Proceedings of the Federated Conference on
Computer Science and Information Systems pp. 1185–1192

ISBN 978-83-60810-51-4

Context Aware Services for Mobile Users: JQMobile vs Flash Builder Implementations

Alfio Costanzo, Alberto Faro, Daniela Giordano, Concetto Spampinato
Department of Electrical, Electronics and Computer Engineering,
University of Catania, viale A. Doria 6,
Catania, 95125, Italy

***Abstract*—Current information systems that help user mobility, i.e., mobility information systems, generally lack of the basic context information that support people activity in time-changing environments: they do not take into account current traffic, weather or car pollution conditions, nor the mobile users are timely informed about accidents, or road works. Also, such systems do not consider the user personal information that influences the user context awareness such as age, or health status. Aim of the paper is to illustrate how context aware services may be offered by an implementation architecture based on a server following the Model-View-Controller paradigm, i.e., Ruby on Rails (RoR), whose controllers implement the use stories of the mobile users, and whose views are JQMobile scripts that provide for each story the most suitable scenario by an user interface based on the familiar Google Maps. How Flash Builder applications resident on the user mobiles may provide the users with similar RoR views, but saving RoR time, is also widely discussed. Furthermore the paper claims that involving actively the mobiles into the mobility information system may support more effectively the context aware decisions of the mobile users.**

I. Introduction

CURRENT information systems that help user mobility, i.e., mobility information systems, generally lack of the basic context information that may help people in environments where the state is time and location dependent: they do not take into account current traffic, weather or pollution conditions, nor the mobile users are timely informed about accidents, strikers or road works. Also, such systems do not consider user personal information and activities that influence their context awareness such as age, health status, time limits to decide, or most suitable services for their task.

For these reasons, several *functional architectures*, e.g., [1], have been proposed to merge in some intelligent system both external and personal information to support user decisions. However, functional architectures are too abstract in location and context aware applications where it is important not only *what* information is offered by the system to the users, but also *how* the system is implemented. Indeed, the aim of the location and context aware applications is to help the users in taking the most appropriate decisions to support the activity to accomplish a task.

Consequently, not only the *user interface* but also the *system internal structure* should be analyzed carefully in all the cases in which the information system is conceived to improve the user location and context awareness, e.g., in all the time-varying scenarios where the user activity is highly depending on the user decisions and where interference due to not relevant information, e.g., information dealing with other scenarios, should be reduced to a minimum.

This assumption is partially at the basis of the User Centered System Design (UCSD) [2]. Indeed, in UCSD system internal structure is not analyzed specifically, whereas the main effort is dedicated to design human-computer interfaces that allow the users to *afford* the needed information, according to the well known Gibson's theory on the human perceptual processes [3]. Criticisms to the little attention given by the UCSD approach to the internal structure has been widely advanced in the past by the authors claiming that, according to the *situated cognition* paradigm [4], [5] also the system internal structure should be designed suitably to support adequately the user activities. In particular, it has been suggested to structure the software per *use stories* [6] to support specifications, verification, testing and reuse [7].

The same point of view has been embraced in the last years by the design paradigm called Model-View-Controller (MVC) [8], where the use stories are implemented by specific *controllers*, the related interfaces by *views*, whereas data are *modeled* as objects although they are physically organized in MySQL tables or XML files.

Aim of the paper is to illustrate how location and context aware services [9], [10], may be offered by an *implementation architecture* based on a server following the MVC paradigm, i.e., Ruby on Rails (RoR) [11], whose controllers implement the use stories, and whose views are JQMobile scripts [12] that implement for each story the most suitable scenario by an user interface based on the familiar Google Maps. Also, how Flash Builder by Adobe[13] may be used to implement applications resident on the user mobiles for providing the users with similar RoR views, but saving RoR time, is discussed.

Transition from a centralized urban database that is at the core of a central data warehouse for the mobile users of a city to a distributed data warehouse organization consisting of datasets resident on separate servers is also outlined in the paper so that privacy, updating policies and management may be favored by a proprietary solution running on the proper machine, whereas data integration is obtained by transforming each dataset in an XML or RDF archive [14].

This transition will lead, as envisaged in the paper, to rethink the role of the user devices from more or less passive visualization systems to active nodes of a new generation of mobility information systems, where we claim that the user mobiles powered by suitable programming environments like

978-83-60810-51-4/$25.00 © 2012 IEEE

Flash Builder, may host mobile data and rules dealing with the user status to be linked to the ones dealing with conditions external to the users resident on public databases This will give rise to information systems that support more effectively the context aware decisions of the mobile users.

Sect.2 proposes two implementation architectures to offer the sought support to location and context aware decisions of the mobile users: one based on an RoR server provided with JQMobile user interface and the other on Flash Builder applications resident on the mobiles. Sect.3 shows how the urban road graph and the current position of the user and destination in such graph (needed by the location aware systems) may be obtained by using Google Maps APIs. Sect.3 points out that both the proposed implementations outperform the current GPS navigation systems since they offer real time information and make feasible mobile applications that support the user tasks such as m-commerce (e.g., buying on the fly), m-government (e.g., everywhere digital certification), and m-bureaucracy (e.g., document management in mobility). Also, this section illustrates how an effective decision support system for mobile users that takes into account external conditions and personal data may be obtained by the joint work of the RoR server and the Flash Builder application resident on the mobile. Finally, in sect. V and sect. VI we compare the JQMobile and Flash Builder solutions and point out future works.

II. Real Time City Information Systems

Fig.1 shows how the traffic information needed to compute the travel time of each road segment is obtained in our system dividing the city into zones controlled by computing nodes that collect the measurements on traffic taken by in situ technologies and from perceptions issued by authorized people, as proposed in [15].

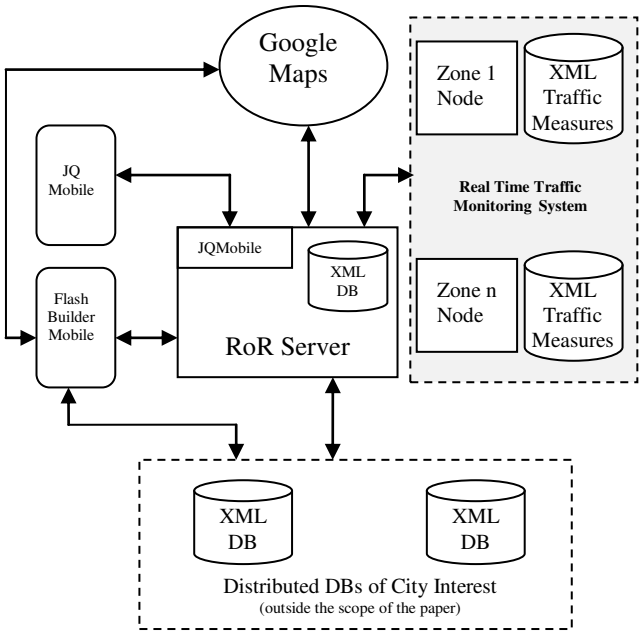

Figure 1. Traffic information system based on real time information: current frontiers

Such nodes should compute the travel times of each road segment of their zone by using suitable formulas depending on the car flow conditions and on the type of road intersection, e.g., the ones pointed out in [16]. We envisage that the travel times will be stored in XML files resident at the zone nodes, where they are updated at regular time intervals or when traffic conditions change suddenly for dangerous events, e.g., accidents or storms. These XML data are collected by a central server that is able to compute the best path for any source-destination pair.

In this architecture, walking and driving users may ask the RoR server, through their mobiles, to suggest the best path to a destination chosen by them or to a destination chosen by the system to best fit the user needs. The display of the path on the mobile is obtained by a JQMobile application resident on the server that uses the relevant Google Maps APIs or by a specialized software, based on Flash Builder, resident on the user mobile.

The former solution has the advantage of informing the users even when they are provided with simple mobiles, but it increases the processing load of the server, especially if it has to behave as a Decision Support System (DSS) to help the mobile user activities. The latter solution relieves the server from the task of displaying the geo-referenced information on the maps; also, it may evolve into a more general software that implements on the mobile, in collaboration with the central server, a sort of city mobility personal assistant that takes into account the current traffic and weather conditions, and the user profile, as will be illustrated in sect.IV.

Let us note that the development of the mentioned personal assistant on the mobile avoids that the server has to create the view that displays on the mobile the map and the graphical frames (i.e., buttons, text boxes, divisions, etc.); but, this is not easy since it requires that the mobile is provided with a software to allow the users to issue their queries and to receive the system responses by a graphical interface comparable with the one obtainable with JQMobile. Also, the mobile should have enough memory and processing speed. Fortunately, the latter requisite may be fulfilled by using the modern cellular phones or PDAs (e.g., iphones, android phones, iPADs, etc.), whereas the former requisite may be satisfied by developing the mobile application using Flash Builder. Consequently, both the solutions drawn in fig.1, as well as both the mentioned implementations are currently feasible.

III. Locating the Users and their Destination in the Urban Road Graph

To compute the minimum path from a source to a destination of a traffic network is necessary:

- to represent the road graph of the traffic network by a mathematical description such as the following array:

$$road\_segment(i_a, i_b, d_{ab}, t_{ab}) \qquad (1)$$

where i_a, and i_b, are road intersections whereas d_{ab} and t_{ab} are the distance and the travel time associated to the road segment from intersection i_a to intersection i_b.

- to identify in what road_segment the users are located, and in what position of the segment, when they issued to the server the request to know the minimum path to the destination. The user position may be identified by adding the following two rows to the road array:

$$\text{road\_segment}(\text{user}, i_b, d_{ub,}\, t_{ub})$$
$$\text{road\_segment}(i_a, \text{user}, d_{au,}\, t_{au}) \quad (2)$$

- to identify in what road_segment the destination is located and in what position. This may be obtained by adding other two rows of the road array:

$$\text{road\_segment}(\text{dest}, i_b, d_{db,}\, t_{db})$$
$$\text{road\_segment}(i_a, \text{dest}, d_{ad,}\, t_{ad}) \quad (3)$$

Fig.2 shows how the road graph should be modified to compute the minimum path. The nodes dealing with the user and destination positions may be deleted after having computed the minimum path, whereas the ones related to the road intersections are fixed points of the traffic network.

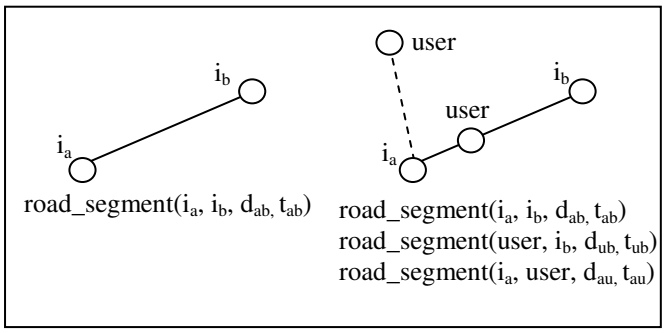

Figure 2. Inserting in the urban road graph, drawn on the left, the current user position at the right distance from the intersections (as shown on the right). If the user segment does not belong to the graph, then a novel segment is added to the road graph (see the dotted segment on the right). For simplicity, the road_segments are two ways streets.

In principle the geo-coordinates of the intersections may be obtained by adopting suitable procedures in the most diffused CAD environments, e.g., Arcinfo [17] or Autocad Map [18], but using Google Maps and related APIs has the advantage of providing the designers with the same programming interface in both building the cartography and in visualizing the information on the user interface.

For this reason, we assume that the road graph consists of a set of geo-referenced intersections derived from Google Maps that can be superimposed to the Google Maps thus making possible to inform timely and in a familiar Google Maps interface the users about changes of the road status (e.g., works in progress, blocked by an accident, or closed for excessive pollution), as well as to take into account the current traffic and weather conditions.

In the Google Maps terminology, such intersections are points located at the crossroad of two or more roads that have the property of being connected by the get_distance API with a path of only one step to the adjacent intersections. These intersections can be found by our novel function, called get_intersections(first_road, second_road) obtained by using the API get_directions with the walking option to go from the first_road to the second_road. Indeed, the sought intersection is the first point of the path, if any, whose second_road is indicated in its info window.

This function has been put at the core of an automated procedure to find the intersections of any city called get_all_intersections(from_road, to_road) that, using the names of the city roads stored in an XML file, computes the intersections of each road in the interval [from_road, to_road] with all the other roads of the city by applying the above mentioned function get_intersections. In our implementation, the intersections identified by the procedure are stored either into an MySQL table and XML file. Since each intersection between two roads is identified by calling the Google Maps API get_directions, we have based our implementation on a listening process that waits 1 second for the response to the query sent to get_directions stored on the Google Maps site.

To save processing time the roads of the city have been subdivided into segments belonging to only one neighborhood. Thus, for example for a medium city, as our city, with 2000 roads and 20 neighborhoods we need about 10.000 seconds to find all the intersections of a neighborhood. This time can be further decreased if one executes get_all_intersections by a cloud of computing nodes working in parallel. In our case, by a cloud of 10 computing nodes we obtained the intersections of a neighborhood in about 15 minutes that is quite acceptable because get_all_intersections is a batch procedure to be executed at a very low rate (e.g., one time per month).

In the implemented procedure the user may choose the city and the road interval [from_road, to_road]. The intersections are stored only if they are featured by a pair (latitude, longitude) different from (0, 0). Of course, an intersection is not stored if it has the same geo-coordinates of a previous stored intersection. This allows us to store only one time the intersection among several roads (see fig.3).

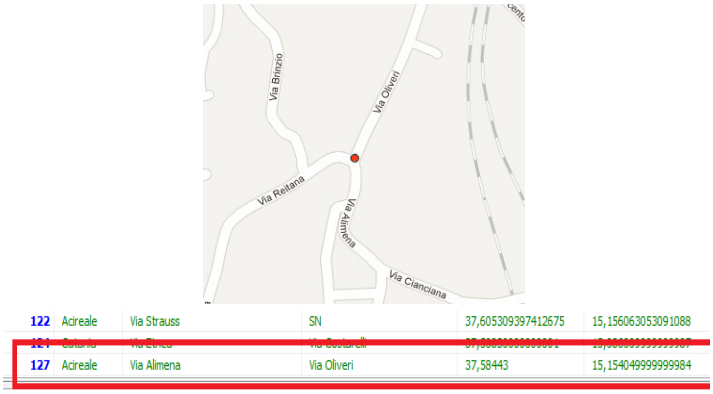

Figure 3. Intersection among more than two roads is identified by only one record on the MySQL intersection table and consequently represented by only one red point on the relevant Google Maps.

In this way we may obtain incrementally all the road intersections of a city that are displayed by small red circles super-imposed on the relevant Google Maps, as shown in fig.4. The user should check the final result to add manually the few intersections that cannot be identified by the mentioned procedure, e.g., since they are related to roads that are not labeled by names.

Figure 4. Intersections of a city obtained by the get_all_intersections function proposed in the paper.

Obtaining the array road_segment from the array intersection(id, first_road, second_road, latitude, longitude) is straightforward, since two intersections are linked by a segment if they are connected by get_directions in only one step. Tracking the position of the users in a road graph visualized on the mobile screen is very easy if one uses the GPS of the adopted navigation system, but describing mathematically their position according to (2) is slightly more complicated. Indeed we have to find the address nearest to the GPS position and then the two addresses associated to the intersections of the same road that are nearest to the current user address. Fig.5 shows how this can be obtained very fast by a Flash Builder mobile without the intervention of the server. Analogously we should operate to find the two road_segment of the destination indicated in (3).

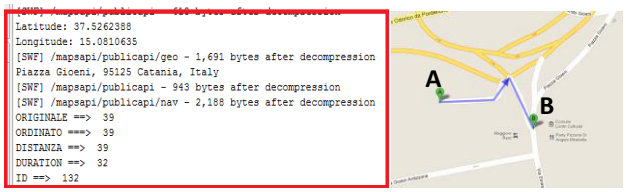

Figure 5. Inserting the user position in A increases the road graph with two segments, but if A is not in a road of the graph the user is localized by a new segment (in blue) that links A to the nearest intersection B (on the right). This modification is found very fast by the program implemented on the Flash Builder (see its operations on the left).

Of course, in the JQMobile version the four road_segment are computed by the RoR server, whereas in the Flash Builder implementation this is done by the mobile that sends such segments together with the identifiers of the origin and the destination to the server where a minimum path program will find the shortest path,, e.g., [19]. In our implementation, this program has been written in Prolog [20] consisting of facts and rules since we have found that it has the same time performance of the traditional program written in C and allows us to solve different problems by simply modifying the facts [15].

For example, with a simple modification of the facts, it is easy to find the minimum path for other location services, e.g., a) to find the nearest park and b) the minimum path to reach the park nearest to the destination.

In the former case, all the city parks are linked to a node, let say parking_node, with a road_segment having distance and travel time equal to zero, as shown in fig.7. The minimum path to go from the user position to the parking_node gives the minimum path to reach the nearest park, being the nearest park the one located at the penultimate node. The same approach may be used to find the nearest hotel, pharmacy and so on.

In the latter case, first, we have to find the park at the minimum distance from the destination following the same approach described above, i.e., finding the minimum path, let say $path_1$, from the parking_node to the destination and then we have to find the minimum path from the current position to the park located at the second node of the previous path $path_1$. These steps are also illustrated in fig.6.

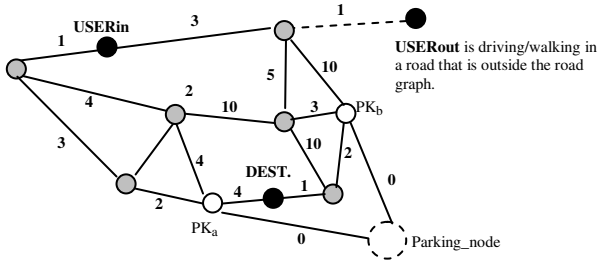

Figure 6. Typically, users are driving/walking in the road graph, e.g., USERin, but if the user is not within such graph, e.g., USERout, a new segment is added to the road graph. The nearest park is the penultimate node of the minimum path to reach the Parking_node, i.e., park PK_a. The park nearest to the destination is given by the second node of the minimum path connecting Parking_node to the destination, i.e., park PK_b.

IV. From Location Aware to Context Aware City Services

In principle, all the mobility information services that take into account the user location and current traffic may be viewed as location aware services, such as the services outlined in the previous section, where the system takes into account the current position of the users and the current travel time featuring the roads to find the best path to destination. Context aware services have a wider scope, since they should take into account also the user personal information and the current user activity. Of course, it is difficult to provide services for each specific user activity, but structuring the city services as use stories certainly may favor the transition from location to context aware systems.

Indeed parking reservation may be viewed as a typical use story where both location and context issues have to be taken into account specifically since they differ from the ones to be taken into account to support the users to enact another story, e.g., to find a pharmacy. For this reason, the paradigm MVC suggests that all the actions and their order in time to be carried out for obtaining a specific service may be conveniently organized in a software module called controller. According to MVC, each action of an use story is supported by a proper view displayed on the user device, including the initial action to start the service. For this reason the main interface of our location/context aware city service provider consists of icons related to the first actions of the stories the are relevant for the current user activities.

For example, fig.7 shows the main screen of an user interested in carrying out bureaucratic practices at institutions located in different places and who has to buy some specific drugs; in this case the user may need

information on pharmacy, parking, minimum path to destination, refueling and costs, the documents that are needed for the relevant practice and the ones that may be obtained automatically from the system, and where the various institutions are located. Let us note that in the current implementation it is the user who chooses from the list of use stories the ones that are more relevant for the task. In future this will be done with the help of an intelligent agent.

Figure 7. Main screen of an user interested in carrying out bureaucratic practices at institutions located in different places and who has to buy some specific drugs.

The location part of the pharms and parks has been treated in the previous section. Fig.8 shows the display of our Flash Builder navigator, running on an android mobile, obtained when the user is searching for a given service, e.g., a pharmacy. On the left, the mobile shows the current position computed on the basis of the mobile GPS and offers two possibilities to reach the destination: the first according to Google Maps and the other according to the minimum path computed by the Prolog program. On the right, we show the response computed by Flash Builder after the user choice.

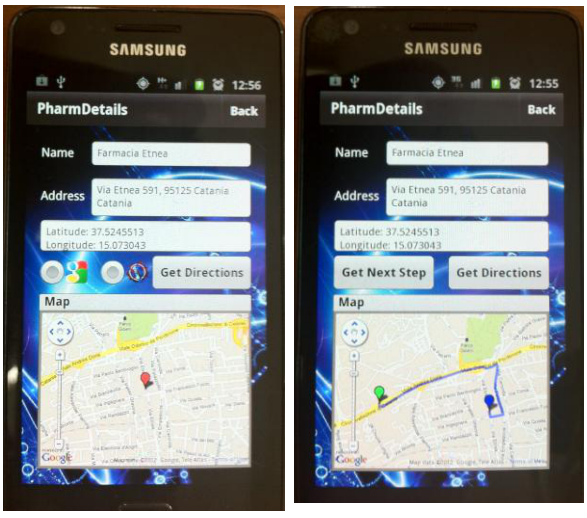

Figure 8. Flash Builder version of the navigator. It shows the current user position computed on the basis of the mobile GPS and offers two possibilities to reach the destination (on the left). On the right the response computed by Flash Builder after the user selection.

A further step towards context awareness may be obtained by offering to the users the services that are most suitable for their activity. For example in fig.9 the user has increased the list of parks normally taken into account (i.e., the ones managed in our city by the company AST) with a set of parks offered in our city by the company AMT that are close to the public services so that they may be considered when the user is searching for a park near to the today destination. Let us note that the Flash Builder application does not need to ask for the park list to the server since it is able to go inside any publicly available XML databases and to extract the relevant information. In our implementation only the network addresses and the general descriptions of the databases are collected into an XML file stored on the server so that the Flash Builder mobile may be addressed to relevant DBs, where it may find the relevant information.

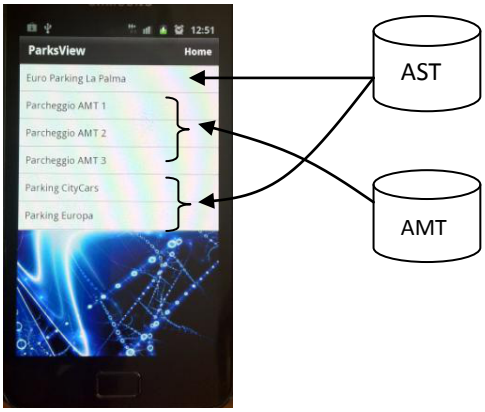

Figure 9. The mobile collects from different sites the names of the parks of possible interest for the today activity, i.e. the ones offered in the downtown by public company named AST and the ones offered within public offices by another company named AMT.

Another step that may help the mentioned bureaucratic user activities is the possibility to have in mobility certificates and templates relevant for the task. For example our Flash Builder application may access the Municipality DB transformed into XML files to receive typical demographic certificates. In particular, the data of such certificates are taken from the Municipality DB and passed to a PHP procedure resident on the RoR server to produce the certificate in pdf that will be sent by the server to the user e-mail.

To take into account the personal data and other context conditions we may apply Fuzzy rules [21] in the following form: if condition 1and .... and condition N are true then the service should be reached in a short time.

As known, in fuzzy logic the truth degree of the conclusion is given by its membership degree to the fuzzy set associated to the conclusion, i.e., in our case the fuzzy set short_time given by the minimum truth degree of the antecedents. After having computed the truth degree of the conclusion, it is easy to find the maximum distance or time to destination that satisfies the rule by simply defuzzyfing the fuzzy set of the conclusion, i.e., passing from the truth degree (on the y-axis) to the value of distance or time (on the x-axis) that qualifies the fuzzy set.

Thus, assuming that the fuzzy set short_time is the one represented in fig.10, we have that if the membership degree of the antecedent is ≈ 1, then the time to reach the service should be between 0 and 3 minutes, let say 1,5 minute, whereas if the degree $\approx$ zero, then the time to reach the service may be greater, let say 2,5 minutes.

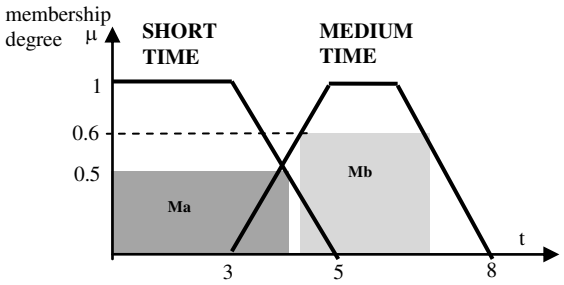

Figure 10. Fuzzy set associated to a position that can be reached in a short or medium time.

If we have to meet more than one rule, we have to compute the truth of each antecedent. As an example, if the fuzzy rules are as follows:

- if the weather is bad, then the service should be reached in a short time,
- if the current time is greater than 9 p.m. then the service could be reached in a medium time
- if the user is an middle age people, then the service could be reached in a medium time
- if the health status is not good the should be reached in a short time.

we have to compute the membership degrees of each antecedent by using the relevant fuzzy sets, i.e., bad weather, nighttime, middle age and bad health status, drawn in fig.11. Therefore, assuming that:

a) from some meteorological service on the web (e.g., yahoo) we have that currently there is a medium rain,
b) the current time is 7,30 pm,
c) the user has forty-five years, and
d) a body temperature of 37,5 °C,

it follows that the maximum acceptable time to reach the destination is given by the $2\,M_a + 5{,}5\,M_b / (M_a + M_b)$, i.e., by the barycentre of the two masses in fig.12: M_a obtained by intersecting the fuzzy set short_time with the line at $\mu = 0{,}5$ (that is the minimum membership of the antecedents dealing with short_time), and M_b obtained by intersecting the fuzzy set medium_time with the line at $\mu = 0{,}6$ (that is the minimum membership of the antecedents dealing with medium_time).

The services that are within the maximum time (or distance) are then displayed to the users by a set of icons enclosed within a circle (see fig.12). The users may inspect the various services to decide their preferred service. Then they may know the best path to the destination by a view similar to the one displayed in fig.8.

Of course, in case the users are interested in the distance of the services since they are walking, the circle gives to them the better information to decide, whereas in case they are driving the circle may induce the users to choose the services nearest to their current position.

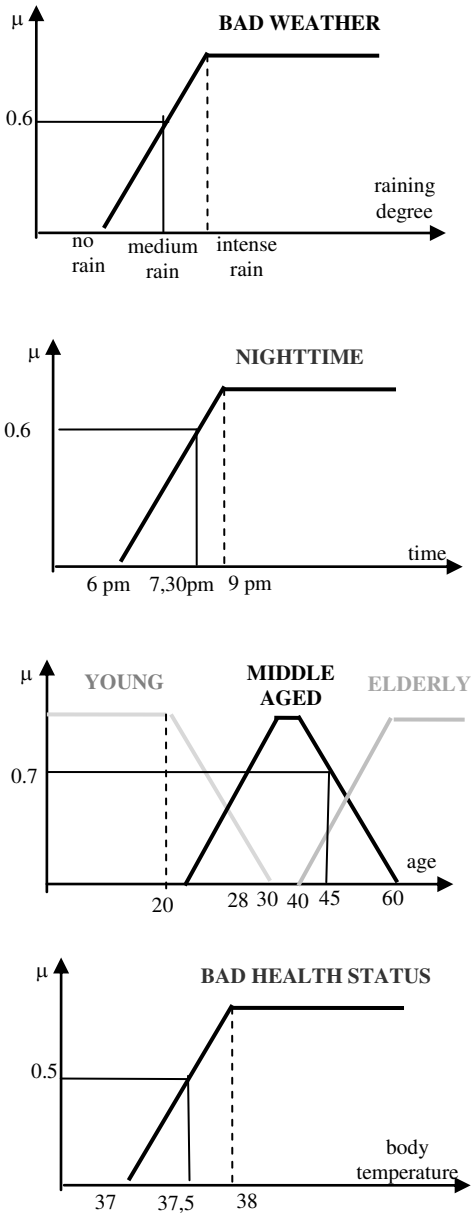

Figure 11. Fuzzy sets associated to bad weather, nighttime, middle age, and bad health status.

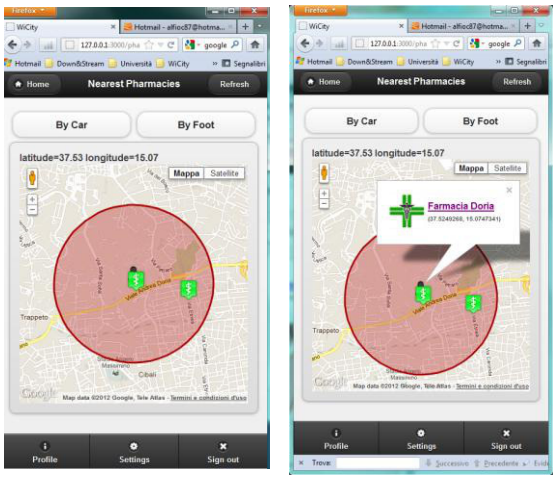

Figure 12. The services pointed out within the circle are the ones that can be reached in an acceptable time (or are within an acceptable distance from the current user position).

V. Comparing RoR-Flash Builder and RoR-JQMobile Navigation Systems

The location based services outlined in the paper, including the proposed extensions dealing with the location and context awareness functionalities, are available on both PCs and mobiles: one version works in either PCs or mobiles and makes use of RoR and JQMobile; the other is for mobiles and is based on RoR and Flash Builder. In both the versions, the server manages the registration process so that the users may pass from the JQMobile to the Flash Builder implementations and viceversa while maintaining their profile. The server is also involved in both versions in finding the basic arrays concerning the traffic, i.e., the arrays intersection and road_segment, and in updating the travel times for each road segment at a regular time interval.

However, the two developed versions show relevant differences. Indeed, in the JQMobile version, all the functions are carried out by the RoR server with an interface that, due to the JQMobile features, has the same format in both PC and cellular phone. The position may be obtained by either a GPS connected to the PC or by the geo-coordinates of the Access Point to which the PC or the mobile is connected. The latter way of localizing the user is not compatible with the precision requested by our method to localize the user position in the road graph, but the former method needs to implement a specific procedure that communicates the GPS coordinates to the server. Also, the server has the responsibility to search from relevant databases the information that better meets the user needs, and the one of managing the certification needed by the mobile users. Moreover, in the JQMobile version the server should display the best path passing to the API get directions the relevant intersections computed by the Prolog program that get directions has to consider as mandatory waypoints.

The main part of the mentioned work of the RoR server may be avoided if one uses the Flash Builder approach. For example, the mobile may find alone the acceptable maximum time or distance from the current position by using the mentioned fuzzy rules and may find also the list of service points of interest by visiting the relevant databases without the server intervention, as well as it cooperates with the server to manage the certification phase. Furthermore, although the server computes the minimum path, it does not build the map for the user mobile. Indeed, it stores the path in an XML file that will be read by the user mobile that will display the path on the relevant Google Map by using the get directions API, possibly enriching the screen with signals useful for the user mobility as shown in fig.13.

Finally, the Flash Builder version allows us to manage the GPS of the mobile, and to compute autonomously the four facts (i.e., road_segment) the are needed to localize source and destination in the road graph. The distances between the above source/destination intersections and the adjacent ones can be derived by get_directions, whereas the travel time should be estimated by using the formulas described in [15].

Currently, in both versions the minimum path is computed by the Prolog program executed on the server since Flash Builder does not allow us to run executable codes with the traditional command system, whereas this is possible in the Ruby code. In future we plan to write this program in Flash Builder, thus limiting at maximum the computing load of the server.

Figure 13. Minimum path computed by the Prolog program and displayed on the user mobile as a set of mandatory waypoints of the Google Maps API get_directions.

VI. Concluding Remarks

The paper has illustrated how transition from traditional GPS navigation systems to navigators implemented on smart phones envisaged in many market studies, e.g. [22], became a reality. Two versions have been presented that are the metaphors of two smart phone navigators: one, i.e., JQMobile navigator, that is mainly supported by a mobility server, the other, i.e., the Flash Builder navigator, that works in a relatively autonomous way. Both the navigators are displayed by a familiar Google Maps interface.

If one is provided with a powerful mobile, the Flash Builder version is better then the JQMobile version since it saves the RoR server time. Also, the minimum path program could be implemented in the Flash Builder navigator at condition of regularly updating the travel time of each road segment. This operation is not time consuming if one considers that in a medium city we have about 20000 road segments and that, consequently, the updating implies the download of a small file of about 20KB.

Both location and context awareness services to improve a mobility information system have been addressed. This has illustrated how this new generation of navigators may support the user decisions taking into account real time external conditions and personal constraints. Moreover, not only the traditional supports to mobility/logistics and m-commerce but also to bureaucratic tasks are provided with the smart phones that may change the way the people works. For example, a workflow similar to the one shown in fig.11 is followed by the Flash Builder version to help the user in filling the declarations in lieu of a notarial act needed to ensure the truth of a report.

Currently, the presented proposals are under test at our city in a project, called K-Metropolis, supported by Regional Funds. However, the use of urban ontologies [23], [24] to represent objects and rules used in the navigational system by a standard RDF notation will allow us to extend without any change the approach to other cities.

Future scenarios, sketched in fig.14, are also under study where the personal assistant resident on the user mobile is able to read the XML file stored on the zone nodes

containing the travel time of each road segment and to carry out locally the computation of the best source-destination path. This evolution seems feasible since the Flash Builder based software is able to consult remote XML files and thanks to the fact that the number of remote XML files dealing with the traffic of a zone is limited. Indeed, the zone number is equal to the number of city neighborhoods that is generally limited to some tens (e.g., the New York neighborhoods are about 50 and the ones of a medium city are about 10).

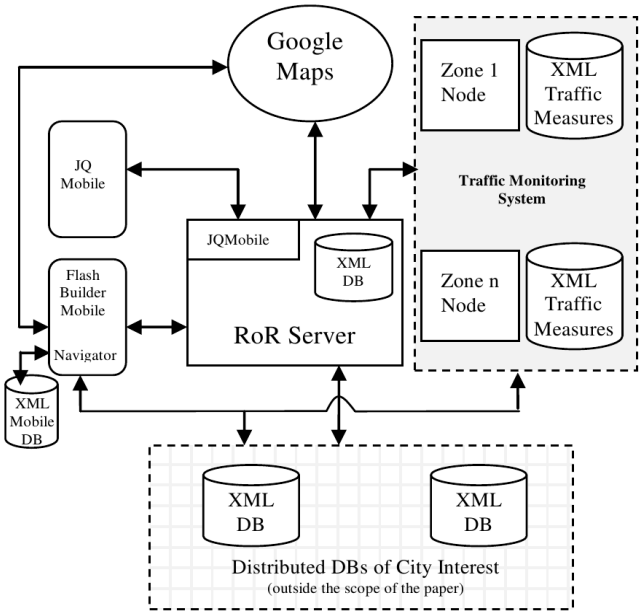

Figure 14. Traffic information system based on real time information: future directions

Finally let us note that in the paper we have not addressed the role of the monitoring system of the traffic parameters, i.e., car density, speed and flow, that are needed to compute the travel time for each road since this is outside the scope of the paper. However, it is important to point out that expensive monitoring systems could limit the feasibility of the navigators proposed in the paper. Fortunately, the studies and the applications available in the literature, e.g., [15], [25], [26], show that effective monitoring systems may be obtained by not expensive webcams, complemented by traditional low cost techniques [27]. Consequently, this makes feasible the real time services at the core of the location and context aware city services proposed in the paper.

REFERENCES

[1] Bettini C., Maggiorini D., Riboni D., Distributed Context Monitoring for Continuous Mobile Services, MOBILE INFORMATION SYSTEMS II, IFIP, Vol.191, 123-137, 2005

[2] Endsley M.R., Jones D.G., Designing for situation awareness: an approach to user centered design, CRC, 2009

[3] Goldstein E. B., The Ecology of J. J. Gibson's Perception, Leonardo, Vol. 14(3), 191-195 , MIT Press, 1981

[4] Elsbach K. D., Barr P.S., Hargadon A.B., Identifying Situated Cognition in Organizations, Organization Science Vol. 16(4), 422–433, 2005.

[5] Faro, A, Giordano, D., Ontology, esthetics and creativity at the crossroads in Information System design, Knowledge Based Systems , Volume 13 (7), 515-525, Elsevier, 2000

[6] Faro, A., Giordano, D. StoryNet : an Evolving Network of Cases to Learn Information Systems Design. IEE Proceedings SOFTWARE, 119-127, 1998

[7] Faro, A., Giordano, D., Concept Formation from Design Cases: Why Reusing Experience and Why Not. Knowledge Based Systems Journal, vol. 11, n.7/8, p.437-448, 1998

[8] Dubberly Design Office, The Model-View-Controller Pattern in a Rails-Based Web Server, http://www.dubberly.com/wp-content/ uploads / 2011/04/DDO_Article _MVC_ Pattern . pdf, 2011

[9] Surhone L.M. et al. Real-Time Locating, Betascript Publishing, 2010

[10] Mller P.F. et al., Context awareness, Alphascript Publishing, 2010

[11] Hartl M., Ruby on Rails 3, Addison Wesley, 2011

[12] David M., Developing Websites with jQueryMobile, Focal Press, 2011

[13] Corlan M., Flash Platform Tooling: Flash Builder, Adobe, 2009

[14] Powers,S. Practical RDF, O'Reilly Media, 2003

[15] Faro, A., Giordano, D. and Spampinato, C., Integrating Location Tracking, Traffic Monitoring and Semantics in a Layered ITS Architecture. *Intelligent Transport Systems,* vol.5(3), 197-206, 2011

[16] Faro, A., Giordano, D. and Spampinato, C.: Evaluation of the Traffic Parameters in a Metropolitan Area by Fusing Visual Perceptions and CNN Processing of Webcam Images, IEEE Transactions on Neural Networks, Vol. 19 (6), 1108-1129, 2008.

[17] Ormsby T., et alii: Getting to Know ArcGIS Desktop, ESRI, 2008

[18] Aubin P.F., Gregg Stanley G., AutoCad Map 2011, Autodesk, 2010

[19] Zhan, F. B., Noon, C. E., Shortest Path Algorithms: An Evaluation Using Real Road Networks, Transportation Science, Vol.32(1), 65–73, 2008

[20] Bratko, I., PROLOG Programming for Artificial Intelligence, Addison-Wesley Educational Publishers Inc, 2011

[21] Wang P.P., 2001. Computing with words. Wiley Inderscience, 2001

[22] Telematics Reserach Group (TRG), Preview portable navigation: the future is bright for connectivity, http://www.telematicsresearch. com/press_data/pdfs/PortableNavigation.pdf, 2010

[23] Zhai, J., Jiang, J., Yu, Y. and Li, J.: Ontology-based Integrated Information Platform for Digital City, *IEEE Proc. of Wireless Communications, Networking and Mobile Comp.,* WiCOM '08, 2008.

[24] Faro, A., Giordano D., Musarra, A.: Ontology Based Mobility Information Systems. Proc. of IEEE Systems, Men and Cybernetics SMC03, vol.3, 4288-4293, 2003.

[25] Faro, A., Giordano, D. and Spampinato, C., Adaptive background modeling integrated with luminosity sensors and occlusion processing for reliable vehicle detection. IEEE Transactions on Intelligent Transportation Systems, Vol.12(4), 1398 - 1411, 2011

[26] Crisafi A., Giordano D., Spampinato C., GRIPLAB 1.0: Grid Image Processing Laboratory for Distributed Machine Vision Applications. Proc. Int. Workshop on Enabling Technologies: Infrastructure for Collaborative Enterprises, WETICE '08. 188-191, IEEE, 2008

[27] Leduc, G., Road Traffic Data: Collection Methods and Applications, Working Papers on Energy, Transport and Climate Change, N.1, JRC European Commission, 47967, 2008