Agent-based system for highway gasoline price negotiations

Sylwia Nowak
Warsaw University of Technology
Poland

Maria Ganzha
Polish Academy of Sciences
Poland

Marcin Paprzycki
Warsaw Management Academy
Poland

Costin Bădică
University of Craiova
Romania

Mirjana Ivanović
University of Novi Sad
Serbia

Sabin Simionescu
University of Craiova
Romania

ABSTRACT

Modern cars are getting intelligent, including ability to contact each-other and various entities in the "the outside world". At the same time, group purchases gain popularity (e.g. Groupon). Furthermore, agent-based autonomous price negotiations have been widely researched. The aim of our work is to combine these trends, to develop an agent-based system for group-style gasoline price negotiations. Specifically, the proposed system is to reward drivers for (re)fueling (and shopping) at gas stations. Negotiations concerning the reward level are to involve autonomous agents representing cars and stations. This paper describes the proposed system, its current state of implementation, and illustrates its work in a simulated environment.

CCS CONCEPTS

• **Information systems** → **Mobile information processing systems**; • **Applied computing** → *Electronic commerce*;

KEYWORDS

multi-agent system, price negotiations, cars communities

ACM Reference format:
Sylwia Nowak, Maria Ganzha, Marcin Paprzycki, Costin Bădică, Mirjana Ivanović, and Sabin Simionescu. 2017. Agent-based system for highway gasoline price negotiations . In *Proceedings of Balkan Conference in Informatics, Skopje, September 2017 (BCI'17),* 8 pages.
https://doi.org/10.1145/nnnnnnn.nnnnnnn

1 INTRODUCTION

Growth of the Internet resulted in, among others, development (in theory and in practice) of new trends in distributed systems, including (e-)commerce. Among them one can find: (1) group purchases, where goods are discounted for groups of customers (e.g. Groupon [22]), (2) customer loyalty programs, where clients are rewarded for making multiple purchases from the same vendor(s) (e.g., Payback Program [9]), (3) dynamic pricing mechanisms, where prices of goods are adjusted in response to market conditions (e.g. hotel rooms, which are cheaper during week-ends and more expensive during work-week), (4) use of auction-type mechanisms in online transactions (e.g. eBay [20]), (5) use of software agents in e-commerce (see, for instance, [12–14] and references compiled there, and consider sniper agents developed for the Allegro [10] auctions) and in the context of car fleet management [24], and (6) fast development of "intelligent cars", capable of communicating and participating in price negotiations (see, among others [17–19, 27, 29]).

In this context, let us assume that multiple drivers desire to fill their cars' tanks, and they are in an area where multiple gas stations are present. Observe that what follows can involve both, "driving on the highway", as well as "driving in a city" scenarios. Using already existing technologies, it is possible to envision the following scenario. Drivers (represented by some software artifacts working in their cars) communicate and "get together", collect their gasoline needs, and one of them contacts gas stations in the area, asking for a "group offer". Next, offers received from gas stations are evaluated (by the "group leader", or using some voting scheme) and one station is selected for the group. In what follows cars sign up to the selected deal (as some of them may not be happy by the selection and withdraw). List of cars that are in the ad-hoc formed group is communicated to the selected station. Starting from this moment, for some pre-specified time (which may also be negotiated), members of this group arrive at the selected station and get fuel at a discounted price. Obviously, this scenario can be enriched by, among others, adding trust management, or multi-round negotiations, but let us omit these interesting considerations and focus on the core functionality of just introduced scenario.

It should be clear that this process can be completed semi-autonomously. Specifically, it can be assumed that, after the driver expresses her preferences (e.g. which stations she likes to use, which is her second choice and which stations she strongly prefers to avoid), the remaining actions can be undertaken by the software. In an "intelligent car", an application controls the level of fuel and, when needed, (autonomously) initiates the process to find the (best / matching owner preferences / selling cheapest fuel) gas station to refuel (and, possibly, purchase additional items, e.g. a coffee and a sandwich). In the station, another application represents the owner and tries to maximize income, when selling gasoline

BCI'17, September 2017, Skopje

ACM ISBN 978-x-xxxx-xxxx-x/YY/MM. . . $15.00
https://doi.org/10.1145/nnnnnnn.nnnnnnn

(and other products). This user story makes it obvious that software agents ([21, 26, 28]) are a very natural mechanism for realizing the needed infrastructure. Therefore, we propose a software agent-based system, enabling autonomous negotiations between groups of drivers and gas stations, concerning fuel price discounts and, possibly, other benefits. For simplicity of the initial version of the system, we assume that all benefits are to be realized as "*points*", collected by drivers (and later exchanged for items available at the stations). However, this assumption is not a limiting one (more realistic reward system can be applied).

Before proceeding, let us make a few comments. (1) Today, majority of chains of gas stations feature loyalty programs. However, these programs are limited to a single company ("bonuses" can be used only within given chain). Nevertheless, in our system, we assume that all stations can join a common platform. While this assumption is somewhat unrealistic, we have decided to make it, to simplify development of initial system. However, participation in a common platform does not contradict existence of separate loyalty programs. For instance, drivers may decide (when initializing their car agents) that they will buy gas from chains X and Y (if only possible). This preference can be used to evaluate proposals and establishing the make-up of the group joining the final contract. (2) Process of exchanging points for products, at gas stations, is out of scope of this paper. (3) While trust management is a key issue in a real-world system, it is also out of scope of our current work. Nevertheless, large body of work, concerning trust management in agent systems, exists [15, 30], and results obtained there can be directly applied to the proposed scenario. (4) Establishing negotiation strategy by the gas station (e.g. using data mining), while very interesting, is also out of scope (in particular, since we do not have access to real-world data). (5) The same applies to variety of possible proposal evaluation mechanisms.

This being said, we proceed as follows. In Section 2 we briefly outline selected related work. We follow, in Section 3, with the description of the developed system. Working system is described in Section 4. We conclude the paper with a summary and description of future work.

2 RELATED WORK

Automated negotiations, based on use of autonomous software agents, have been considered in multiple domains. Various approaches are presented for Intelligent Transportation Systems and (semi-)autonomous car services: dynamic parking negotiation [16], Parking Negotiation and Guidance [25], car pooling negotiations [23], dynamic vehicle routing [11].

In [16] authors presented an intelligent agent system that takes care about negotiable parking prices and selects the car parking place which has been established to be optimal for the driver. Inspired by modern cities, and problems with parking places and prices, they focused on establishing advanced parking assistant systems to provide drivers with parking information. Usual parking information systems ignore parking prices and do not provide car parking places matching drivers' demand. Parking prices are not negotiable and consumers lose bargaining position (to get cheaper parking). So, authors propose an intelligent agent system to solve this problem.

Paper [25], also presents an intelligent parking negotiation and guidance system that integrates mobile agent and a multi-agent system. Bargaining platform allows dynamic, stable and fast negotiation between cars and parking places.

Paper [23], brings an interesting agent-based solution for car pooling problem. Proposed model is used for simulating interactions of autonomous agents with their agenda. They negotiate, to adapt their schedules, based on preferences of individuals, time of the day, and duration of the trip.

An agent-based system to enable dynamic vehicle routing is presented in [11]. System deals with the dynamic vehicle routing problem (in fact, a pick-up problem) where some tasks can be dynamically transferred to other vehicles if the work time can exceed divers' daily work limit. This is achieved through, specially developed, negotiation protocol, allowing collaborative decision making (among agents) and adjustments during the course of the planned route.

All mentioned approaches are connected to optimization processes in Intelligent Transportation Systems and car services, where (agent-based) negotiation and planning play important role. But, to our knowledge, there are no published research papers that deal with group negotiations of gasoline prices, based on application software agent technologies, which is the essence of our approach. In this way, we extend boundaries of applicability of agent technologies to a new domain (which is naturally very well suited for them).

3 SYSTEM OVERVIEW

As stated in Section 1, for group negotiations of gasoline prices we will apply software agents. Here, agents "placed in cars" are going to represent preferences of car drivers. For instance, they will know that driver $D1$ uses standard gasoline and prefers to buy it at stations of company X (but is also willing to buy it in stations belonging to company Z), while driver $D2$ uses premium gasoline and likes to refuel in stations of company Y (but will do her best to avoid visiting stations belonging to company W). We assume also that such "car agents" have access to information such as: level of gas in the tank, gas consumption (and thus the available driving distance), navigation software (including locations of gas stations). This information, allows them to (autonomously) establish when the car needs to be refueled, and where are the, near-by, gas stations. Here, observe that, at the time of writing of this contribution (June 2017) such information is already available in majority of modern cars (class C and above). Obviously, adding other preferences is possible (and relatively simple) and will be done in the near future.

Agents representing gas stations can have a complex criteria build-in. Such criteria can allow creation of a comprehensive business strategy based on: (i) day of the week, (ii) holidays (upcoming and current), (iii) time of the day, (iv) global promotions in the chain of stations, (v) gasoline levels (and date of next delivery), etc. Proposed strategy can be

established by the management, or be a result of data mining applied to data collected at the station. However, development of the gas station strategy is out of scope of this paper. However, it will also be considered in the future.

Based on these assumptions we have developed an agent system, which includes 4 main modules.

- Mobile application for drivers, realized via agents running on smart phones. This application is used (a) for drives to specify their preferences, and (b) to participate in price negotiations (communicating with other driver-agents and gas-station agents).
- Agent-based application for gas stations. Its main role is negotiating with representatives of groups of drivers. Currently implemented mobile application can be easily transformed into a server-based system (e.g. for real-world deployment). Such system would collect and analyze data generated at the station. It can also be a part of a more comprehensive infrastructure belonging to a chain of gas stations, where data for all stations would be collected and analyzed. However, as a proof-of-concept, a lightweight Android-based application has been developed.
- Central unit, realized by agents running on a server. Their role is to control and track information exchanged between software agents representing gas stations and drivers. Central unit is connected directly to a database, storing information about all actors in the system.
- Simulation module, realized as a separate server application. It generates the environment, with cars and gas stations, and enables simulating drivers trips, cars changing locations, and vehicle data. It also creates gas stations and their strategies used during negotiations. Finally, it creates as many agents (representing simulation participants) as needed, and generates data for simulations.

Figure 1 depicts the main modules of the system and their interrelations.

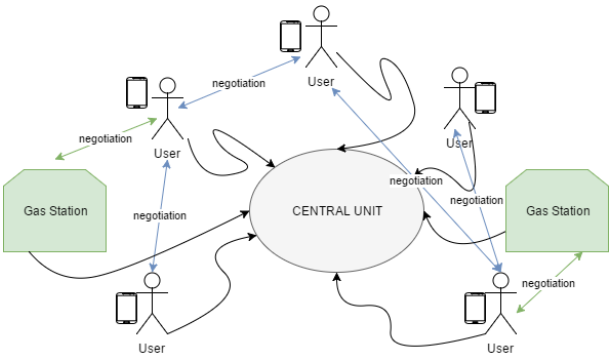

Figure 1: Multi agent system for negotiations

It can be observed that gas station agent negotiates directly with only one driver agent, which is called the "representative agent". This representative agent communicates with other driver agents that are potentially willing to participate in group purchase. Driver agent becomes a representative agent by starting process of seeking discounted gasoline. Then, it obtains information about other cars that may be interested in participating in price negotiations, and contacts them to establish the initial group makeup. This information is available from the central unit, which stores information about current location of all cars (and their states). While this approach is not likely to scale, it was the simplest way of facilitating the core functionality (price negotiations). However, we plan to return to this issue and modify the proposed architecture to make it more scalable.

Diagram shows that the central unit communicates with all agents, present in the system, bidirectionally. Green arrow represents negotiating between the representative agent and the gas station agent. The blue one indicates negotiations between driver agents and the representative agent. Note that one driver can participate in two, or more, negotiations at the same time. Issues involved in strategic decisions of the type: "should I accept the current proposal, or should I wait for the remaining 2 negotiations to complete", while very interesting and worthy further investigation, are out of scope of current contribution. Drivers that are too far and/or prefer to not to refuel, are not included in group forming.

Let us now describe, in more detail, each of the modules of the developed system.

3.1 Application for drivers

Let us start from the mobile application for drivers. It enables users to create new accounts, or use existing ones, to log into the system called *SmartRoad*. User interface (seen in Figure 2) enables configuring preferences. Currently, the following preferences can be specified: favorite and not-liked gas station chains, minimum time left for the trip before the vehicle must be refueled, maximum refuel time left before negotiating could be started (i.e. do not negotiate when tank 1/2 full and gas will suffice for 300 km). It is also possible to enable vehicle agent to select gas stations and make final decisions on its own. Obviously, as indicated above, list of preferences can be easily extended (as needed) within the current system setup.

Application is assumed to be connected with the car computer, to receive the necessary information to estimate when to negotiate with other agents (the "time left" parameter), when to start its own search for gas, etc. Obviously, in the current version of the simulation, a vehicle data generator – developed during the project – is used (it is a part of the simulation module).

Currently, it is assumed that the driver defines the start and end points for a given trip. Here, the system suggests places with Google Places API [5] and navigates with Google Maps API [4]. This allows use of Google Maps ([3]) for the display. Obviously, any other map software can be applied (if needed).

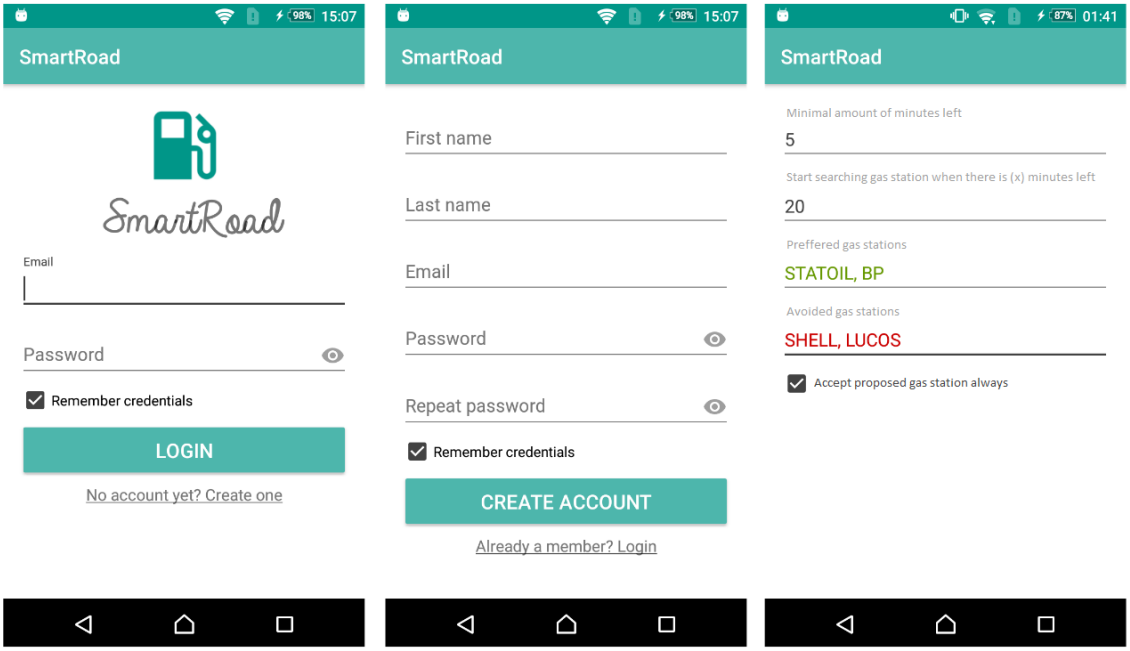

Figure 2: Mobile application for drivers (1)

As a result of negotiations, the display of the application for drivers (on the mobile device) displays a notification about number of possible reward points, time window to reach the selected gas station and complete the purchase, and the current number of drivers that are ready to join the group. Then the user can accept or reject the offer. To do this she presses button "thumb up" join group for a given station or "thumb down" to reject the proposal (and continue search). Obviously, in the case when the driver agent is allowed to make decisions fully autonomously, driver will not be distracted from driving, but informed where to go to refuel (using standard build-in mapping software). Figure 3 presents the starting point of the trip, the navigation route to the end point, and the information about the proposed gas station.

The following agents have been instantiated to support the required functionality of the application for the drivers:

- LoginRegisterAgent – communicating with the Central Unit to log-in or to register new user.
- ConfigurationAgent – updating driver's settings on the server (part of the Central Unit).
- TrackerAgent – updating the Central Unit with the current location of the car and the trip information.
- NegotiatorAgent – responsible for negotiating with the gas station (agent) and informing other drivers' agents. In the current setup, only stations that are not on the list of not-preferred, and located in the neighborhood of the current location of the car are invited to negotiate with this agent.
- ActionAgent – responsible for saving information on the server (the Central Unit) about actions undertaken by the driver/her agent, such as: visiting a specific gas station, rejecting proposal to join existing group, etc. Currently this data is "just collected". However, obviously, it could be further analyzed.

3.2 Application for gas station's representatives

To participate, gas stations have to register in the *SmartRoad* system. Here, an Android application enables creating a new account and configuring it with the company name, address, contact information and logo that will be presented to the users. It is assumed that gas station software agent generates strategy for negotiating with the drivers. Currently it is implemented as a random point rewards from a fixed interval, for a random size of group (obviously, this strategy makes sense only for testing core functions in the system). Strategy can be changed at any moment, but it will not modify existing contracts. If negotiation between representative agent of a new customers group finishes with success, application receives notification about details of the deal (and displays them to the station representative). Gas station can configure its application to receive notifications about ongoing negotiations so that the strategy can be changed even during the negotiations. However, this feature has been put in place for future experiments with the system (to make it more robust). Figure 4 presents the log-in form with the user name and the secret code, configuration panel – enabling changing displayable information about the gas station, and editable negotiation strategy view, with strategy for a given day.

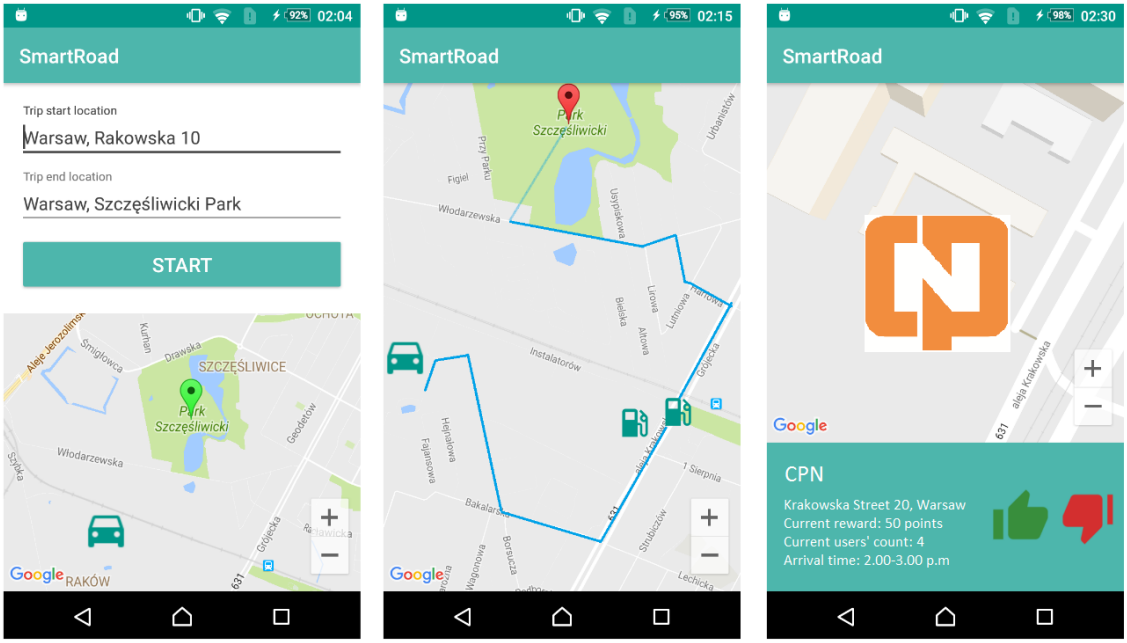

Figure 3: Mobile application for drivers (2)

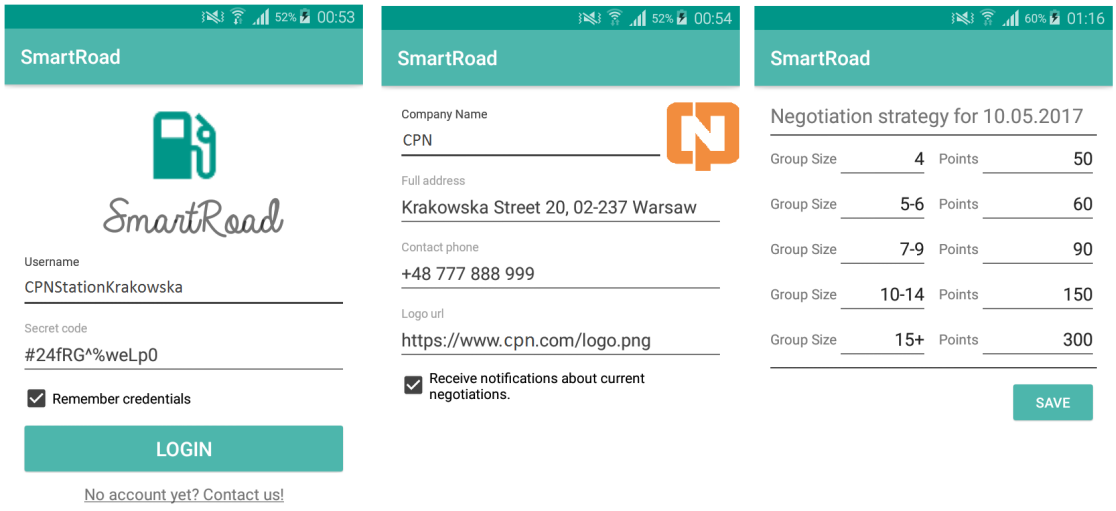

Figure 4: Mobile application for gas stations

The following agents have been instantiated to deliver functions needed by the gas station application:

- LoginRegisterAgent – communicating with the Central Unit to log in, or to register, the gas station. It also can update information about the station at the server (as needed).
- ConfigurationAgent – updating gas station's negotiation strategies.
- NegotiatorAgent – negotiating with driver agents.
- TrackerAgent – informing the server about negotiations and their results.

3.3 Central Unit

Central Unit is responsible for channeling selected information to appropriate software agents. This module registers and logs-in the drivers (their agents). It stores information about active users, their preferences, current location, trip destination and desire to negotiate. Central Unit passes data

about drivers software agents' to the representative agent. It also provides information needed for the representative agent to communicate with gas stations' agents. Central Unit is connected with a MySQL database [8], storing all information crucial to the functioning of the system: drivers' and stations' details, drivers' agents decisions and stations' negotiation strategies. Each software agent in the system can find other software agents via the Central Unit. Central Unit address is available to the remaining modules of the designed system. Note that, to make the proposed system more focused, we have decided to use our own approach to "agent management" instead of using, so called, Directory Facilitator module, available in the JADE agent platform ([6]). Each module contains configuration file containing information about names and addresses of all Central Unit's software agents. Central Unit is the one that runs the JADE "MainContainer" (see, [6]). There, the following software agents reside:

- LoginRegisterAgent – responsible for log in and registering drivers and gas stations.
- ConfigurationAgent – updating drivers and gas stations configuration.
- TrackerAgent – tracking drivers and gas stations deals.
- ActionAgent – receiving information that should be stored in database, like actions completed by the drivers.
- HelperAgent – sending information to representative agents that seek to start negotiations, about other driver agents ready to negotiate.

3.4 Simulator unit

The implemented solution contains also a simulation module. Its main role is to represent the environment, in which the negotiations take place (e.g. a highway, or a city, or its part). Furthermore, in the simulator, one can configure the initial values defining: driver accounts and their preferences, current trips, and vehicle data. Simulator creates gas stations equipped with their negotiation strategies. All simulated data is prepared within a single XML configuration file. Trips and current locations of vehicles are simulated with the GoogleMaps API [4], enabling generating location-points in a selected path within a specific map. This approach enables fast changing the observable simulated environment. All information connected with the negotiation, like: negotiation between representative agent $R1$ and gas station $P1$ started, $R1$ communicated with drivers agents $D1$, $D2$ and $D3$, are presented within the standard output of the simulator. Communicates sent and received by all agents in the system are fully logged. Currently, the simulator is using only a "command line" interface, which will be upgraded to represent ongoing negotiations in a more human-consumable format.

4 EXPERIMENTS

Let us now illustrate, how system works in two scenarios. These two scenarios present (in next two sections) (1) how driver's agent behaves when it becomes a representative agent for a negotiation (Scenario 1), and (2) how the representative agent reacts to new propositions of a driver agent that would like to join an existing group (Scenario 2). Obviously, these two scenarios represent two snapshots of the same process (initial stages of gasoline purchase), viewed from the perspective of the group leader and of the group members. The complete process considered here, as actually being executed, is described in Subsection 4.3.

4.1 Scenario 1 - Starting group negotiation

Let us start from the description of the process of starting group negotiations. The description follows what has actually happened in an experiment that was executed. However, due to the lack of space, it was impossible to include here, for instance, actual logs generated in the process. Nevertheless, these logs (full print-out) can be found in [2].

- It's 11:00 a.m.
- Driver1 agent realizes that it needs to purchase gasoline and requests information about available gas stations and their offers (form the Central Unit agent).
- Driver1 agent selects Station1, with the best starting offer: if 4 customers will tank within one hour time window, all of them would receive 40 points.
- Driver1 agent becomes negotiation representative. This being the case, it won't be able to join other groups.
- Driver1 agent starts to seek other agents to select Station1. Driver1 agent receives continuously list of ready to negotiate, located in the neighborhood, other driver's agents.
- 4 other drivers decide to go to Station1 within 1 hour.
- Driver1 agent renegotiates with Station1 agent a better offer. Station1 proposes award 60 points for each customer.
- During whole hour Driver1 can renegotiate offer (e.g. for 20 customers everybody will get 2000 points).
- If Driver1 agent dies, negotiation stops and the last offer is proposed during whole hour to users in the neighborhood of Station1. Here, users receive information from Station1 agent.
- At 11:59 all customers (from time period 11:00-11:59) receive the negotiated amount of points. Users' received points depend on number of transactions (last Station1 offer was 2000 points but only 10 cars tanked). All users that declared to tank and did not do that lose 10 percentages of reward (-200 points).
- Driver1 receives 120 percentages of reward.

4.2 Scenario 2 – Joining group negotiation

Let us now look at this scenario from the perspective of a driver who joins the existing group.

- It's 11:45 a.m.
- Driver2 agent receives 2 notifications about possible rewards for purchasing gasoline (Station1 – 40 points for 4 customers, Station2 – 60 points for 5 customers).

- Both gas stations Station1 and Station2 are preferred by Driver2. Here, Driver2 does not allow her agent to make final decision.
- Driver2 receives notification about selected Station2. She has to decide if she accepts or rejects proposed terms.
- Driver2 rejects software agent's proposition.
- Driver2 receives second notification about Station1.
- Driver2 makes final decision – she accepts terms of Station1.
- At 11:59 Driver2 receives 100 points, because more customers did come to Station1.

4.3 Negotiation overview

Section BECOME REPRESENTATIVE in [2] presents starting negotiation by Driver1 with Station1. This situation follows Scenario 4.1. It is the beginning of a new time interval (11:00 a.m.-11:59 a.m.) so there are no contraindications to make Driver1 representative of a new group. Driver1 agent asks the Central Unit's helper agent for information about other, ready to negotiate, drivers. Upon receiving a list, it starts forming a group.

Section CREATE GROUP in [2] presents the situation when the Driver1 negotiating agent communicates with the Driver2 negotiating agent and informs it about possible reward. As described in Scenario 4.2, Driver2 accepts proposition and declares joining the group. Driver1 negotiating agent communicates also with other drivers and builds a list of potential customers of Station1 (for the time period 11:00 a.m - 11:59 a.m).

Section CONFIRM USER CAME in [2] shows the confirmation of a deal with Driver2, by the Station1 tracker agent. This information is passed to the Central Unit's tracker agent to build the list of drivers and their rewards at the end of each 1 hour time period. Central Unit generates rewards with HelperReward behavior. It can be observed that, according to Scenarios 4.1 and 4.2, both drivers did receive points. Due to being the representative, Driver1 received 120% of reward, which was 100 points.

5 TECHNICAL REQUIREMENTS

Based on our experiences with actual use of various software agent platforms, we have decided to use the JADE [6] platform to implement and test the proposed system. One of the main reasons for this selection was maturity of JADE and its flexibility in cooperating with other software artifacts. In particular, this was reported in [24], where JADE was used to develop a TAXI corporation management software (which involved, among other, agent negotiations and renegotiations, and use of mapping software).

Mobile applications have been implemented for the Android API 19 [1] and higher. Since we have decided to use JADE agent platform, we have followed the JADE for Android technology to create agent container with "MicroRuntimeServiceBinder" configured with the location of the Main Container. Running software agents in "Services" prevents Android system from killing their processes and lets them stay alive even when application is running as a background task. Each mobile device has its own container within which all software agents exist. All containers are created during start of applications.

Central Unit runs as a service supporting Java Standard Edition version 8 (see, [7]). Simulation module requires same technology.

6 CONCLUDING REMARKS

This article illustrates use of software agents in a complex system involving agent negotiations combined with geopositioning. Specifically, group price negotiations concern purchase of gasoline by a group of cars. As presented, the initial implementation has been completed and tested. This allows us to start adding features, e.g. trust management, more extensive user profile, time window parameters, negotiation strategies, etc. We will report on our progress in subsequent publications.

ACKNOWLEDGEMENT

The work presented in this paper was supported in part by PAS-RAS bilateral project "Semantic foundation of the Internet of Things", as well as a collaboration agreement between University of Novi Sad, University of Craiova, SRIPAS and Warsaw University of Technology.

REFERENCES

[1] 2017. Android Api 19. (May 2017). Retrieved May 14, 2017 from https://developer.android.com/about/versions/android-4.4.html
[2] 2017. Application Logs. (2017). Retrieved June 17, 2017 from https://pages.mini.pw.edu.pl/~ganzham/logs.txt
[3] 2017. Google Maps. (May 2017). Retrieved May 14, 2017 from https://www.google.com/maps/
[4] 2017. Google Maps API. (May 2017). Retrieved May 14, 2017 from https://developers.google.com/maps/
[5] 2017. Google Places API. (May 2017). Retrieved May 14, 2017 from https://developers.google.com/places/
[6] 2017. JADE. (May 2017). Retrieved May 14, 2017 from http://jade.tilab.com/
[7] 2017. Java Standard Edition (Java SE 8). (2017). Retrieved May 14, 2017 from http://www.oracle.com/technetwork/java/javase/downloads/jdk8-downloads-2133151.html
[8] 2017. MySQL server 5.7.18. (May 2017). Retrieved May 14, 2017 from https://www.mysql.com/
[9] 2017. Payback. (2017). https://www.payback.pl/
[10] Allegro. 2017. (2017). http://www.snajper.net/
[11] Dmontier Pinheiro Arag ao Jr., Antônio Galv ao Novaes, and Mônica Maria Mendes Luna. 2015. A multi agent based system to enable dynamic vehicle routing. *Transportes* 23, 1 (2015), 69–77. https://doi.org/10.14295/transportes.v23i1.765
[12] C. Badica, M. Ganzha, and M. Paprzycki. 2007. Developing a Model Agent-based E-commerce System. (2007), 555–578.
[13] Costin Badica, Maria Ganzha, and Marcin Paprzycki. 2007. Implementing Rule-Based Automated Price Negotiation in an Agent System. *Journal of Universal Computer Science* 13, 2 (feb 2007), 244–266. http://www.jucs.org/jucs_13_2/implementing_rule_based_automated.
[14] Costin Bădică, Sorin Ilie, Alex Muscar, Amelia Bădică, Liviu Sandu, Raluca Sbora, Maria Ganzha, and Marcin Paprzycki. 2007. Deciding equivalances among conjunctive aggregate queries. *Computing and Informatics* 54, 2, Article 5 (April 2007), 50 pages. https://doi.org/10.1145/1219092.1219093
[15] Alberto Caballero, Teresa García-Valverde, Juan A. Botía, and Antonio Fernandez Gómez-Skarmeta. 2009. A Trust and Reputation Model as Adaptive Mechanism for Multi-Agent Systems.

Inteligencia Artificial, Revista Iberoamericana de Inteligencia Artificial 13, 42 (2009), 3–11. http://polar.lsi.uned.es/revista/index.php/ia/article/view/593

[16] Shuo-Yan Chou, Shih-Wei Lin, and Chien-Chang Li. 2008. Dynamic parking negotiation and guidance using an agent-based platform. *Expert Systems with Applications* 35, 3 (2008), 805–817.

[17] Josh Constine. 2017. Val.ai lets self-driving cars bid for parking spots. TechCrunch. (May 2017). https://techcrunch.com/2017/05/14/self-parking-vehicle/.

[18] Claudia Di Napoli, Dario Di Nocera, and Silvia Rossi. 2014. Agent Negotiation for Different Needs in Smart Parking Allocation. In *Advances in Practical Applications of Heterogeneous Multi-Agent Systems. The PAAMS Collection: 12th International Conference, PAAMS 2014, Salamanca, Spain, June 4-6, 2014. Proceedings*, Yves Demazeau, Franco Zambonelli, Juan M. Corchado, and Javier Bajo (Eds.). Lecture Notes in Computer Science, Vol. 8473. Springer International Publishing, 98–109. https://doi.org/10.1007/978-3-319-07551-8_9

[19] Claudia Di Napoli, Dario Di Nocera, and Silvia Rossi. 2014. Using Negotiation for Parking Selection in Smart Cities. In *Advances in Practical Applications of Heterogeneous Multi-Agent Systems. The PAAMS Collection: 12th International Conference, PAAMS 2014, Salamanca, Spain, June 4-6, 2014. Proceedings*, Yves Demazeau, Franco Zambonelli, Juan M. Corchado, and Javier Bajo (Eds.). Lecture Notes in Computer Science, Vol. 8473. Springer International Publishing, 331–334. https://doi.org/10.1007/978-3-319-07551-8_31

[20] eBay. 2017. (May 2017). Retrieved May 14, 2017 from http://www.ebay.com/

[21] Maria Ganzha, Marcin Paprzycki, and Andrea Omicini (Eds.). 2013. Special Issue on Software Agents. *Computing Now* 6, 11 (Nov. 2013). http://www.computer.org/portal/web/computingnow/archive/november2013

[22] Groupon. 2017. (2017). Retrieved Maj 14, 2017 from https://www.groupon.com/

[23] Hussain Iftikhar, Knapen Luk, Bellemans Tom, Janssens Davy, and Wets Geert. 2015. An Agent-based Negotiation Model for Carpooling: A Case Study for Flanders (Belgium). The Transportation Research Board (TRB), 94th TRB Annual Meeting,. (2015). http://hdl.handle.net/1942/18346.

[24] M. Leszczynski, M. Niedabylski, R. Kutkowski, M. Ganzha, and M. Paprzycki. 2016. Software agents in support of a taxi corporation. In *2016 20th International Conference on System Theory, Control and Computing (ICSTCC)*. 429–434. https://doi.org/10.1109/ICSTCC.2016.7790703

[25] W. Longfei, C. Hong, and L. Yang. 2009. Integrating Mobile Agent with Multi-Agent System for Intelligent Parking Negotiation and Guidance. In *4th IEEE Conference on Industrial Electronics and Applications*. IEEE, 1704–1707.

[26] Pattie Maes. 1994. Agents That Reduce Work and Information Overload. *Commun. ACM* 37, 7 (July 1994), 30–40. https://doi.org/10.1145/176789.176792

[27] John R. Quain. 2014. Ford demo of car-to-car communication shows what happens when cars talk. FOXNEWStech. (February 2014).

[28] Stuart J. Russell and Peter Norvig. 2014. *Artificial Intelligence: A Modern Approach* (3rd ed.). Pearson Education Limited.

[29] Peter Valdes-Dapena. 2016. U.S. rules cars must talk to each other. CNNtech. (December 2016).

[30] Reda Yaich, Olivier Boissier, Gauthier Picard, and Philippe Jaillon. 2012. An Agent Based Trust Management System for Multi-Agent Based Virtual Communities. In *Advances on Practical Applications of Agents and Multi-Agent Systems - 10th International Conference on Practical Applications of Agents and Multi-Agent Systems, PAAMS 2012, Salamanca, Spain, 28-30 March, 2012*. 217–223. https://doi.org/10.1007/978-3-642-28786-2_24