуществляется заданием маршрута поездки и дополнительных условий, таких как плата за проезд, время поездки и др.

- Поиск через закрытые веб-сайты. Закрытость выражается в необходимости приглашения для участия. Например, сервис

Zimride [7], вместе с общедоступным интерфейсом имеет закрытый, который могут использовать различные организации для обеспечения подвоза своих сотрудников к месту работы.

- Использование специализированного программного обеспечения для поиска. На мобильные устройства загружается программа, которая позволяет отредактировать профиль пользователя, указать маршрут и найти попутчиков. Примеры: PickupPal [6], Avego [11].
- Поиск через различных агентов (таксомоторные компании).
- Случайный подбор попутчиков на точках встречи, которыми могут быть, например, остановки общественного транспорта.

Уже существующее программное обеспечение для мобильных устройств поддерживает клиент-серверную архитектуру, реализация которой предусматривает наличие централизованного сервера и клиентов, осуществляющих запросы на обработку данных к серверу.

Данная работа, предусматривает децентрализацию вычислительного комплекса, которая осуществляется путем использования технологий Semantic Web и, в частности, интеллектуальных пространств (*Smart Spaces*) с помощью платформы Smart-M3, разрабатываемой Nokia Research Center [12, 15].

3. Платформа Smart-M3. Платформа Smart-M3 объединяет в себе идеи распределенных сетевых систем и Semantic Web. Ее ключевыми идеями является независимость от конкретных производителей, оборудования, области применения и возможность обмена информацией между различными программными модулями посредством простого и общедоступного информационного брокера. Благодаря использованию Semantic Web в качестве основы, обмен информацией между участниками пространств может осуществляться на основе протокола HTTP и с использованием унифицированных идентификаторов ресурсов (Uniform Resource Identifier — URI) [13].

Общая структура платформы представлена на рис. 1. Ядро системы подразделяется на 2 элемента: СИБ (семантический информационный брокер, *Semantic Information Broker — SIB*) и физическое хранилище данных. СИБ предоставляет доступ информационным агентам к информационному пространству, обеспечивая их функциями обработ-

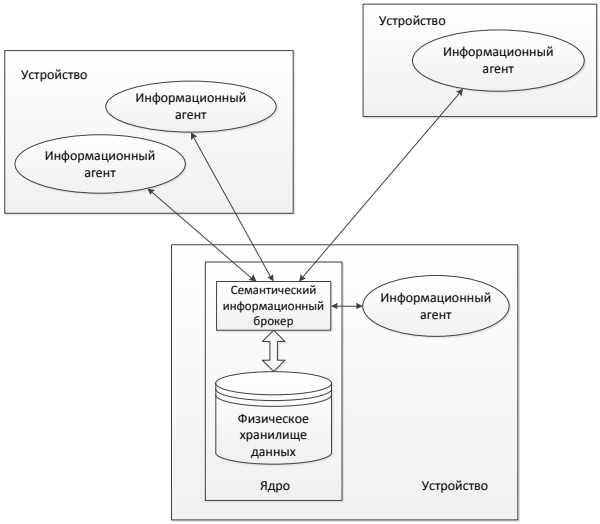

Рис. 1. Структура платформы Smart-M3.

ки информации, такие как вставка, извлечение, редактирование, удаление и подписки на изменение информации в интеллектуальном пространстве. В хранилище данных вся информация сохраняется как граф, удовлетворяющий требованиям стандарта RDF (*Resource Description Framework*) [14], описывающего семантические сети, в которых узлы и дуги имеют унифицированные идентификаторы ресурсов. Каждое утверждение, в соответствии с этим стандартом, описывается тройкой «субъект – предикат – объект», например, «Иван – имеет – автомобиль» и, по своей сути, является простым предложением. Информационные агенты — это программные модули, с помощью которых осуществляется взаимодействие с семантическим информационным брокером через Smart Space Access Protocol (SSAP — протокол доступа к интеллектуальному пространству) [12,15].

4. Архитектура системы. Архитектура системы представлена на Рис. 2. Информационными агентами являются программы, установленные на портативных устройствах, принадлежащих пользователям, среди которых можно выделить пассажиров и водителей. Информационный брокер — это программный модуль, осуществляющий поиск

Рис. 2. Архитектура системы.

совпадающих частей пути на основе информации, передаваемой пассажирами и водителями в интеллектуальное пространство и информации, запрашиваемой у географической информационной системы (ГИС). Информацией об участнике является:

- информация о пользователе (имя, тип пользователя – водитель/пассажир и др.)
- координаты точки отправления,
- координаты точки прибытия,
- время отправления,
- максимальная задержка,
- максимальное отклонение от минимального расстояния.

Если пользователь является водителем, то для него дополнительно учитывается тип транспортного средства: количество мест для пассажиров и для багажа.

5. Онтология системы. В ходе анализа и формализации компонентов системы была спроектирована онтология логистической системы, представленная на Рис. 3.

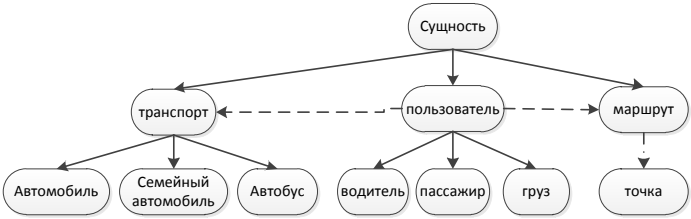

Рис. 3. Онтология логистической системы на макроуровне.

В системе присутствует три типа сущностей: транспортные средства, пользователи и точки. Транспортными средствами могут быть обычные автомобили, в которых число мест для пассажиров не больше четырех, семейные автомобили с числом мест от пяти до восьми и автобусы, имеющие девять и более мест для пассажиров.

«Точка» является частью маршрута и используется для определения путей пользователей. Ее онтология представлена на Рис. 4.

Пользователями в системе выступают водители, пассажиры и грузы. При этом все они имеют связь с транспортами и точками. Так, например, водитель является собственником транспорта и ему соответствуют несколько точек, определяющих положение дома, работы, мест остановок и др. Пассажир может отдавать предпочтение определенным видам транспортного средства и ему также соответствуют точки на карте, характеризующие его положение и пункт назначения. Груз имеет определенные размеры, и не каждый тип транспорта может подойти для его перевозки.

5.1. Онтология путей пользователей. Для описания пути используется множество точек, являющихся экземплярами сущности «Точка». Этот набор представляет собой упорядоченный список опорных точек, полученных с помощью алгоритмов нахождения кратчайшего пути на графе (например, алгоритмом Дейкстры или Флойда-Уоршелла), «натянутом» на топографический план местности. Вершины графа при этом являются опорными точками, а дуги повторяют положение и направление основных дорог. Фрейм «точка» представлен на рис. 4.

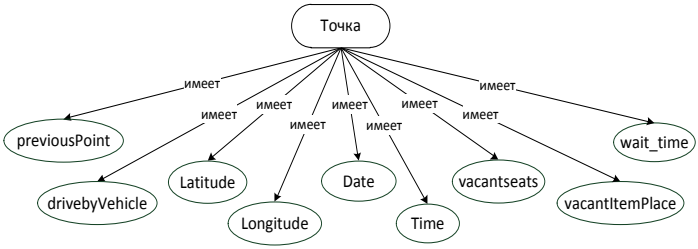

Рис. 4. Фрейм «точка пути».

В составленной онтологии фрейм «Точка» содержит в себе следующие атрибуты:

- previousPoint. Содержит предыдущую точку пути. Имеет значение FALSE, если точка является начальной;
- Latitude. Широта точки пути;
- Longitude. Долгота точки пути;
- driveByVehicle. В случае, если рассматриваемая точка является точкой пути пассажира, указывает на водителя, который в данный момент подвозит этого пассажира. Если пассажир идет пешком, имеет значение FALSE;
- vacantseats. число свободных мест для пассажиров в транспортном средстве в данный момент;
- vacantItemPlace. число свободных мест для груза в транспортном средстве в данный момент;
- Date. Расчетная дата нахождения пользователя в данной точке;
- Time. Расчетное время нахождения пользователя в данной точке;
- Wait_time. Время, в течение которого пользователь должен находиться в данной точке.

5.2. Онтология пользователя. Структура фрейма «пользователь» представлена на рис. 5.

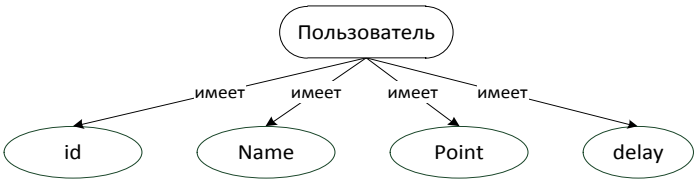

Рис. 5. Фрейм «пользователь».

Основными атрибутами для всех пользователей являются:

- ID. Уникальный идентификатор пользователя;
- Name. Имя пользователя;
- Point. Точки, соответствующие пользователю (как минимум 2: начальная и конечная)
- Delay. Время, в течение которого данный пользователь согласен ждать других пользователей.

Пассажиры, водители и грузы наследуются от сущности «пользователь». Сущность «водитель» наследует все атрибуты «пользователя», при этом расширяя его еще двумя атрибутами:

- Vehicle. Тип транспортного средства,
- Detour. Максимально возможное отклонение от кратчайшего маршрута.

Сущность «пассажир», так же, как и «водитель», наследует все атрибуты «пользователя» и добавляет к ним атрибут Detour, аналогичный подобному в сущности «водитель».

Сущность «груз» помимо атрибутов сущности «пользователь» имеет атрибут size, определяющий размеры груза.

5.3. Представление онтологии в интеллектуальном пространстве. Онтология, представленная выше, в стандарте RDF будет записана следующим образом [17]:

Водитель:

(‘user1’, ’name’, ‘Name Surname’)

(‘user1’, ‘is_a’, ‘Driver’)

(‘user1’, ‘vehicle’, ‘vehicle_type’)

(‘user1’, ‘detour’, ‘meters’)

(‘user1’, ‘delay’, ‘minutes’)

Описание точек для водителя:

(‘user1’,	‘point’,	‘user1point1’)	– начальная точка
(‘user1point1’,	‘previouspoint’,	’FALSE’)	
(‘user1point1’,	‘x’,	’Latitude’)	
(‘user1point1’,	‘y’,	’Longitude’)	
(‘user1point1’,	‘date’,	’dd.mm.yyyy’)	
(‘user1point1’,	‘time’,	’hh.mm’)	
(‘user1point1’,	‘vacantseats’,	’number’)	
(‘user1point1’,	‘vacantitemplaces’,	’number’)	
(‘user1’,	‘point’,	‘user1point2’)	– конечная точка
(‘user1point2’,	‘previouspoint’,	’user1point1’)	
(‘user1point2’,	‘x’,	’Latitude’)	
(‘user1point2’,	‘y’,	’Longitude’)	
(‘user1point2’,	‘date’,	’dd.mm.yyyy’)	
(‘user1point2’,	‘time’,	’hh.mm’)	
(‘user1point2’,	‘vacantseats’,	’number’)	
(‘user1point2’,	‘vacantitemplaces’,	’number’)	

Пассажир:

(‘user2’, ’name’, ‘Name Surname’)

(‘user2’, ‘is_a’, ‘Passenger’)

(‘user2’, ‘detour’, ‘meters’)

(‘user2’, ‘delay’, ‘minutes’)

Описание точек для пассажира:

(‘user2’, ‘point’, ‘user2point1’)

(‘user2point1’, ‘previouspoint’, ’FALSE’)

(‘user2point1’, ‘x’, ’Latitude’)

(‘user2point1’, ‘y’, ’Longitude’)

(‘user2point1’, ‘date’, ’dd.mm.yyyy’)

(‘user2point1’, ‘time’, ’hh.mm’)

(‘user2point1’, ‘drivebyvehicle’, ’driver_ID or FALSE’)

(‘user2’, ‘point’, ‘user2point2’)

(‘user2point2’, ‘previouspoint’, ’user1point1’)

(‘user2point2’, ‘x’, ’Latitude’)

(‘user2point2’, ‘y’, ’Longitude’)

(‘user2point2’, ‘date’, ’dd.mm.yyyy’)

(‘user2point2’, ‘time’, ’hh.mm’)

(‘user2point1’, ‘drivebyvehicle’, ’driver_ID or FALSE’)

Багаж:

(‘user3’, ’name’, ‘Name Surname’)

(‘user3’, ‘is_a’, ‘item’)

(‘user3’, ‘size’, ‘size’)

(‘user3’, ‘delay’, ‘minutes’)

Описание точек для багажа:

(‘user3’,	‘point’,	‘user3point1’)
(‘user3point1’,	‘previouspoint’,	’FALSE’)
(‘user3point1’,	‘x’,	’Latitude’)
(‘user3point1’,	‘y’,	’Longitude’)
(‘user3point1’,	‘date’,	’dd.mm.yyyy’)
(‘user3point1’,	‘time’,	’hh.mm’)
(‘user3point1’,	‘drivebyvehicle’,	’driver_ID or FALSE’)
(‘user3’,	‘point’,	‘user3point2’)
(‘user3point2’,	‘previouspoint’,	’user1point1’)
(‘user3point2’,	‘x’,	’Latitude’)
(‘user3point2’,	‘y’,	’Longitude’)
(‘user3point2’,	‘date’,	’dd.mm.yyyy’)
(‘user3point2’,	‘time’,	’hh.mm’)
(‘user3point1’,	‘drivebyvehicle’,	’driver_ID or FALSE’)

6. Информационный брокер логистической системы поиска попутчиков для водителей. На рис. 6 приведена схема взаимодействия информационного брокера и интеллектуального пространства.

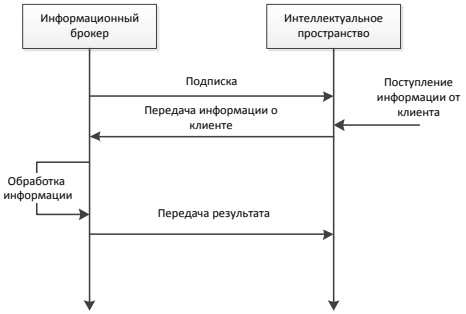

Рис. 6. Взаимодействие информационного брокера и интеллектуального пространства.

При запуске информационный брокер осуществляет подписку на информацию, поступающую в интеллектуальное пространство. Это означает, что при появлении новой информации, участник, совершивший подписку, будет оповещен и автоматически получит доступ к ней. После обработки в информационном брокере, результат помещается в интеллектуальное пространство, откуда, впоследствии, попадет к участникам, запросившим данное вычисление. Подобные действия повторяются до тех пор, пока подписка не будет закрыта, либо работа брокера не будет остановлена.

6.1. Функционирование информационного брокера. Общее функционирование информационного брокера можно разделить на следующие этапы: старт и инициализация, обработка запросов, завершение работы. Обобщенная блок-схема представлена на рис. 7.

Рис. 7. Схема работы информационного брокера.

На этапе старта и инициализации производится кэширование информации о дорогах из ГИС, с целью ускорения их последующей обработки. Далее осуществляется подключение к интеллектуальному

пространству и подписка на поступление новой информации от пользователей. Подписка осуществляется следующим образом:

```
result_rdf = rs1.subscribe_rdf([(None, "show_path",
None), 'literal')], RdfMsgHandler())
```

`None` в данном случае указывает на то, что не важно, какая информация содержится в поле. Таким образом, при появлении в интеллектуальном пространстве данных, удовлетворяющих маске `(None, "show_path", None)`, информационный брокер будет автоматически оповещен и приступит к следующему этапу работы — обработке запроса.

На рис. 8 представлена блок-схема алгоритма, который обрабатывает информацию, поступающую из интеллектуального пространства при осуществлении подписки.

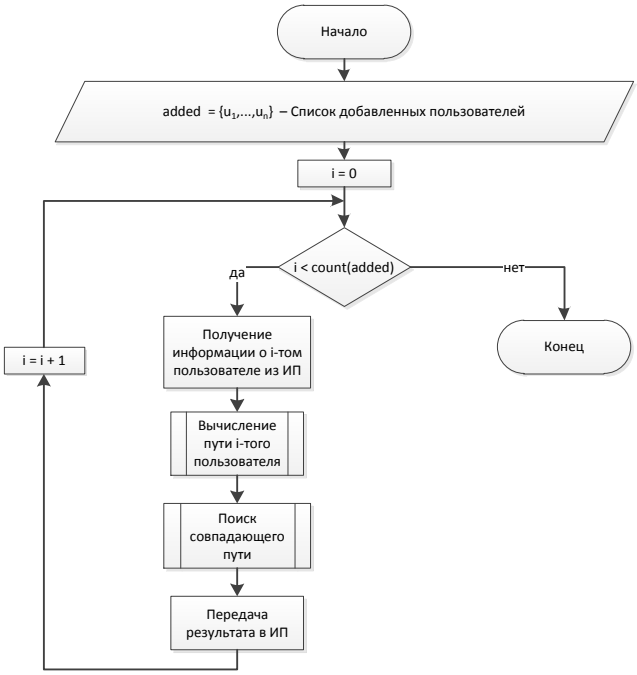

Рис. 8. Блок-схема алгоритма, обрабатывающего информацию из ИП.

Его работа заключается в следующем: из интеллектуального пространства извлекается информация о добавленном пользователе, затем вычисляются пути и атрибуты пути пользователя, предполагая, что у него нет попутчиков. Результаты всех вычислений пересылаются в информационное пространство и в дальнейшем используются для вычисления параметров совпадающего пути.

6.2. Получение информации о пользователе. Информация о пользователе передается из интеллектуального пространства путем формирования и отправки запроса информационным брокером и обработки им ответа. Блок-схема процедуры получения информации о пользователе представлена на рис. 9. Формирование и обработка запроса осуществляются с помощью метода `CreateQueryTransaction()`, входящего в класс `Node` модуля smart-m3, создается объект, который содержит функцию `rdf_query()`, позволяющую сформировать и передать запрос в виде RDF-тройки и возвращающая результат запроса. Например, если необходимо получить список всех водителей, присутствующих в интеллектуальном пространстве, то формируется запрос вида:

```
result = query.rdf_query((None,"is_a", "Driver"),
"literal")
```

Результатом запроса будет список всех RDF-троек из интеллектуального пространства, удовлетворяющих условиям, перечисленным в запросе.

Таким образом, сначала запрашивается информация о статусе пользователя в системе, его времени ожидания и дополнительном пути, который он согласен пройти или проехать. Если пользователь является водителем, то также запрашивается информация о типе транспортного средства и количество пассажиров и багажа, которое он может перевезти. Следующим шагом извлекается информация об опорных точках пути. С учетом построенной онтологии первоначально требуется получить информацию о конечной точке пути. Все остальные точки получаются «проходом» пути в обратном направлении, то есть следующей точкой, информация о которой будет извлечена из интеллектуального пространства, будет являться точка, имя которой записано в атрибуте previousPoint текущей точки. Процесс извлечения останавливается, когда значение атрибута previousPoint будет «FALSE», обозначающее, что предыдущие точки у текущей отсутствуют и она является начальной. Перед занесением в соответствую-

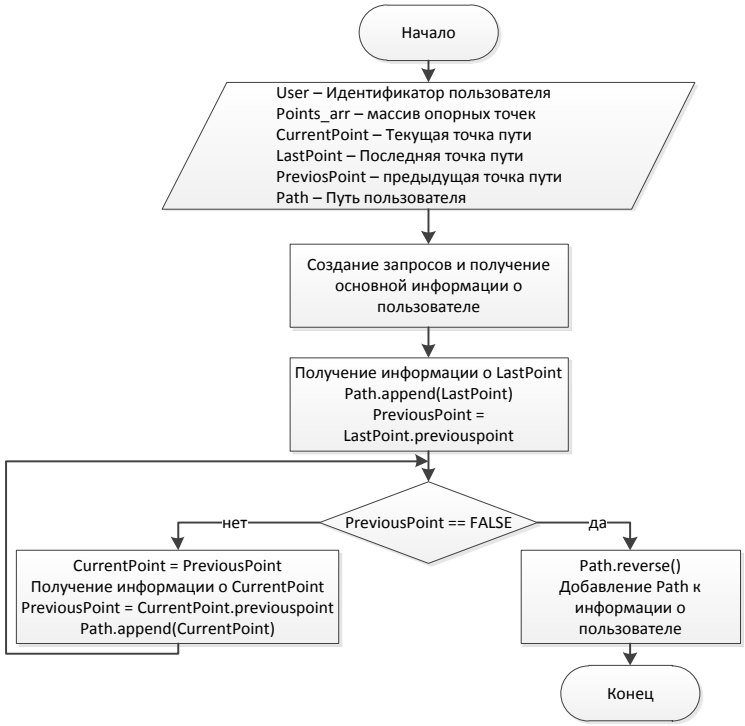

Рис. 9. Получение информации о пользователе.

щий атрибут пользователя список реверсируется для удобства анализа и обработки.

6.3. Получение информации о пути. Задача поиска кратчайшего пути решается средствами используемой ГИС. Однако, учитывая большую размерность задачи и необходимость довольно часто вычислять кратчайшие расстояния между точками пути, была проведена предварительная индексация минимальных путей между любыми двумя точками. Алгоритм вычисления пути по нескольким опорным точкам представлен на рис. 10.

Для построения пути по списку опорных точек на каждой итерации выбирается 2 соседних элемента списка. Используя их, производится поиск элемента индекса, содержащего в себе точки пути между

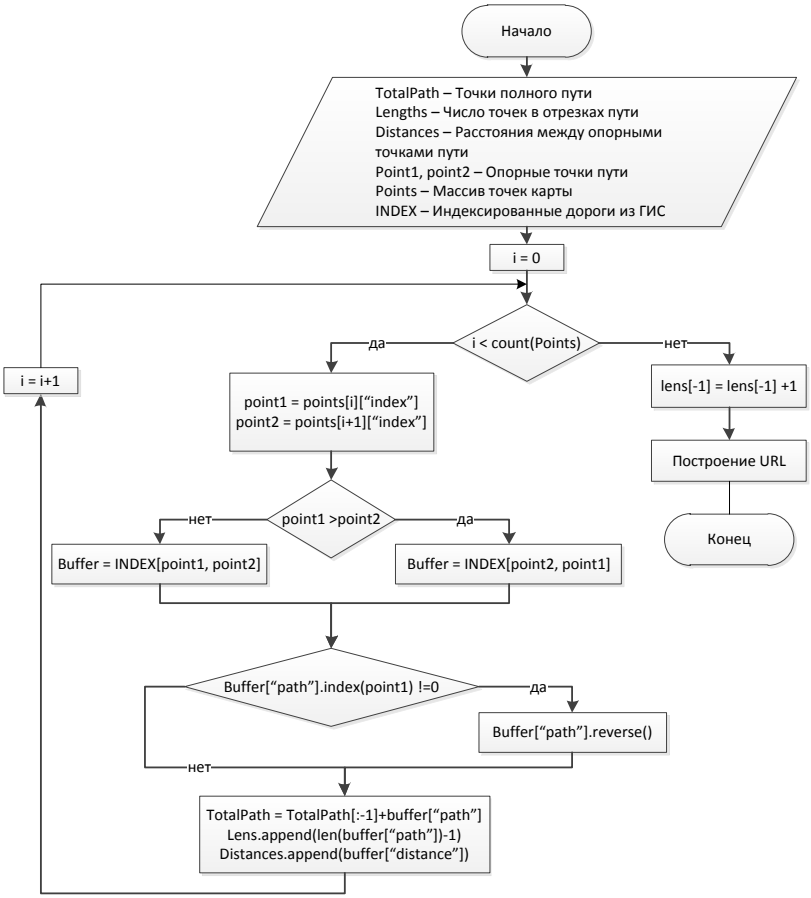

Рис. 10. Построение пути по опорным точкам.

двумя опорными, а также расстояние между ними. При этом обязательно необходимо учитывать свойство нижнетреугольности матрицы индекса, возникающее вследствие инвариантности пути относительно начальной и конечной точек. Для этого введена проверка на то, какая из них имеет больший индекс. Но при этом возможна ситуация, в которой из-за инвариантности путей последовательность точек получится обратной. Для обработки такой ситуации предусмотрена проверка на начальную точку пути. Если индекс указанной начальной точки в

полученном пути не равен «0», то путь реверсируется, иначе не производится никаких изменений. Общий путь (TotalPath) получается объединением всех отрезков пути, длины отрезков пути складываются для получения общей длины пути. Последним действием является построение URL адреса, с помощью которого путь будет отображен на карте. Этот адрес содержит в себе последовательность идентификаторов точек в том порядке, в котором они следуют в пути. Например, путь может иметь следующий вид:

```
http://path_to_the_map/index.php?id[]=1&id[]=2&id[]=3
&t[0]=1,
```

где `id[]` – идентификатор точки, `t[]={0..3}` — метка, указывающая статус пользователя в системе. При этом 0 обозначает пешехода, 1 — автомобиль, 2 — автобус, 3 — совпадающий путь. Считается, что метка действительна от точки указанной в квадратных скобках либо до следующей метки, либо, если меток больше не встречается, до конца пути.

6.4. Поиск совпадающих путей и точек встречи. Для поиска совпадающих путей составляются списки водителей и пассажиров, при этом для каждой пары осуществляется поиск совпадений. Из полученных пар выбираются пары, наиболее удовлетворяющие критериям, заданным участниками.

На рис. 11 рассмотрен общий принцип поиска совпадающего пути.

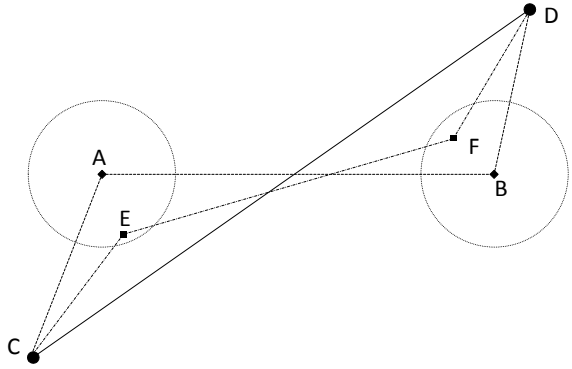

Рис. 11. Общий принцип поиска совпадающего пути.

Пусть A, B — начальная и конечная точки пути пешехода, C, D — начальная и конечная точки пути водителя. Сплошной линией обозначен кратчайший путь водителя, полученный с помощью ГИС. Как видно из рисунка, водитель и пешеход осуществляют движение практически в одном направлении и при некоторой корректировке пути водителя, он может подвезти пешехода, что обозначено на рисунке пунктирной линией (путь CABD). Рассмотренный случай является самым простым, так как точки встречи в нем совпадают с точками начала и конца пути пешехода. Более сложными вариантами является поиск точек встречи, удовлетворяющих и водителя и пассажира, но не обязательно являющихся частью их кратчайшего пути. На рисунке один из возможных вариантов обозначен штрих-пунктирной линией, а точки встречи обозначены буквами E, F (путь CEFD). Выбор этих точек должен удовлетворять следующим основным условиям:

1. Расстояние от начальной точки пассажира до точки встречи не должно превышать максимального пути, который согласен пройти пассажир. На рис. 11 эта область обозначена пунктирной окружностью.

2. Отклонение нового пути водителя от кратчайшего не должно превышать максимально допустимого отклонения, задаваемого водителем.

Общая задача поиска совпадающих путей имеет достаточно большую размерность, поэтому необходимо применять эвристики для уменьшения размерности задачи. На рис. 12 представлен алгоритм выбора точек начала и конца совпадающего пути.

Работа алгоритма начинается с определения возможности подвоза пассажира выбранным водителем. Для этого необходимо выполнение следующих условий:

$$(pp_1^x - dp_i^x)^2 + \left(pp_1^y - dp_i^y\right)^2 \le (PDetour + DDetour)^2 \qquad (1)$$

$$(pp_2^x - dp_i^x)^2 + (pp_2^y - dp_i^y)^2 \le (PDetour + DDetour)^2, \qquad (2)$$

где pp_1, pp_2 — начальная и конечная точки пути пассажира, dp_i — точка пути водителя, PDetour, DDetour — отклонение от минимального пути пассажира и водителя. Если путь водителя не удовлетворяет условиям (1), (2), то считается, что водитель не может подвезти пассажира и действие алгоритма прекращается, иначе составляются

списки точек пути водителя, соответствующих условиям и осуществляется переход к следующему шагу, на котором производится выбор всех возможных точек встречи.

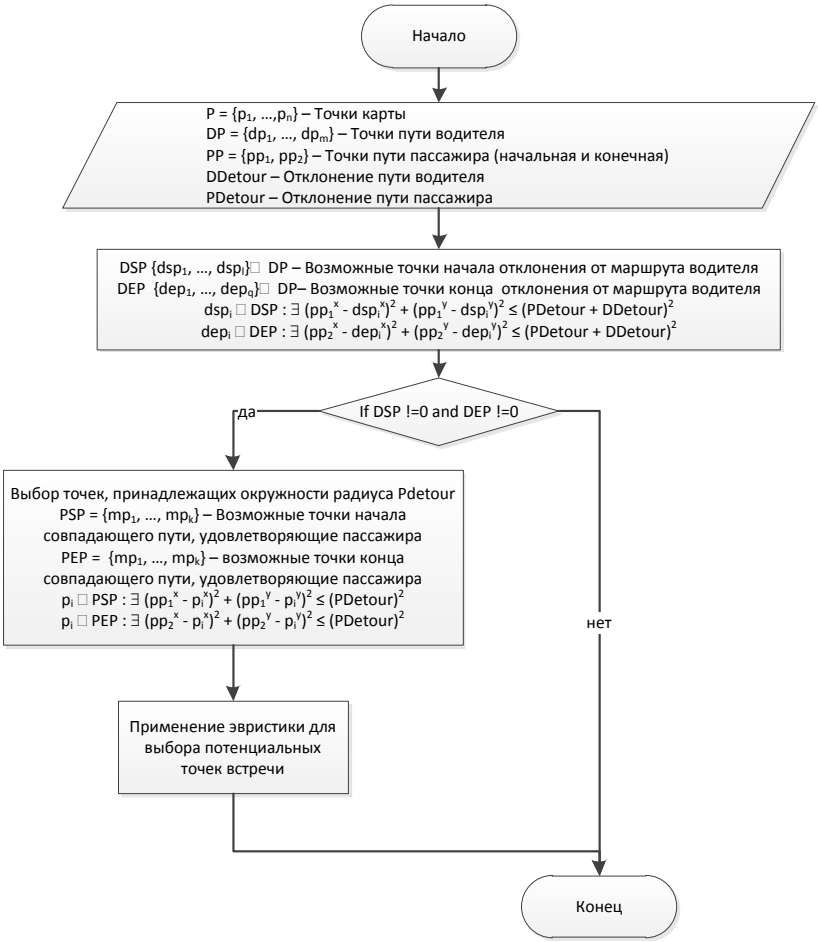

Рис. 12. Алгоритм выбора точек начала и конца совпадающего пути.

Следующим шагом осуществляется выбор всех точек, до которых может дойти пешеход. Эти точки принадлежат окружности радиуса `PDetour` и выбираются по следующему правилу:

$$(pp_A^x - p_i^x)^2 + \left(pp_A^y - p_i^y\right)^2 \leq PDetour^2, \quad (3)$$

где pp_A — точка, принадлежащая пути пассажира, p_i — точка на карте. Если p_i удовлетворяет условию (3), то она помечается как возможная точка встречи. На рис. 13 точки, удовлетворяющие условию (3) обозначены буквами L, M, N, K.

Выбор из возможных точек встречи пары, наиболее удовлетворяющей критериям, заданным пассажиром предусматривает перебор всех пар точек с вычислением всех требуемых для выбора параметров. Данная операция является крайне затратной и требует сокращения множества возможных точек встречи. Для решения поставленной задачи были разработаны эвристики, позволяющие на основе имеющихся данных сократить множества.

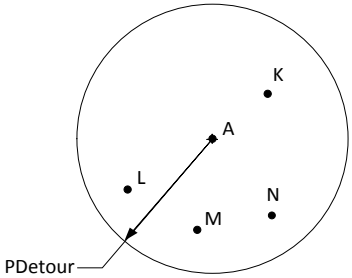

Рис. 13. Выбор точек встречи для пассажира.

Первая эвристика предполагает выбор точек из сектора окружности, со стороны которой будет ехать водитель. На рис. 14 представлена эвристическая модель для ситуации, в которой у водителя нашлась только одна точка (точка C), удовлетворяющая условию (1) или (2), в зависимости от рассматриваемой точки пути пассажира.

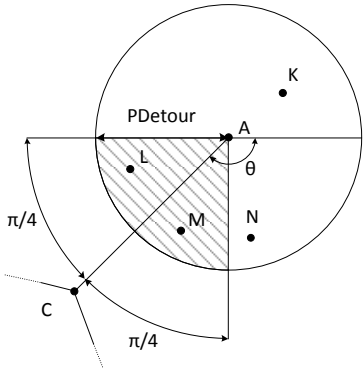

Рис. 14. Схема для одной точки пути водителя.

Чтобы определить потенциальные точки встречи вычисляется угол

$$\theta = \text{arctg}\left(\frac{C^y - A^y}{C^x - A^x}\right) \quad (4)$$

и выбираются точки, которые попадают в область $\left[\theta - \frac{\pi}{4}, \theta + \frac{\pi}{4}\right]$. На рис. 14 это точки L и M. Точка A, являющаяся начальной точкой для пассажира, всегда будет включена в множество возможных точек встречи. Если нашлись две и более точки, удовлетворяющих условию (1), то область выбора точек расширяется. На рис. 15 представлена схема применения эвристики, полученной из схемы, изображенной на рис. 14, путем добавления второй точки, удовлетворяющей условию (1) или (2). При этом видно, что в расширенную область вместе с точками L и M теперь входит и точка N.

Применение подобной эвристики позволяет рассматривать только точки, находящиеся со стороны движения водителя. Однако, она обладает существенными недостатками:

– Выбор точек, расположенных дальше, чем максимальное отклонение водителя от кратчайшего пути.

– Область выбора, ограниченная определенным углом. Из-за некорректного выбора угла возможна потеря потенциальных точек встречи.

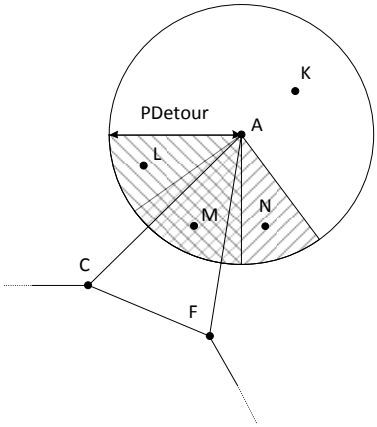

Рис. 15. Схема для двух точек пути водителя.

Вторая эвристика построена с учетом недостатков, выявленных при анализе первой. Схема применения второй эвристики представлена на рис. 16.

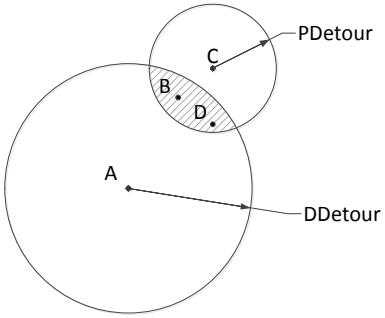

Рис. 16. Схема применения второй эвристики.

Суть второй эвристики заключается в том, что осуществляется выбор точек, находящихся на пересечении окружностей радиусов PDetour и DDetour. За счет этого все точки являются потенциально достижимыми как для водителя, так и для пассажира, при этом отпадает необходимость определения угла, ограничивающего область выбора. При этом, как и в первой эвристике, рассматриваемая область может расширяться за счет увеличения количества точек пути водителя, удовлетворяющих условию (1) или (2).

Обе эвристики для эффективной работы требуют учета ряда ограничений:

- Большое количество водителей. Если водителей будет мало, то из-за жесткого отсева поиск точек будет редко давать положительный результат.
- Малое значение DDetour. При больших значениях DDetour эвристика будет бесполезной.
- Равномерное распределение дорог по карте. Неравномерность распределения (наличие рек, озёр, заводов и т.п.) ведет к отсутствию дорог в некоторых секторах, что может привести к потере возможных точек встречи, связанных с необходимостью объезда препятствия и заезда за пешеходом с другой стороны при больших значения DDetour.

После применения эвристик осуществляется расчет параметров нового пути для всех пар точек. Этими параметрами являются новая длина пути водителя, расстояние, преодолеваемое пассажиром до точки встречи, время ожидания водителя и пассажира.

Расчет параметров нового пути начинается с проверки, не совпадает ли точка встречи с одной из точек существующего пути водителя. Если совпадения нет, то вычисляется положение точки встречи относительно существующих точек. Для этого берутся две соседние точки существующего пути и находится расстояние от новой точки до выбранных. Новая точка будет располагаться между парой, дающей наименьшее расстояние. Как в случае добавления, так и в случае совпадения необходимо сохранить индексы положения точек для дальнейших расчетов. На рис. 17 представлен вариант пересчета пути водителя, в котором точка встречи F не совпадает с опорными точками пути водителя C_i и C_{i+1}. Опорными точками пути считаются точки встречи с пассажирами, которых выбранный водитель согласился подвезти. Точка A на рисунке обозначает точку пути пассажира.

После получения положения точек встречи в списке необходимо проверить наличие свободных мест у водителя на отрезке между этими точками. Работа алгоритма прекращается, если свободные места отсутствуют. Если свободные места есть, то производится расчет новой длины пути водителя, расстояния, преодолеваемого пассажиром до точки встречи и времени ожидания водителя и пассажира.

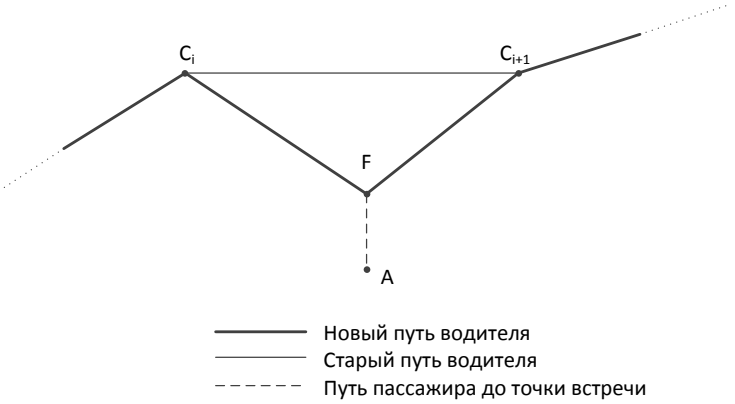

Рис. 17. Добавление новой точки к пути водителя.

Также, для всего пути пересчитываются временные метки для каждой точки с учетом времени ожидания и времени, затрачиваемого на достижение точек встречи. По результатам этих расчетов будет принято решение о выборе наилучшего пути. Критерии, по которым осуществляется принятие решения, определяют сами пользователи при передаче информации в интеллектуальное пространство. Этими критериями могут быть:

- Минимальная длина общего пути (актуально для водителя);
- Минимальное время ожидания в точке встречи;
- Минимальное расстояние до точек встречи (актуально для пассажира).

После выбора пути, наиболее удовлетворяющего заданным критериям, осуществляется изменение параметров всех точек, находящихся между точками начала и конца совпадающего пути, которое происходит следующим образом:

- Для пассажира значение параметра driveByVehicle меняется с FALSE на ID водителя, который согласен его подвезти, например:

```
('user2point*', 'drivebyvehicle', 'FALSE') →
('user2point*', 'drivebyvehicle', 'user1')
```

- Для водителя число свободных мест на всем отрезке пути становится меньше на 1, например:

  `('user1point1', 'vacantseats', '4')` → `('user1point1', 'vacantseats', '3')`

- Также следует изменить значение параметра previousPoint. В параметр добавленной точки копируется содержимое этого параметра следующей за ней точки, а в параметр previousPoint следующей точки записывается имя добавленной точки, т.е.:

  `p[1], ... , p[i], p[i+1], … ,p[n]` — начальный список точек. i — индекс точки, после которой будет добавлена новая точка.

  `p[1], … , p[i], p[i+1], p[i+2], … p[n+1]` — список после добавления, p[i+1] — новая точка.

  ```
  p[i+1].previosPoint = p[i]
  ```

  ```
  p[i+2].previousPoint = p[i+1]
  ```

После согласования путей в терминах модели, необходимо отобразить совпадающие пути на карте. Делается это, как и в случае с одним пользователем, с помощью построения URL, задающего порядок следования точек в маршруте. Отличие состоит в том, что этот URL содержит в себе не только индексы всех точек пути, но и метки, определяющие, в какой точке происходит встреча и до какой точки продолжается совместное передвижение.

Результаты вычислений переносятся в интеллектуальное пространство и пользователи, для которых искались совпадающие пути, автоматически получают уведомления о результате вычислений.

6.5.Оценка работы алгоритма. Тестовые испытания проводились на компьютере с центральным процессором Intel Pentium 4 с тактовой частотой 1,6 ГГц, ОЗУ — DDR1 512 Мб. Испытания представляли собой прогон алгоритма на случайных значениях координат точек начала и конца, выбранных на карте Хельсинки, для заданного числа водителей и пассажиров. Результаты тестовых испытаний представлены в таблице.

Результаты тестовых испытаний

Число водителей	Число пассажиров	Время поиска точек совпадающего пути
1	1	0,0135
5	5	0,0316
10	10	0,0641
20	20	0,2248
40	40	1,5462
60	60	2,2416
80	80	3,4725

На рис. 18 представлена графическая зависимость времени выполнения алгоритма от количества пользователей в системе.

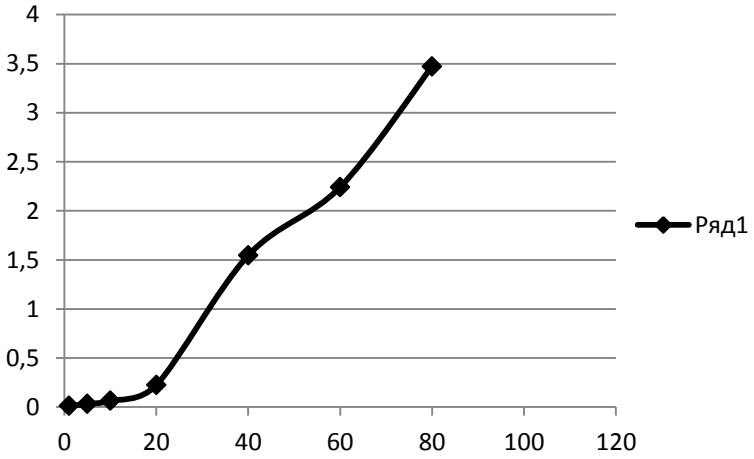

Рис. 18. Зависимость времени выполнения от количества пользователей.

7. Реализация логистической системы поиска попутчиков для водителей. Информационный брокер и программное обеспечение информационных агентов реализованы с использованием высокоуровневого, интерпретируемого языка Python, модулей python-qt для построения GUI на основе Qt4 и smart-m3 для обеспечения работы с интеллектуальным пространством.

Вследствие того, что целевой аудиторией являются обычные пользователи, а местом применения будут являться крупные города, акцент делался на мобильность и удобство пользования. По этим причинам в качестве клиентских устройств были выбраны мобильные компьютеры Nokia N810 и N900, ОС которых является Maemo — дистрибутив Linux, основанный на архитектуре Debian и адаптированный для использования в мобильных устройствах.

Начиная с выпуска Maemo (Diablo) в ОС поддерживается инструментарий Qt4, предназначенный для кроссплатформенной разработки приложений с GUI на C/C++, также имеющий привязки ко многим другим широко распространенным языкам программирования (Python, Ruby, Java, PHP и другие.). Весь используемый инструментарий является Open Source проектами с бесплатными лицензиями для некоммерческого пользования.

Выбор данного инструментария обусловлен, в том числе, простотой разработки и сопровождения. Код, написанный на Python, удобен для чтения и восприятия, при этом существует огромное множество библиотек, имеющих оптимизированную реализацию общих функций. Библиотеки Qt4 и дизайнер интерфейсов Qt Designer позволяют быстро создать качественный GUI и без труда подключить его к любой программе.

На рис. 19 представлено главное окно программы, в котором можно выбрать статус пользователя в системе (рис. 20), определяющий набор используемых параметров.

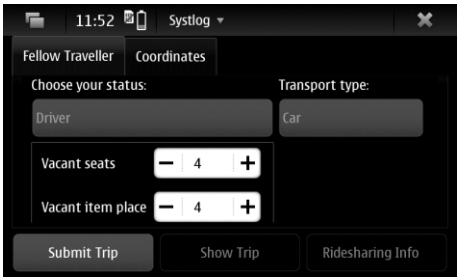

Рис. 19. Главное окно клиентской части.

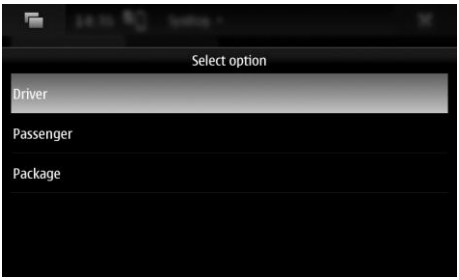

Рис. 20. Выбор статуса в системе.

Если пользователь является водителем, то он может выбрать тип используемого транспортного средства. На рис. 21 представлен диалог выбора, который включает в себя следующие типы транспортных средств: велосипед, автомобиль, семейный автомобиль и автобус. Они отличаются между собой числом мест для пассажиров и грузов.

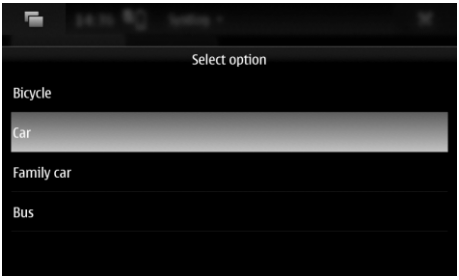

Рис. 21. Выбор транспортного средства водителя.

Для пассажира выбор дополнительных параметров не предусмотрен (рис. 22).

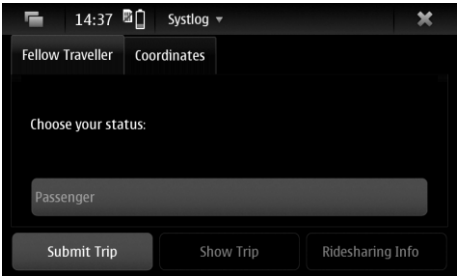

Рис. 22. Главное окно для пассажира.

Для груза возможен выбор размера груза и типа упаковки (рис. 23).

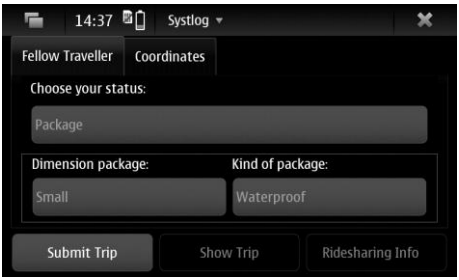

Рис. 23. Главное окно для груза.

Вкладка Coordinates используется для задания координат точек старта и конца пути, а также времени отправления и прибытия (рис. 24). Кнопки Work Location и Home Location позволяют автоматически заполнить поля координат данными из профиля пользователя, содержащими координаты Дома и Работы. Координаты можно ввести также либо вручную, либо указав точки на карте, которая открывается нажатием кнопки Map. После подтверждения выбора точки ее координаты автоматически заносятся в соответствующие поля.

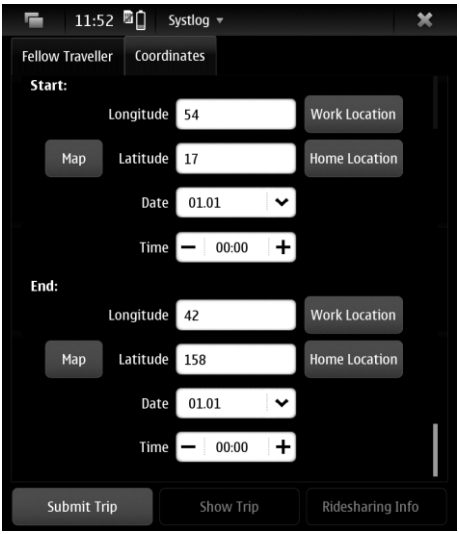

Рис. 24. Задание информации о пути.

После уточнения всех данных о пользователе и его пути, по нажатию кнопки Submit Trip, происходит соединение с Smart Space и передача в него собранных данных. По окончании вычислений пользователь оповещается всплывающим окном с информацией о результате вычислений (рис. 25).

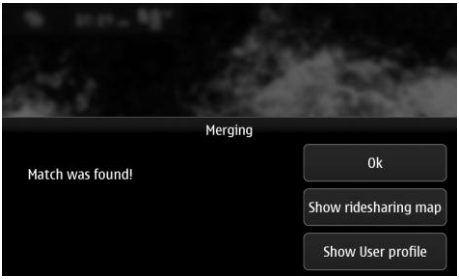

Рис. 25. Оповещение.

При этом становятся доступными действия Show Trip и Ridesharing Info. Нажатие на кнопку Show Trip покажет карту, на которой будет отображен путь пользователя без учета совпадения с другими участниками. Ridesharing Info позволяет просмотреть профиль пользо-

вателя, с которым предстоит совместная поездка, и совпадающий маршрут.

Вызов главного меню в программе осуществляется нажатием на заголовок. Оно содержит действия выхода из программы и конфигурации профиля пользователя и интеллектуального пространства (рис. 26). При выходе данные пользователя удаляются из Smart Space, сохраняется профиль и происходит завершение работы программы. Выбор действия конфигурации позволяет настроить профили интеллектуального пространства (рис. 27) и профиль пользователя (рис. 28)

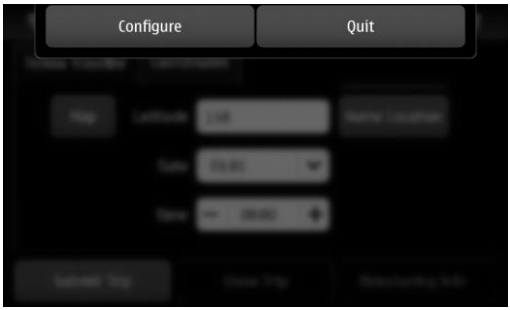

Рис. 26. Главное меню.

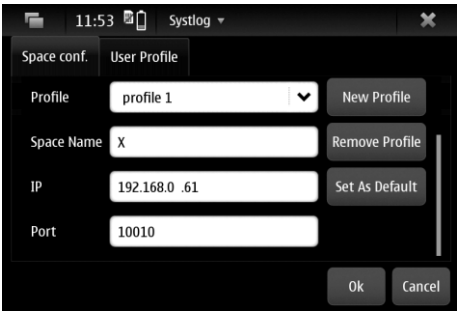

Рис. 27. Конфигурация профиля Интеллектуального пространства.

Конфигурация профиля позволяет создавать новые профили, редактировать существующие, удалять их, либо установить чей-либо профиль профилем по умолчанию. Профиль содержит в себе собственное имя, Имя Smart Space, адрес и порт, к которому программа должна обращаться в процессе работы.

Вкладка User Profile позволяет настроить профиль пользователя (рис. 28), который содержит в себе:

- Имя пользователя,
- Фотография пользователя,
- Статус в системе. В зависимости от статуса профиль может дополнительно содержать: для водителя — тип транспортного средства, число пассажиров и объем грузов, которые он может подвезти; для груза — его габариты и характер упаковки,
- Координаты расположения дома и работы,
- Максимальная задержка,
- Максимальное отклонение от изначального пути.

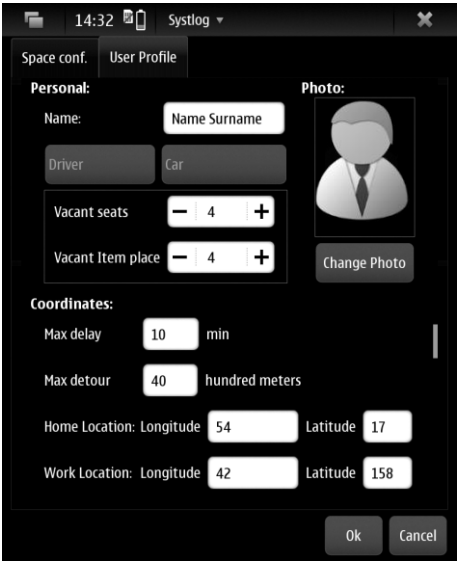

Рис. 28. Профиль пользователя.

Заключение. В результате проведённой работы была разработана архитектура, онтология и алгоритмы для логистической системы поиска попутчиков для водителей. На основе описанных выше алгоритмов реализована система, осуществляющая поиск попутчиков для во-

дителей. Результатом работы является путь, наиболее удовлетворяющий критериям заданным участниками. Система была протестирована при различном количестве участников. При 80 водителях и 80 пассажирах и карте дорог региона Хельсинки (включающего в себя город Хельсинки и его ближайшие пригороды) время поиска такого пути для каждого пассажира (или констатация факта отсутствия водителя, который может подвезти пассажира) составило 3,47 сек, что является допустимым временем для системы такого класса. Реализация системы также включает в себя пользовательский интерфейс, с помощью которого пользователи могут в удобной форме передать информацию о своем пути и получить наглядный результат поиска попутчиков.

Основное направление дальнейшей работы включают в себя:

1. Разработку бизнес-модели, определяющей экономическую составляющую в таком классе систем. Необходимо исследовать параметры, влияющие на стоимость поездки.

2. Интеграцию с социальными сетями, например Facebook и ВКонтакте. Интеграция позволит пользователям использовать свои профили из социальных сетей для авторизации в системе.

3. Проведение моделирования работы системы с использованием различных эвристик. Необходимо выявить количественные оценки повышения производительности алгоритмов при использовании эвристик.

4. Исследование возможности использования сторонних геоинформационных систем, например, Google Maps.

Литература

1. Carpool // Wikipedia. URL: http://en.wikipedia.org/wiki/Carpool (дата обращения: 3.02.2011).
2. Совместные поездки с попутчиками набирают популярность. Карпулинг приходит в Россию // Интернет-газета «Фонтанка». Дата обновления: 29.12.2010. URL: http://www.fontanka.ru/2010/12/29/017/ (дата обращения: 15.02.2011).
3. Давай со мной // [Электронный ресурс]. URL: http://www.gowithme.ru/ (дата обращения: 15.02.2011).
4. Довезу!ру // [Электронный ресурс]. URL: http://www.dovezu.ru/ (дата обращения: 15.02.2011).
5. eRideShare.com // [Электронный ресурс]. URL: http://erideshare.com/ (дата обращения: 16.02.2011).
6. PickupPal // [Электронный ресурс]. URL: http://www.pickuppal.com/ (дата обращения: 16.02.2011).
7. Zimride // [Электронный ресурс]. URL: http://www.zimride.com/ (дата обращения: 16.02.2011).

8. RideshareOnline // [Электронный ресурс]. URL: http://www.rideshareonline.com/ (дата обращения: 16.02.2011)
9. Rideshare 511 // [Электронный ресурс]. URL: http://rideshare.511.org/ (дата обращения: 17.02.2011)
10. CarJungle // [Электронный ресурс]. URL: http://www.carjungle.ru/ (дата обращения: 15.02.2011)
11. Avego // [Электронный ресурс]. URL: http://www.avego.com/ (дата обращения: 16.02.2011)
12. *Jukka Honkola, Hannu Laine, Ronald Brown, Olli Tyrkkö*. «Smart-M3 Information Sharing Platform». FRUCT. Системн. требования: Acrobat Reader. URL: http://fruct.org/conf7/Honkola_Smart_M3.pdf (дата обращения: 15.02.2011)
13. *T. Berners-Lee, R. Fielding, L. Masinter*. RFC 3986 – Uniform Resource Identifier (URI): Generic Syntax. // [Электронный ресурс]. Дата обновления: январь 2005. URL: http://tools.ietf.org/html/rfc3986 (дата обращения: 15.02.2011)
14. Resource Description Framework (RDF) // [Электронный ресурс]. Дата обновления: 7.03.2010. URL: http://www.w3.org/RDF/ (дата обращения: 18.02.2011)
15. Smart-M3. Wikipedia. // [Электронный ресурс]. Дата обновления: 23.12.2010. URL: http://en.wikipedia.org/wiki/Smart-M3 (дата обращения: 19.02.2011)
16. *Alexander Smirnov, Alexey Kashevnik, Nikolay Shilov, Sergey Balandin, Ian Oliver, Sergey Boldyrev*. On-the-Fly Ontology Matching in Smart Spaces: A Multi-Model Approach. In: Smart Spaces and Next Generation Wired/Wireless Networking // Proceedings of the Third Conference on Smart Spaces, ruSMART 2010, and 10th International Conference NEW2AN 2010, St.Petersburg, Russia, August 23-25, 2010. Springer, LNCS 6294, pp.72-83.
17. *A. Smirnov, A. Kashevnik, N. Shilov, H. Paloheimo, H. Waris, S. Balandin*. Smart Space-Driven Sustainable Logistics: Ontology and Major Components, Sergey Balandin, Andrei Ovchinnikov (eds.) // Proceedings of the 8th Conference of Open Innovations Framework Program FRUCT, Lappeenranta, Finland, 9-12 Nov., 2010. St. Petersburg: Saint-Petersburg State University of Aerospace Instrumentation, 2010. Pp. 184-194. ISBN 978-5-8088-0567-5.

Кашевник Алексей Михайлович — канд. техн. наук; старший научный сотрудник лаборатории интегрированных систем автоматизации учреждения Российской академии наук Санкт-Петербургского института информатики и автоматизации РАН (СПИИРАН). Область научных интересов: управление знаниями, профилирование, онтологии, интеллектуальные пространства, логистические системы. Число научных публикаций — 85. alexey@iias.spb.su; СПИИРАН, 14-я линия, д.39, Санкт-Петербург, 199178, РФ; р.т. +7(812)328-8071, факс +7(812)328-0685.

Тесля Николай Николаевич — студент факультета компьютерных технологий и информатики Санкт-Петербургского Государственного Электротехнического Университета «ЛЭТИ» (СПбГЭТУ). Область научных интересов: онтологии, интеллектуальные пространства, защита информации. nick.teslya@gmail.com; СПИИРАН, 14-я линия, д.39, Санкт-Петербург, 199178, РФ; р.т. +7(812)328-8071, факс +7(812)328-0685. Научный руководитель — Кашевник А.М.

Kashevnik Alexey — Ph.D.; senior researcher of the laboratory of computer aided integrated systems institution of the Russian Academy of Sciences St.Petersburg Institute for Informatics and Automation of RAS (SPIIRAS). Research area: knowledge management, profiling, ontologies, smart-spaces, logistics systems. Number of publications — 85. alexey@iias.spb.su;

SPIIRAS, 14th Line V.O., 39, Saint-Petersburg, 199178, Russia; office phone +7(812)328-8071, fax +7(812)328-0685.

Teslya Nikolay — student of Saint Petersburg Electrotechnical University (SPbETU). Research area: ontologies, smart-spaces, information security. nick.teslya@gmail.com; SPIIRAS, 14th Line V.O., 39, Saint-Petersburg, 199178, Russia; office phone +7(812)328-8071, fax +7(812)328-0685. Research manager — Kashevnik A.

Поддержка исследований. В публикации представлены результаты исследований, выполненные при поддержке Финско-Российского консорциума университетов в области телекоммуникаций (FRUCT), а также исследований, поддержанные грантами РФФИ 10-07-00368-а, программой Президиума РАН «Интеллектуальные информационные технологии, математическое моделирование, системный анализ и автоматизация» (проект 213) и Министерством образования и науки в рамках ФЦП «Научные и научно-педагогические кадры инновационной России на 2009-2013 годы» (госконтракт № 14.740.11.0357).

Рекомендовано лабораторией ИСА, зав. лаб. А.В. Смирнов, д-р техн. наук, проф.
Статья поступила в редакцию 01.07.2011.

РЕФЕРАТ

Кашевник А.М., Тесля Н.Н. **Архитектура логистической системы поиска попутчиков для водителей.**

На сегодняшний день в крупных российских городах наблюдается резкое увеличение транспортных средств и, как следствие, увеличение нагрузки на существующие транспортные сети и ухудшение экологической обстановки в регионе. Одной из возможных мер по решению данной проблемы является совместное использование автомобилей. Например, сотрудники компаний часто подвозят друг друга к месту работы. Некоторые люди сами отыскивают водителей через общих знакомых или различные интернет-сервисы и форумы. В странах Западной Европы и Северной Америки подобные совместные поездки существуют еще с 1970-х годов и называются «карпул» (carpool, также известный, как ridesharing). Совместное использование автомобилей позволяет существенно разгрузить транспортные сети и более рационально использовать ресурсы автомобилей.

В статье описывается архитектура логистической системы поиска пассажиров для водителей, которая основана на концепции интеллектуальных пространств. Проработана онтология системы поиска попутчиков, включающая в себя три типа сущностей: пользователей, транспортные средства и маршруты. Для каждой сущности выделяются уникальные атрибуты. В статье онтология представлена как в графическом виде, так и в виде RDF-троек, используемых при реализации системы. В статье представлена архитектура логистической системы, основными элементами которой являются интеллектуальное пространство, информационные агенты и информационный брокер. Информационными агентами являются программные модули, установленные на портативных пользовательских устройствах, среди которых можно выделить пассажиров и водителей. Информационный брокер – это программный модуль расположенный на устройстве с большим быстродействием, осуществляющий поиск совпадающих частей пути на основе информации, передаваемой пассажирами и водителями в интеллектуальное пространство и информации, запрашиваемой у географической информационной системы. Особое внимание уделено алгоритмам поиска совпадающего пути и выбору точек встречи.

Для демонстрации возможностей системы разработан прототип автоматизированной системы поиска попутчиков для водителей. Также, для информационных агентов разработан прототип пользовательского интерфейса, с помощью которого они могут отправлять запросы на поиск попутчика и получать ответ о результате поиска.

SUMMARY

Kashevnik A.M., Teslya N.N. **Architecture of logistics system finding fellow-travellers for drivers.**

Nowadays a significant increase of the number of vehicles, raising the pressure on the existing transport networks and environmental degradation, is observed in the major Russian cities. One of the possible measures to solve this problem is carpooling. For example, employees can bring up each other to their working place. Some people find the drivers themselves through friends or a variety of Internet services and forums. In Western Europe and North America carpooling exists since 1970 and is also known as ridesharing. The carpooling can significantly relieve the traffic network and improve cars' using.

The paper describes the architecture of the logistics system in finding fellow-travelers for drivers and based on the idea of smart spaces. The system is based on the ontology including three types of entities: users, vehicles and routes. In the paper the ontology is presented in a graphical form, as well as in the form of RDF-triples used in case study. Main elements of the architecture are intelligent space, information agents and information brokers. Information agents are program modules installed in portable users' (passengers' and drivers') devices. Information broker is a software module that searches matching parts of routes on the basis of information provided by passengers and drivers to the smart space and information requested from a geographic information system (GIS). Due to the significant complexity of the algorithm for finding appropriate fellow-travelers for drivers as well as definition of the pick-up and drop-off points meeting requirements of both drivers and passengers, it is described in detail. The usage of proposed heuristics significantly reduces the dimension of the task and speeds up its solving.

The developed prototype demonstrates the possibilities of the architecture and its underlying components. The user interface developed for the information agents can be used for interaction with the system.