International Journal of Systems and Service-Oriented Engineering
Volume 8 • Issue 1 • January-March 2018

A Case Study: Mobile Service Migration Based Traffic Jam Detection

M. Mohanned Kazzaz, Brno University of Technology, Brno, Czech Republic
Marek Rychlý, Brno University of Technology, Brno, Czech Republic

ABSTRACT

This article provides a proof-of-concept of the applicability and reusability of the authors proposed framework for web service migration through a traffic jam detection case study. The framework migrates mobile hosted web services between mobile vehicles using context-aware self-adaptive mechanism in order to guarantee service availability and quality. A decision-making process is implemented to select the best destination vehicle from between the found possible migrations based on prioritized criteria set.

KEYWORDS

Context-awareness, Mobile Host, Mobile Service, Service Migration

1. INTRODUCTION

Mobile service development and provisioning have become main focuses of nowadays researches because of the huge improvements in mobile device capabilities and the vast availability of wireless networks. As in the traditional Service Oriented Architecture (SOA) (Erl, 2005) researches, context-awareness (Abowd et al., 1999) and self-adaptation (Garlan, Cheng, Huang, Schmerl, & Steenkiste, 2004) have been the main approaches proposed to enable and leverage SOA capabilities in mobile systems (Papakos, Capra, & Rosenblum, 2010; Hoang; & Chen, 2010; Paspallis, 2008; Alkhabbas, Spalazzese, & Davidsson, 2017).

Service mobility has been proposed in ad-hoc networks to support services sharing and consuming on the fly between mobile devices (AlShahwan, Carrez, & Moessner, 2012; Wagh & Thool, 2014; Zuo & Liu, 2015). A service can be moved to perform location-based tasks (i.e., device tracking or search for surrounding devices), to process data on other devices and/or to temporally use resources of available devices in the network (such as processing power or sensors). In order to enable service mobility between mobile devices, a shared semantic understanding between devices to express the status of their specifications and requirements. Additionally, it is required to provide a semantic description to define services specifications and properties. Contextual information such as location, speed, hosted services, and service description must be semantically presented and dynamically generated to provide a real-time information and status of the participating devices and services in the system.

The possibility of having several adaptation plans for a client's service to move over a set of possible destination hosts requires a decision making process. This process provides the adaptation

DOI: 10.4018/IJSSOE.2018010103

system with a selection mechanism to decide where to migrate its service based on defined criteria that affect service quality and system adaptation in different levels.

In this work, we demonstrate a traffic jam detection scenario in peer-to-peer network as the proof-of-concept to the applicability of the Mobile Web service migration framework in Kazzaz and Rychlý (2017). The framework is proposed to enable service provisioning and migration in Mobile SOA by providing system adaptation to context changes of the mobile device resources. The provided case study presents the reusability of the demonstrated migration framework by adopting a traffic jam detection scenario and extending system ontology and decision-making process introduced in (Kazzaz & Rychlý, 2015).

This work is motivated by the traffic jam scenarios presented in (Riva, Nadeem, Borcea, & Iftode, 2013; Weyns, Malek, & Andersson, 2010) where the migration framework is installed on a group of cooperative cars. A car *A*, can plan its route from point *X* to point *Y* and investigate a traffic jam possibility on this route. The traffic jam investigation is performed through migrating Tra*fficJamSearch se*rvice of car *A* and running it on another car *B* (i.e., a new service provider) located in the area of interest (AOI) defined by car *A*. By calling the migrated service, car *A* will acquire the required information in order to plan a better route by avoiding traffic jams.

In this example, there are two criteria governing the service migration decision making process:

1. Speed*Criteria: rep*resents the speed difference between a subject car *A* and the destination car *B*, and
2. Center*DistanceCriteria: repr*esents the distance of destination car *B* from the AOI's center of car *A*.

The SpeedCr*iteria promote*s the migration selection to destination car with the speed closest to the speed of *A*. While the CenterDi*stanceCriteria promote*s the selection of service TrafficJ*amSearch migrati*on to the car closest to the center of the AOI of car *A*.

When a car *B* is chosen as a new destination for the TrafficJa*mSearch service,* the migration controller on *A* starts the physical migration process to *B*. Then, *A* executes a search process on *B* by calling TrafficJamSea*rch to discover* the number of existed cars in *A*'s AOI in order to detect a traffic jam on its planned route. This work adopts the Internet of Things (IoT) approach through depending on a migrated service that is locally hosted on a mobile device instead of relying on an external cloud service.

The rest of this paper is organized as follows. Section 2 discusses the related work on context-awareness and related implementation on traffic jam detection. Section 3 presents the ontology-based context model provided to describe services and vehicles properties and preferences. Section 4 demonstrates the migration framework for mobile service migration between vehicles. Section 5 provides a detailed description of the framework implementation. Section 6 provides description of the experiment performed to test the context-aware mobile Web service migration approach through cooperative vehicles scenario. Finally, the researchers present the work conclusion in Section 7.

2. RELATED WORK

This section presents the related work utilizing context awareness and self-adaptation to solve traffic jam problems. It discusses the differences between these works and this one.

In (Feld, & Müller, 2011), the authors demonstrated an automotive ontology-based vehicle and user models to support knowledge sharing between cars and to allow system adaptation and recommendation based on user's preferences.

In the work of (Hu, Li, Ngai, Leung, & Kruchten, 2014) context-awareness has been proposed to enable the usage of several resources of contextual data such as user's personal activities, social data and environment context (i.e., location, temperature). The authors defined an ontology to describe a mobile smart city system in a crowdsensing scenario. Context-awareness was implemented through context monitoring and matching of the collected context data and providing system recommendation to the user.

The authors of (Autili, Cortellessa, Di Benedetto, & Inverardi, 2015) provided a framework for adaptive context-aware mobile services. They defined a Service Level Specification (SLS) notion to describe extra-functional information to be used in selecting the proper service during the adaptation. The adaptation process is presented by rebuilding service source code based on user preferences. However, the adaptation process is still limited by the need to statically define the adaptable classes and their alternatives.

The authors in (Deng et al., 2017) propose a peer-to-peer architecture for mobile Web service selection and composition. The proposed architecture composer is responsible for discovering the services hosted on nearby mobile devices and composing the required service to respond to a mobile user service request. However, their proposed algorithm uses only service response time factor to select the best services to involve in the composition. In the contrast, this work's framework allows to use a dynamic set of services' and devices' properties and preferences in order to find a set of possible destinations. Moreover, the work authors utilize the Analytic Hierarchy Process (AHP) (Saaty, 1990) decision making algorithm with a dynamic set of criteria to determine the best migration to perform. However, this work proposes service migration as an adaptation so that the service can be migrated and hosted on the requesting device, not only to be used while the requester is close to the service origin device.

Bauza and Gozálvez, 2013 provided a fuzzy-logic based system to detect road traffic congestion using Vehicle-to-Vehicle (V2V) communications. The system implements a mechanism of two stages to detect the congestion. First, it receives information messages published by surrounding cars to locally estimate the possible congestion. Second, if a congestion is sensed, the system shares and utilizes other estimations of surrounding cars to make more accurate estimation of the congestion.

Another cooperative aware vehicle communication system is proposed in the work of Santa, Pereñíguez, Moragón, and Skarmeta, (2014) to provide information about traffic status and events. A Cooperative Awareness Message (CAM) and Decentralized Environmental Notification Message (DENM) are proposed to describe exchanged messages between cars stating their current status and position. While CAM is used for status notification in one-hop communication, DENM messages are broadcasted over multi-hop communication to cover a specific geographic area. A car hosts services that allow the system to retrieve its position and status to be used in traffic tracking and monitoring applications. Compared to their work, the contribution of this work provides a generic context aware adaptive system that can be customized and utilized in different scenarios including the traffic monitoring system.

In order to use the mobile Web service migration framework introduced in (Kazzaz & Rychlý, 2017) in a traffic jam detection scenario, we customized and extended the ontology provided in (Kazzaz & Rychlý, 2015) with new traffic jam domain-specific classes to describe system components models, properties and related criteria governing the AHP-based decision-making algorithm proposed to select the best migration to perform. On the other hand, we improved the decision-making algorithm with new weighting approach during the decision-making process. For example, when weighting the speed properties based on the speed criterion, a car with speed closer (both higher or lower) to the source car speed should have higher weight and priority to be selected from between all other possible destination cars.

3. SYSTEM COMPONENTS CONTEXT REPRESENTATION

System components, status, attributes, and preferences are contextual information that must be formally described and understood in order to enable system adaptation and context awareness. For this purpose, an ontology is utilized to semantically describe system components and their contextual information. As discussed in Section 2, the researchers customized the Web service migration ontology proposed in (Kazzaz & Rychlý, 2015) to describe a service migration between devices located on cars in a cooperative vehicle scenario. The ontology enables the implementation of context reasoner (for instance, Jena reasoner) to generate new information (facts) about system devices (cars) based on the collected context. The new derived facts can trigger a context-aware process and lead the service migration-based adaptation process. In Figure 1, a graphical presentation demonstrates the proposed ontology-based model supporting service migration in cooperative cars scenario. The model describes system components of services and service providers. A Service is either a Fr*ameworkService o*r a Mi*gratableService.* A Fr*ameworkService i*s a type of service that is hosted to enable the adaptation process, the communications between system components and publishing the context model of its hosting service provider. On the other hand, a Mi*gratableService i*s the type of service that can be migrated from their current hosting device to another service provider. A service provider SP *c*an be defined as a Ca*ndidateDestinationServiceProvider f*or a given Mi*gratableService S i*f SP *s*atisfies the requirements of S. Similarly, new context information can be generated by the reasoning process of the context model. For example, a Mi*gratableService S w*ill be noted as a Ca*ndidateForMigrationService i*f its current service provider SP *i*s no longer capable to host it (i.e., S). Hence, a new information will be assigned to SP *a*s an instance of Ca*ndidateOriginServiceProvider t*ype.

Component properties are presented in the ontology through instances of the P*roperty* class. The utilized example defines the following properties: (1) S*peed,* (2) C*enterDistance,* (3) S*ervicePriority,* and (4) C*riteriaProperty.* A C*riteriaProperty* is defined as the criterion that is evaluated in the decision-making process to choose the most suitable service provider to host the migrated service. Two criteria are considered in this example, (1) S*peedCriteria,* and (2) C*enterDistanceCriteria.* The S*peedCriteria* is the criterion that values the migration to the car that has the closest speed to the speed of the source car. While C*enterDistanceCriteria* values the migration to the car that is closest to the center of the AOI identified by the source car.

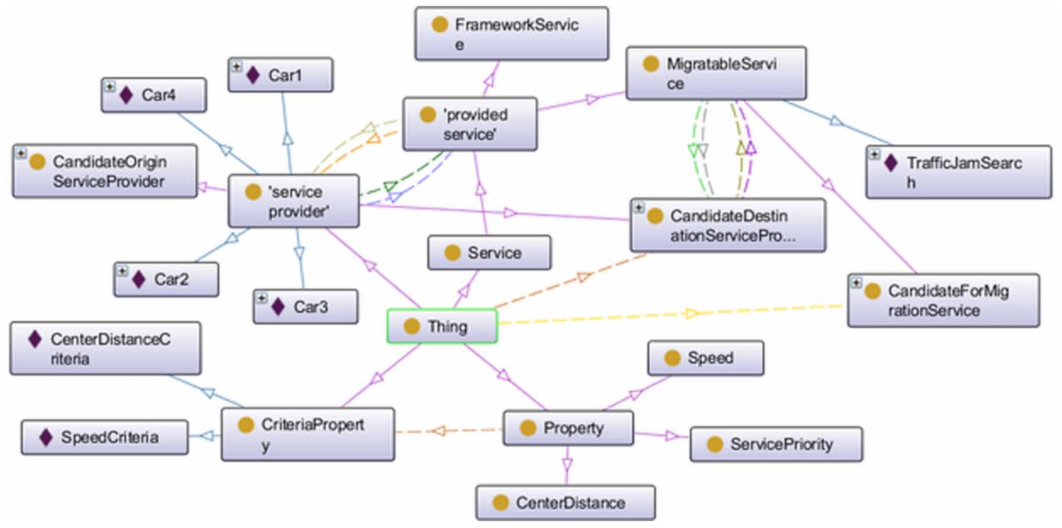
On the other hand, the component context model states the functional and non-functional preferences and rules using Jena framework (Jena Apache, 2017). A Jena rule is described as a set of Resource Description Framework (RDF) (World Wide Web Consortium, 2014) triples that can derive new OWL[1]/RDF entries in system model. These rules can be reasoned using Apache Jena reasoner to generate new context entries. Through a continuous context-aware monitoring of system context, when a violation of the rules is sensed, the system launches an adaptation process of service migration to a new service provider. In the example scenario, the violation is caused by the traffic information service's absence that rises the need to migrate a *TrafficJamSearch* service to a neighboring car in order to search and discover the number of surrounding cars so that it can estimate traffic status and predict traffic jams in an AOI.

4. MOBILE MIGRATION FRAMEWORK ARCHITECTURE

In this section the authors demonstrate the framework architecture supporting service migration in a cooperative car scenario. For this purpose, the authors consider the migration for one service or more to be temporally migrated/published on a new destination car (i.e., service provider) that exists in an AOI identified by the source car. Based on that, the framework's *Controller* of the source car will lead both the context-aware and adaptation processes. On the first hand, the context-aware process is presented in the system through the following functionalities:

1. Discovering cars located in the AOI of source car.
2. Checking the destination service provider availability.
3. Monitoring the quality of service after migration and deciding if another migration is required.

Figure 1. System model ontology-based representation



On the other hand, system adaptation is presented through the ability of system to migrate the service to a new car and retrieve the necessary information from the service after the migration. The provided framework supports stateless service migration and a migration authorization process based on real-time assessment of destination car rule defined in its context model (for example, the rule allows service migrations only from cars with the same car manufacture of the destination car).

Figure 2 demonstrates the proposed framework architecture. The architecture consists of the following modules.

4.1. Discovery Module

The Discovery Module is responsible for a car discovery process and retrieve a list of surrounding cars located in the AOI of the source car.

4.2. System Context Manager Module

This module is responsible for generating, monitoring and reasoning system context periodically to enable system context awareness. The module creates the system context model containing all retrieved partial context models of discovered service providers and the *MigratableService* model intended for migration. It is also responsible for generating the partial models of system components that define a real-time status of their properties and state preference rules of the subject *MigratableService* and the surrounding cars.

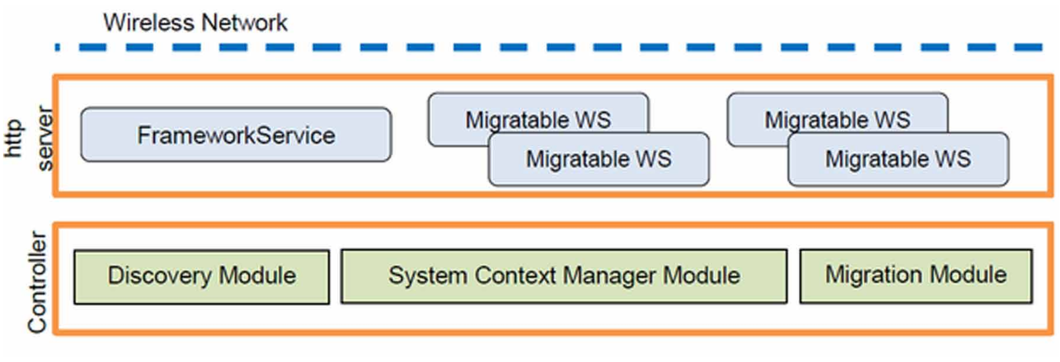
After creating the system context model of the discovered service providers, the *Controller* looks for a destination car that can host *MigratableService* where the pre-defined rules of both *MigratableService* and the destination car can be satisfied.

This process is performed through utilization of an ontology reasoning process of system context model so that new RDF triples of service provider instances noted as *possibleDestinationProvider*-s for the *MigratableService* will be created in the model. The output of this unit is a list of triple entries stating the *MigratableService*, the source car, and the possible destination car.

4.3. Migration Module

This unit is responsible for selecting the best migration to perform from the input set of possible migrations provided by the System Context Manager Module. It is also responsible for the physical

Figure 2. Mobile Web service migration framework architecture



transferring and deploying of the service package on a destination car service provider. The migration selection is provided through a multi-criteria decision-making process using the AHP decision making method. A detailed description of the decision-making process is noted in the previous work of Kazzaz and Rychlý (2015).

5. FRAMEWORK IMPLEMENTATION DESCRIPTION

The framework implementation is provided through two parts, the grounding framework service and the Android mobile application. The framework service is a Restful-based web service implemented using the Restlet framework (Louvel, Templier, & Boileau, 2012). It is responsible for publishing the car/service provider instance in the network in order to be discovered by other cars. The authors utilize the light-weight stack WS4D-JMEDS (Zeeb, Moritz, Timmermann, & Golatowski, 2010) designed to support service and device discovery in ad-hoc network. The discovery process is enabled by creating and starting JMEDS device instance on the mobile device. An example URI request to perform a discovery process on destination car is:

```
http://{IP}:{Port}/FrameworkService/discover/centerLng/
{centerLng}/centerLat/{centerLat}
```

The service provider has the functionalities to retrieve car's context model, GPS location, speed (provided by the Google location service on Android device), and the position of its AOI. Moreover, it provides the functionalities required for the physical migration and installation of service's WAR packages on the new host device. The context model of a *MigratableService* is integrated in its WADL file while the context model of each service provider can be retrieved by calling the *FrameworkService* method *getProviderContext* through the following URI:

```
http://{IP}:{Port}/FrameworkService/getProviderContext/json
```

The service migration Android application is responsible for the following tasks:

1. Discovery process of cars located in a defined area of interest.
2. Generating system context model by adding the context models of discovered cars.
3. System context model reasoning and migration suggestion process.
4. Performing the migration decision making process.
5. Publishing *TrafficJamSearch* service on the selected destination car service provider.
6. Calling *TrafficJamSearch* service to retrieve the information of traffic status in the defined area.

The context model of a car is retrieved in JSON format by calling the *getProviderContext* method of its framework service. Based on its IP, a car is identified in the network, and its context model can be retrieved in order to create system context model as explained in Section 4.2.

When car *A* launches the traffic jam detection process over a certain area on its route, it starts searching for a car in that area to host its T*rafficJamSearch* service. When a car *X* is discovered, the source car *A*'s framework application requests the location and speed information of car *X* to determine whether the discovered car *X* exists in its AOI or not based on the distance between the car *X* location and car *A*'s AOI center. Only cars located in that area will be considered in the migration process so that the framework will request its context model to include their properties and preferences in the model reasoning process.

The framework application calculates the distance of each discovered car from the center of the AOI of car *A* and adds it to the system model as a CenterD*istance proper*ty of the related car. The calculation of CenterD*istance uses t*he precise location of discovered car and the AOI center location of the source car. Similarly, the most recent speed values of discovered cars are added to the system model as Speed p*roper*ties. The authors choose to consider average speed (estimated during the last 60 seconds) and precise location values to 1) keep the decision making more reliable and realistic and 2) to avoid system failure of service migration to a car with an outdated location.

Finally, the AHP algorithm starts to weight the migrations based on two criteria: (1) Cente*rDistanceCriteria; and* (2) Speed*Criteria, so* that the migration with the destination car closer to the center of the AOI will have a higher weight to be chosen by the decision-making process. Similarly, the car with a speed that is equal or around *A*'s speed will be more highly chosen as a destination car. Finally, the decision-making process chooses the best destination with the highest composite weight calculated based on the aforementioned criteria.

The physical migration of the subject Migra*tableService is p*rovided through calling the Frame*workService meth*ods that enable sending and deploying the Migra*tableService WAR* package on the matching destination car. The Contr*oller call*s the following URI to perform the migration of Migra*tableService's im*plementation from source to destination:

```
http://{destination.IP}:{destination.Port}/FrameworkService/
download/{source.IP}/port/{source.Port}/temp/{source.TempFolder}/
service/{MigratableServiceWAR}
```

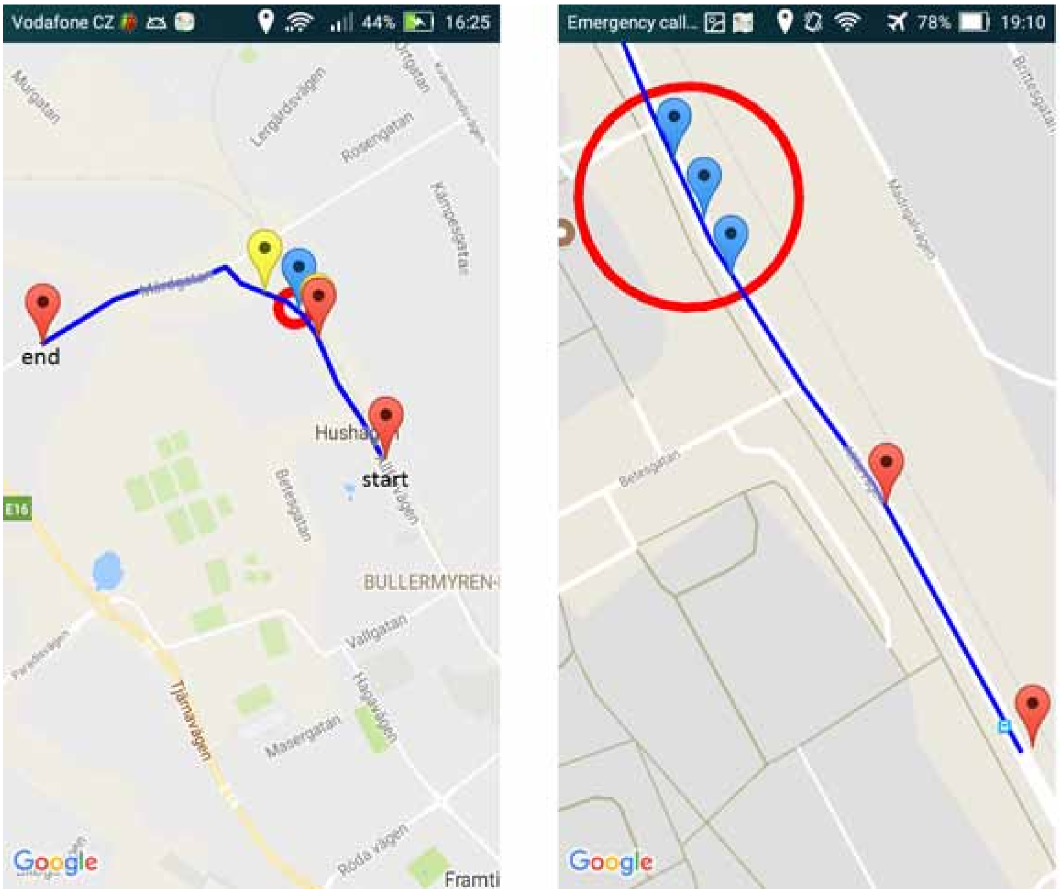
6. EXPERIMENT AND RESULTS

This section presents a scenario of traffic jam detection service migration from one car to another in order to perform a car discovery process and avoid traffic jams over a specific route. In this experiment, the authors use 3 Ge*nymotion A*ndroid emulators and one real mobile device as service providers. Each device represents a car on a planned route of a single lane where cars go from point A to point B. Speeds of the emulator devices are mocked to have random values between 10 to 40 km/h while a fixed speed is set to 20 km/h for the source car. To present the possibility of setting rules for migration process, the authors set a rule for Car3 to not accept services with a priority less than 70%. The subject migratable service, Tr*afficJamSearch, h*as a priority property of 50%.

The researchers initialize the locations of these cars with 100 meters distance between them consequently. Figure 3 demonstrates the interface visualizing the route and the AOI of the subject car, Car1, is marked as a red circle. Each car is presented in a blue marker when located inside the AOI of Car1.

On Car1, the framework starts searching for other cars located in its AOI looking for a suitable destination car to host its search service so that it can call *TrafficJamSearch* service on its new location and can get feedback about the status of traffic in that area. The area of interest, is set to be 200 meters ahead from Car1 with a diameter of 100 meters.

Figure 3. Application interface showing the planned route and surrounding cars during the experiment example



At first the framework on Car1, starts searching for neighboring cars. When a car is found, The Car1 framework requests the *FrameworkService* hosted on the discovered car to get its current location and check whether the discovered car is located inside Car1's AOI to consider in the migration process or not. After having a list of discovered cars, Car1 framework starts to create system context model of the discovered cars and the subject *TrafficJamSearch* context models.

In Figure 5, the discovered cars are presented with their distances from the center of Car1 AOI. To demonstrate the decision-making process, the authors choose the situation when the three cars are located inside Car1's AOI. Table 1 contains the *Speed* and *CenterDistance* properties of the cars during the example's migration decision. At this moment, the framework suggests two possible migration of

Table 1. Cars properties during migration example

Car Name	CenterDistance (m)	Speed (km/h)
Car1	Not applicable	20
Car2	41.54	18
Car3	17.5	30
Car4	12.34	24

the subject service. *Mig A* and *Mig B, Mig A* suggests the migration of the *TrafficJamSearch* to *Car2* while *Mig B* suggests the migration to Car4. Even though Car3 is located inside the specified area but it is not chosen as a destination to host the service regarding to Car3's preference that allows it to host only services with priorities higher that 70% while the *TrafficJamSearch* has a priority of 50%.

The next step for the framework to perform is to decide which one of this migration is best to execute. This is decided based on the AHP process using the defined criteria priorities and the related properties of the migration destinations. To enable the AHP decision making method, the defined criteria priorities must ∈ {*1, 3, 5, 7, 9*} which gives weight to a criterion respectively from the lowest priority of 1 to the highest priority of 9. For this experiment, the priority of *SpeedCriteria* and *CenterDistanceCriteria* is set to 9 and 3 respectively.

In respect of the criteria proiorties, the criteria comparison matrix *A* is initiated using the AHP-based I*nitializeCriteriaMatrix* algorithm, (see Figure 4), introduced in the previous work (Kazzaz & Rychlý, 2015). *A* is a square matrix of real numbers with a dimensions $m \times m$, where m is the number of considered criteria. Each *A*'s entry states the weight of the considered criterion to other criteria based on their priorities. To set *A* entries, the Initial*izeCriteriaMatrix algori*thm performs pair-wise evaluation between each two criteria and maps the difference between their Criteri*aPriority values* to a value ∈ {1, 3, 5, *7, 9} or it*s reciprocal. The criteria comparison matrix generated by Initial*izeCriteriaMatrix is dem*onstrated in Table 2. According to the defined criteria priorities, the criteria comparison matrix entries show that service migration to a car D1 with a speed closer to the speed of the source car is 7 times important than service migration to a car D2 located closer to the interest area center of the source car than car D1.

Similarly, the AHP algorithm initiates for each criterion a pair-wise migration weight comparison matrix with dimensions $n \times n$, where n is the number of the possible migrations, based on the migration properties related to that criterion. The CenterD*istance proper*ty is governed by the CenterD*istanceCriteria. The Ce*nterD*istanceCriteria and Ce*nterD*istance are pr*oportional so that the CenterD*istance proper*ty will have the highest weight of 50 when equals to 0 meter (i.e., located in the middle of AOI) while it will have the lowest weight when it equals to 50 meters (i.e., located on the boundary of AOI). On the other hand, Speed p*roper*ty is governed by the SpeedCr*iteria which* is dynamically calculated by the algorithm based on the current speeds of cars existed in the AOI. For each migration the algorithm queries the system context model for the current cars speeds. Later, the algorithm evaluates the weights of each car's Speed p*roper*ty based on the speed of the source car so that the car that has speed slightly different and closer to the speed of source car will have higher weight to be chosen as a destination car. On the contrary, the car with a speed that significantly differs from the source car's speed will have lower weight and will be less likely to be chosen as a destination car based on the SpeedCr*iteria factor*. The migration comparison matrices are provided in Table 3.

Finally, the AHP algorithm computes the composite weight vector p through Equation 1 and the migration related to the highest value from between p entries will be chosen and executed.

$$p = V \bullet W \qquad (1)$$

where V is the $n \times m$ *matrix* of priority vectors of migration comparison matrices and W is the weight vector of matrix A. The co*m*puted weights of the possible migrations are noted in Equation 2.

$$p = \begin{pmatrix} 0.677 \\ 0.323 \end{pmatrix} \qquad (2)$$

where p_{11} and p_{21} entries represent the weights of Mig *A and* Mig *B res*pectively. Based on the highest value of composite weight vector p, Mig *A wi*ll be performed as it has the highest priority ($p_{11} = 0.677$).

This experiment is repeated 10 times in order to measure the time spent to perform the migration process and its sub processes. The experiment shows that the average time for the whole migration process is 7.5 Sec while the average time to create system context model and perform the ontology inferencing process is 4.136 Sec and the time to make the decision using the AHP algorithm is 0.251

Figure 4. The *InitializeCriteriaMatrix* algorithm to compute a pair-wise criteria comparison matrix for AHP based on *Criteria* priorities of individual criteria

```
Require: $\langle p_1, p_2, \ldots, p_m \rangle$ as values of CriteriaPriority of criteria $\langle c_1, c_2, \ldots, c_m \rangle$
Ensure: $A$ is a pair-wise criteria comparison matrix for given criteria $\langle c_1, c_2, \ldots, c_m \rangle$
 1: for $i \leftarrow 1$ to $m$ do
 2:      $a_{ii} \leftarrow 1$
 3:      for $j \leftarrow i+1$ to $m$ do
 4:           $difference \leftarrow |p_i - p_j|$
 5:           if $difference \geq 8$ then
 6:                $judgment \leftarrow 9$
 7:           else
 8:                $judgment \leftarrow 2\left\lfloor \frac{difference}{2} \right\rfloor + 1$
 9:           end if
10:           if $pi > pj$ then
11:                $a_{ij} \leftarrow judgment$
12:           else
13:                $a_{ij} \leftarrow judgment^{-1}$
14:           end if
15:           $a_{ji} \leftarrow a_{ij}{}^{-1}$
16:      end for
17: end for
18: return $A$
```

Sec. Based on the experiment settings, the authors see that the required migration time is acceptable as the source car, moving with a speed of 20 km/h, will cross almost 40 meters only while performing the migration. This leaves the car 160 meters away from its AOI which means it will have enough distance for the routing system to call the *TrafficJamSearch* service and re-plan the route if necessary.

The result shows the validity of the proposed approach to solve a problem of traffic information service absence in a real-time application through a seamless context-aware service migration adaptation.

7. CONCLUSION

The authors designed and implemented a framework for *TrafficJamSearch* service migration between cars to support traffic jam detection over a specified area. The experiment result demonstrates the usability of the implemented framework supporting service mobility between mobile devices.

Table 2. Main criteria comparison matrix and its priority vector

$\lambda m_{ax = 2}$, $CI = 0$; $CR = 0$			
	CenterDistanceCriteria	**SpeedCriteria**	**Priority vector - *W***
CenterDistanceCriteria	1.0	0.14	0.125
SpeedCriteria	7.0	1.0	0.875

Table 3. Migrations comparison matrices and priority vectors

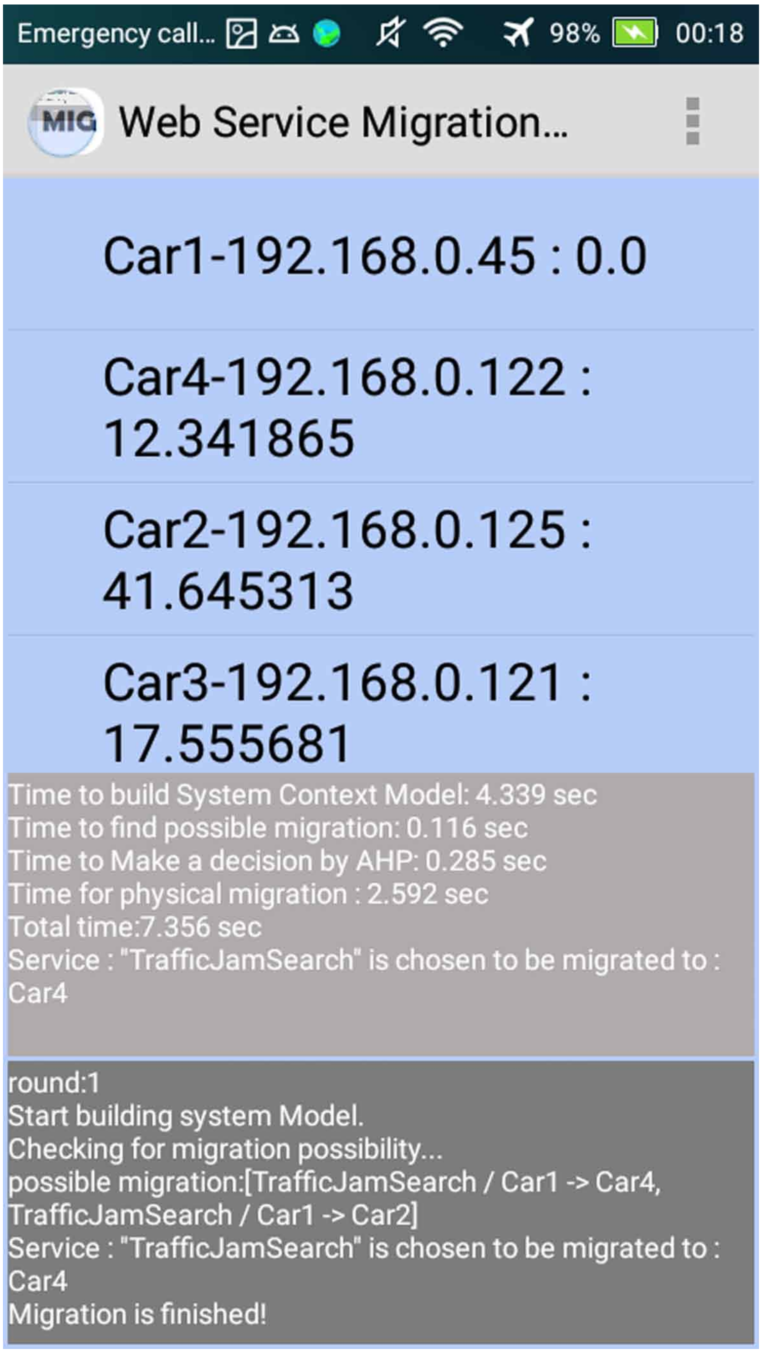
Migration Comparison Matrix with regard to *CenterDistanceCriteria*			
$\text{»}_{max} = 2$, $CI = 0, CR = 0$			
	Mig A	*Mig B*	Priority vector – $V^{(1)}$
Mig A	1.0	0.2	0.166
Mig B	5.0	1.0	0.833
Migration Comparison Matrix with regard to *SpeedCriteria*			
$\text{»}_{max} = 2$, $CI = 0, CR = 0$			
	Mig A	*Mig B*	Priority vector – $V^{(2)}$
Mig A	1.0	3.0	0.75
Mig B	0.33	1.0	0.25

Moreover, it presents the applicability of the proposed service migration adaptation approach to survive service absence during route planning as real-life scenario.

For future work, the researchers plan to improve the proposed adaptation approach due to the highly volatile system environment and migration criteria by performing multiple migrations to all discovered cars with post-evaluation of those migrations. Which means that the subject service will be migrated directly to any discovered car where the matching and reasoning process will be performed locally on the discovered car. This delegation of this reasoning process to the discovered car will help to improve system performance. Thus, the source car can start using a migrated service and later decide which one and only of its migrated instances is the optimal choice to keep and utilize while all other service instances will be removed. Another considered improvement is to optimize the speed estimator and the defined AOI's diameter through new experiments for more reliable experience.

Acknowledgements: This work was supported by The Ministry of Education, Youth and Sports of the Czech Republic from the National Programme of Sustainability (NPU II) – project IT4Innovations excellence in science LQ1602 and BUT internal grant "ICT tools, methods and technologies for smart cities" FIT-S-17-3964.

Figure 5. Mobile Web service migration framework application



```
Car1-192.168.0.45 : 0.0
Car4-192.168.0.122 : 12.341865
Car2-192.168.0.125 : 41.645313
Car3-192.168.0.121 : 17.555681
```

```
Time to build System Context Model: 4.339 sec
Time to find possible migration: 0.116 sec
Time to Make a decision by AHP: 0.285 sec
Time for physical migration : 2.592 sec
Total time:7.356 sec
Service : "TrafficJamSearch" is chosen to be migrated to :
Car4
```

```
round:1
Start building system Model.
Checking for migration possibility...
possible migration:[TrafficJamSearch / Car1 -> Car4,
TrafficJamSearch / Car1 -> Car2]
Service : "TrafficJamSearch" is chosen to be migrated to :
Car4
Migration is finished!
```

REFERENCES

Abowd, G. D., Dey, A. K., Brown, P. J., Davies, N., Smith, M., & Steggles, P. (1999, September). Towards a better understanding of context and context-awareness. In *International Symposium on Handheld and Ubiquitous Computing* (pp. 304-307). Berlin: Springer. doi:10.1007/3-540-48157-5_29

Alkhabbas, F., Spalazzese, R., & Davidsson, P. (2017, April). Architecting emergent configurations in the internet of things. In *2017 IEEE International Conference on Software Architecture (ICSA)* (pp. 221-224). IEEE. doi:10.1109/ICSA.2017.37

AlShahwan, F., Faisal, M., Karaata, M. H., & AlShamrani, M. (2015). RESTful-Based Bi-level Distribution Framework for Context-Based Mobile Web Service Provision. *JSW*, *10*(3), 260–287. doi:10.17706/jsw.10.3.260-287

Autili, M., Cortellessa, V., Di Benedetto, P., & Inverardi, P. (2015). On the adaptation of context-aware services. arXiv:1504.07558

Bauza, R., & Gozálvez, J. (2013). Traffic congestion detection in large-scale scenarios using vehicle-to-vehicle communications. *Journal of Network and Computer Applications*, *36*(5), 1295–1307. doi:10.1016/j.jnca.2012.02.007

Deng, S., Huang, L., Taheri, J., Yin, J., Zhou, M., & Zomaya, A. Y. (2017). Mobility-aware service composition in mobile communities. *IEEE Transactions on Systems, Man, and Cybernetics. Systems*, *47*(3), 555–568. doi:10.1109/TSMC.2016.2521736

Erl, T. (2005). *Service-oriented architecture: concepts, technology, and design*. Pearson Education India.

Feld, M., & Müller, C. (2011, November). The automotive ontology: managing knowledge inside the vehicle and sharing it between cars. In *Proceedings of the 3rd International Conference on Automotive User Interfaces and Interactive Vehicular Applications* (pp. 79-86). ACM. doi:10.1145/2381416.2381429

Garlan, D., Cheng, S. W., Huang, A. C., Schmerl, B., & Steenkiste, P. (2004). Rainbow: Architecture-based self-adaptation with reusable infrastructure. *Computer*, *37*(10), 46–54. doi:10.1109/MC.2004.175

Hoang, D. B., & Chen, L. (2010, December). Mobile cloud for assistive healthcare (MoCAsH). In *Services Computing Conference (APSCC), 2010 IEEE Asia-Pacific* (pp. 325-332). IEEE.

Hu, X., Li, X., Ngai, E., Leung, V., & Kruchten, P. (2014). Multidimensional context-aware social network architecture for mobile crowdsensing. *IEEE Communications Magazine*, *52*(6), 78–87. doi:10.1109/MCOM.2014.6829948

Jena, A Semantic Web Framework for Java, URL: http://jena.apache.org/

Kazzaz, M. M., & Rychlý, M. (2015, June). Web Service Migration using the Analytic Hierarchy Process. In *2015 IEEE International Conference on Mobile Services (MS)* (pp. 423-430). IEEE. doi:10.1109/MobServ.2015.64

Kazzaz, M. M., & Rychlý, M. (2017, June). Restful-based Mobile Web Service Migration Framework. In *2017 IEEE International Conference on AI & Mobile Services (AIMS)* (pp. 70-75). IEEE. doi:10.1109/AIMS.2017.18

Louvel, J., Templier, T., & Boileau, T. (2012). *Restlet in action: developing restful web apis in Java*. Manning Publications Co.

Papakos, P., Capra, L., & Rosenblum, D. S. (2010, November). Volare: context-aware adaptive cloud service discovery for mobile systems. In *Proceedings of the 9th International Workshop on Adaptive and Reflective Middleware* (pp. 32-38). ACM. doi:10.1145/1891701.1891706

Paspallis, N., Rouvoy, R., Barone, P., Papadopoulos, G. A., Eliassen, F., & Mamelli, A. (2008, November). A pluggable and reconfigurable architecture for a context-aware enabling middleware system. In *OTM Confederated International Conferences" On the Move to Meaningful Internet Systems"* (pp. 553-570). Berlin:Springer. doi:10.1007/978-3-540-88871-0_40

Riva, O., Nadeem, T., Borcea, C., & Iftode, L. (2007). Context-aware migratory services in ad hoc networks. *IEEE Transactions on Mobile Computing*, *6*(12), 1313–1328. doi:10.1109/TMC.2007.1053

Saaty, T. L. (1990). *Decision making for leaders: the analytic hierarchy process for decisions in a complex world*. RWS publications.

Santa, J., Pereñíguez, F., Moragón, A., & Skarmeta, A. F. (2014). Experimental evaluation of CAM and DENM messaging services in vehicular communications. *Transportation Research Part C, Emerging Technologies*, *46*, 98–120. doi:10.1016/j.trc.2014.05.006

Wagh, K., & Thool, R. (2014). Mobile Web Service Provisioning and Performance Evaluation of Mobile Host. *International Journal on Web Service Computing*, *5*(2), 1–10. doi:10.5121/ijwsc.2014.5101

Weyns, D., Malek, S., & Andersson, J. (2010, May). On decentralized self-adaptation: lessons from the trenches and challenges for the future. In *Proceedings of the 2010 ICSE Workshop on Software Engineering for Adaptive and Self-Managing Systems* (pp. 84-93). ACM. doi:10.1145/1808984.1808994

World Wide Web Consortium. (2014). RDF 1.1 Concepts and Abstract Syntax. Retrieved from http://www.w3.org/TR/rdf11-concepts/

Zeeb, E., Moritz, G., Timmermann, D., & Golatowski, F. (2010, September). Ws4d: Toolkits for networked embedded systems based on the devices profile for web services. In *2010 39th International Conference on Parallel Processing Workshops (ICPPW)* (pp. 1-8). IEEE.

Zuo, Y., & Liu, J. (2015, April). Mobile agent-based service migration. In *2015 12th International Conference on Information Technology-New Generations (ITNG)* (pp. 8-13). IEEE. doi:10.1109/ITNG.2015.7

ENDNOTES

1 https://www.w3.org/OWL/

M. Mohanned Kazzaz is currently a Ph.D. student in the Department of Information Systems at Brno University of Technology, Faculty of Information Technology (BUT FIT). He received the B.Eng. degree in Computer Engineering from the University of Aleppo, Syria, in 2007. His research interests are in the area of Software Engineering and more specifically in development of self-adaptive software architectures. Kazzaz published five conference papers and one article related to his dissertation topic.

Marek Rychlý is an assistant professor at Brno University of Technology, Faculty of Information Technology. His research interests include component-based and service-oriented architectures, formal description of software architectures and their evolution, functional and quality-driven automatic web services composition and testing, big data and distributed software systems. He authored more than 25 papers in reviewed scientific journals and conference proceedings.