Ontology Based Integration and Decision Support in the Insigma Route Planning Subsystem

Piotr Szwed*, Piotr Kadłuczka*, Wojciech Chmiel*, Andrzej Glowacz* and Joanna Sliwa†
*AGH University of Science and Technology
E-mail: {pszwed,pkad,wch,aglowacz}@agh.edu.pl
†Military Communication Institute
E-mail: j.sliwa@wil.waw.pl

***Abstract*—The route planning subsystem is an important component of the Intelligent System for Global Monitoring Detection and Identification of Threats (INSIGMA). Its goal is to calculate an optimal route taking into consideration contextual information and values of dynamically updated parameters describing the current traffic. Developing the system we have taken an approach consisting in providing a set of simpler route planning algorithms that can be used in various situations instead a single all purpose procedure. A key issue encountered during the system development was the correct choice and configuration of algorithm to be used. The selection depends on such factors, as: user profile and preferences, dynamically collected traffic data and historical records. In the developed system the knowledge about these factors, their relations and rules is gathered in ontologies. The paper presents the system architecture and an execution scenario, in which the decision on selection and configuration of one of the several implemented route planning algorithms is based on semantic information and build in rules.**

***Index Terms*—ontology, dynamic route planning, personalization**

I. Introduction

The objective of the INSIGMA project is to develop and implement heterogeneous information system for complex detection, identification of threats, monitoring and identification of mobile objects. The project is to propose innovatory solutions in 5 areas related to:

- Monitoring and identification of vehicles and people,
- Dynamic monitoring and identification of threats within the traffic,
- Analysis of traffic and optimizing routes for external users,
- Identification of suspicious people and threats,
- Discovery of data and multimedia with the watermarking technology.

The Route Planning Subsystem is one of the key components within the INSIGMA system. Its goal is to calculate an optimal route between two locations taking into consideration contextual information and values of dynamically updated parameters describing the current traffic. These parameters originate from various types of integrated detectors: microwave and inferred lasers, ultrasonic sensors, inductive loops, camera equipped video image processors, acoustic arrays, mobile GPS trackers, and others.

Work has been co-financed by the European Regional Developement Fund under the Innovative Economy Operational Programme, INSIGMA project no.POIG.01.01.02- 00-062/09.

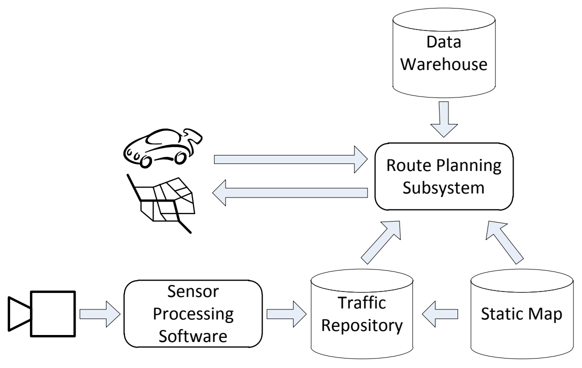

Fig. 1. Main components of the route planning subsystem

Main components of the INSIGMA system relevant to the Route Planning Subsystem are presented in Fig. 1. The subsystem uses the information about a road network stored in the Static Map, current values of traffic monitoring parameters from the Traffic Repository and historical data stored in the Data Warehouse. The figure refers to one of the system operation mode, where optimal route calculation are done at the server side.

During the system development there were elicited several requirements regarding its architecture and various quality attributes. They became (sometimes conflicting) drivers for making particular architectural decisions. Below we list the most important of them:

- An ontology will be used to define all monitoring parameters stored in the Traffic Repository to enable information sharing and integration with sensors and clients (modifiability);
- The Traffic Repository must meet assumed performance requirements, what can be an indication to use a relational database;
- The Route Planning Subsystem will perform dynamic and personalized planning, what favours using semantic techniques to describe user profiles, preferences and contextual information (customizability).

- From project management perspective the dynamic route planning combined with personalization is attributed with a high risk, as it is still a novel and challenging technology, to mitigate this risk the system will provide a framework for integrating multiple route planning algorithms and making decisions which of them should be used for a particular route planning task.

The main idea presented in this paper is the following: instead of developing a single and monolithic route planning algorithm whose behaviour should be adapted to fit all possible conditions, it is proposed to implement a set of simpler algorithms. Then, for a particular task instance one of them will be selected and configured taking into account various attributes characterizing road network, traffic and user preferences. Such approach is consistent with the growth scenario for the system: new algorithms can be added during the development and maintenance phases without degrading the existing functionality. Moreover, simpler algorithms can be easier to develop and work on them can be conducted concurrently by independent teams.

To adopt successfully such approach, two conditions must be fulfilled:

1) the system must have a highly modular system architecture providing appropriate separation of concerns accompanied by a suitably chosen integration technique;
2) there must be implemented a framework enabling proper selection and configuration of algorithms.

In was decided that in both cases the the ontologies and Semantic Web tools will be used as to support the integration and decision making. It should be stressed, that collecting data from sensors, accessing them, as well as realization of the route planning tasks must satisfy real-time performance requirements. Due to this, the selection of an algorithm should be made within relatively short time with use of a small and efficient set of rules. The decision supporting system uses ontological representation of knowledge about task properties, task parameters and algorithm capabilities to select most promising optimization procedure and properly configure it.

The rest of the paper is organized as follows. The next section II presents works related to the route planning. In the section III the logical structure of maps used within the INSIGMA route planning subsystem is presented. Section IV discusses algorithms used for route planning and their data model. Section V briefly presents ontologies developed to support the system services. The route planning subsystem architecture and basic use case scenario is described in the section VI. The next section VII discusses the rules for algorithm selection. Concluding remarks are presented in VIII.

II. Related work

The route planning problem can be formulated as follows: having a directed graph with non-negative weights assigned to edges find a path between a start point and end point minimizing the cost function usually assumed to be the sum of weights. The weights usually express such properties as travel time, distance, fuel consumption, air pollution, driver satisfaction, tolls, etc. The term *static route planning* denote the case where, these weights remain constant over time in opposition to *dynamic route planning*, where weights are variable due to changing traffic and weather conditions (e.g. traffic congestion increases the travel time and pollution or a rain slows down the velocity). For *personalized route planning*, a separate set of weights can be used for each user type (static dimension) but the weights may also reflect dynamically changing user state (preferences, fatigue, remaining fuel) .

Probably, the most known algorithm for determining the shortest path in a road network is Dijkstra's algorithm [1]. Dijkstra's algorithm can be improved by taking in to account an estimation of the cost from a location to the destination. A*-algorithm [2] uses a heuristic function to determine the order in which the search visits nodes in the tree. Nowadays the A*-algorithm is the most commonly used shortest path algorithm in geographical networks. Both algorithms can be used in either static or dynamic route planning problems.

In last years there was observed a rapid speed up of static route planning methods (up to 10^6 times in comparison to classical Dijkstra's algorithm). An excellent survey on this topic can be found in [3].

The general idea behind all speed up methods in static route planning relies in *precomputing* information that can speed up the queries. Assuming, that the graph of routes has n vertices, precomputing all n^2 routes between nodes and storing them in a lookup table would be the fastest solution. This, however, is still infeasible for large graphs (e.g. the graph of road network for Western Europe, which has 12 mln. nodes). Due to this, the novel optimization methods are combinations of precomputation and efforts to reduce the task complexity.

A multilevel technique described in [4] decomposes the graph into disjoint subgraphs with common border nodes and calculates additional edges (shortcuts) linking the border nodes.

Highway hierarchies [5] analyze precalculated shortes paths and selects highway edges, i.e. segments of optimal routes not close to the start and end points. They also remove low degree nodes drawing shortcuts. This method is somehow similar to the approach taken in commercial software that favours high class roads and looks for alternatives only when close to the source or target.

Highway node routing [6] arranges nodes into multilevel hierarchies and introduces shortcuts between them. The method is suited for the dynamic route planning, as a weight assigned to a shortcut edge at the level l can be calculated by performing low complexity optimization task at level $l - 1$.

As regards dynamic route planning, an algorithm for planning an optimum route that takes into account daily congestion patterns is considered in [7]. Routing algorithms for the k-shortest path problem for graphs with time-dependent edge costs was described in [8]. Adaptive route planning algorithm for handling real-time information, which enable only to determine the next road segment in the route was proposed in [9]. Dynamic route planning for car navigation using virus

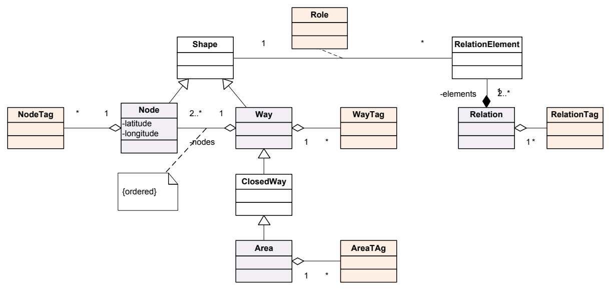

Fig. 2. Domain model of OSM

genetic algorithms was described in [10].

The practical problem of personalized route plannig, and in particular using ontologies to represent the task context, user preferences and profile, was studied in [11] and [12].

III. The maps

The INSIGMA system stores maps in databases whose structure is based on Open Street Map (OSM). OSM model [13] defines a map as a collection of vectors of three basic types (Fig. 2):

- Points (*Node*), whose attributes are geographical coordinates (longitude and latitude)
- Polylines (*Way*) defined as ordered sequences of nodes (including closed polylines: ClosedWay),
- Polygons (*Area*) being subclasses of Ways.

These vectors can be grouped within containers of *Relation* type and attributed with roles, typically: *route* (a route element) or *multipolygon* (an area with holes, eg. a lake with marked islands).

A *Node* element can be interpreted as any object represented by a point on the map, whereas a *Way* can be any linear element: a road, shoreline, railroad or a fence. OSM does not define dedicated data structures for different types of objects that can appear on the map. Instead, it uses a large set of tags that can be attributed to vectors or groups of vectors. They have the form of (*key*, *value*) pairs. For each key a set of possible values is defined. The set of pairs assigned to vectors determines, how they are interpreted. For example, a tunnel is defined as a *Way* with the *tunnel=yes* tag, a street can be specified as a single element Way with an attribute *highway=residential*, but also as a few *Way* vectors with the same name tag, a route with an assigned identifier can be defined as a relation, whose elements are *Way* segments.

In the INSIGMA system databases storing maps are logically decomposed into six components (Fig. 3)

- *Physical objects* containing routes and other persistent elements, as building outlines, railroads, water reservoirs, bridges
- *Traffic organization* information about prescribed directions, bans and limits affixed to the physical structure
- *Dynamic parameters* encompassing traffic parameters delivered by sensors (flow of vehicles, jams, travel time for selected roads) and information on events influencing the traffic flow (e.g. snowfall or road accidents)

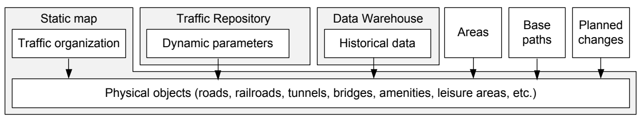

Fig. 3. Logical structure of INSIGMA maps

- *Historical data* containing the registered traffic parameters, that can be used for future route planning. The database stores also previously calculated routes (information on the start and end point location is limited to road segments).
- *Areas* auxiliary classifications of parts of the map used for selecting appropriate algorithm (heuristics) of route planning
- *Base paths* a set of predefined routes used by planning algorithms, in most cases they correspond to main roads
- *Planned changes* define future (planned) changes in traffic organization related to road works or various events.

The physical objects and traffic organization components are collectively referred as *Static Map*, dynamic parameters as *Traffic Repository* and historical data as *Data Warehouse*.

IV. Optimization algorithms

The Route Planning Subsystem operates on the model in form of a weighted graph obtained by a conversion of the dynamic map according to the problem parameters (start point, end point, area), temporal characteristics (travel planned at present or in the future), criterion function and user profile (constraints, preferences). A desirable effect of the conversion is a reduction of the graph size, what may speed up optimization procedures. Elements of the graph model are shown on Fig. 4. The basic model components are *Crossroads* (graph node) and *RoadSegment* (corresponding to an edge). Depending on the particular algorithm, the model can be extended by crossroads location and points defining geometry of road segments (*RoadPoints*), maneuvers at crossroads (*Turns*) and assigned parameters as length of the queue or passage time (*RoadWeights* and *TurnWeights*) or base path markers.

The subsystem determines a route based on an open set of implemented algorithms, which can be supplemented by new procedures. In the most sophisticated version two-phase approach was taken, where in the first phase inauguration methods are applied and in the second improvement algorithms (including population ones) are executed. The aim of the inauguration method is to deliver quickly an initial solution with accepted quality. Assessment of the calculated route is realized by a goal function reflecting expectations and criteria related to a user profile.

The goal function can be based on the travel time, summary distance, safety factors (crossroads with traffic lights, separation of lanes), the time spend in traffic jams, the number of points of interest passed by and any combination of these parameters.

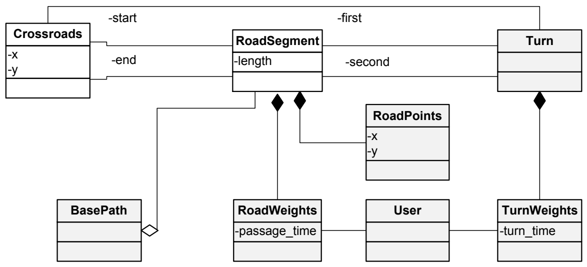

Fig. 4. Graph model used in route planning algorithms

In the inauguration phase one or more construction algorithms are used; their selection is influenced by various factors related to the algorithm type, task type, user attributes and areas in which the route is to be planned.

The general algorithm scheme consists in building a path starting from the initial node and selecting subsequent nodes among the neighbor nodes in line with the selected criterion (deterministic or with a probability proportional to the goal function value). The procedure is continued until the target node is reached.

In order to reduce the execution time of algorithms and increase the quality of obtained initial solutions, various supplementary information are used, that are usually ignored in classical algorithms finding the shortest path. They include geographical coordinates of the nodes, their distances or angles between the vectors spanned on possible edges and the vector pointing from the current to the target point. These factors allow to tune algorithms for coarse route calculation promoting nodes approaching the path to the specified target or intermediate targets. Another factor taken into account are predefined base paths determining recommended routes between selected nodes. In most cases they correspond to the high intensity road infrastructure (1st or 2nd class roads) and frequently selected travel directions. Where possible, subpaths of the base paths are used while constructing the planned route. The earlier established and stored solutions have similar role.

The efficiency of algorithms is also related to properties of the area covering the currently constructed path. Density of road connections, their directions, lengths of segments, average throughput, occurrence of traffic jams and other characteristics may determine the selection of an algorithm, that may cope with the area specificity.

Calculated initial solutions (satisfying constraints related to user preferences) are assessed according to user-defined criteria based on the current and predicted data (originating from the dynamic map and the data warehouse). The best of them can be immediately realized (e.g. by emergency services).

At the second stage improvement algorithms are applied taking as input full or partial solutions calculated in the first stage. This enables applying other approach then realized earlier. Inauguration algorithms of the first stage use multiple simple construction algorithm (e.g. use greedy rules) [14], [15] to yield a set of diverse solutions constituting the initial data for population methods aimed at solution improvement. Acting on a set guarantees that in most cases that several alternative routes can be obtained. At present several types of improvement algorithms are applied in the second stage e.g.: Evolution Algorithm (EA) [16], [17], PSO Algorithm (Particle Swarm Optimization) [18], [19] and Taboo Search Algorithm [20]. The first one is a population algorithm inspired by genetic principles. In the second case, a method of new solutions creation was adapted to the problem type, by directing particles in the solution space on the basis of specifically calculated velocity vector . The trace of the algorithm execution (stored in frequency-based memory) influences the probability of the road segment occurrence in a calculated solution.

V. Ontologies

Route Planning Subsystem uses several ontologies formalized in OWL language [21] to store semantic information about objects appearing on the maps, including road types, users, traffic monitoring parameters, weather conditions, recognized user preferences, types of algorithms as well as procedures implementing them with appropriate parameters. The ontologies were developed according to the methodology described in [22]. The set of ontologies is highly modular and shares a small common upper ontology.

According to [23] ontologies in information systems can be used during development and run-time phases of the software lifecycle. In the development phase ontologies were used to generate schemas of databases: Static Map and Traffic Repository. In spite of relational representation, the semantic information (namely URIs of classes and properties) are maintained as additional attributes providing semantic interoperability at the run time.

The rest of the section discusses main ontologies relevant to the route planning tasks.

OSM ontology: The ontology defines classes of objects appearing on maps: roads, railways, water ways, amenities, emergency infrastructure, public transport, shops, tourist attractions, etc. This large ontology contains about 660 classes, which were identified based on the published set of OSM tags and their values [24]. This set is continuously extended and refined to satisfy various needs and local specificities. As the Open Street Map community is rapidly growing (the number of registered users reached 500000 in 2011), the list of map features constitute a common knowledge shared among large group of committed users. The OSM ontology formalizes this knowledge and additionally provides information about how these classes of map objects are represented by combinations of map primitives (*Node*, *Way*, *Area*), attributed tags and their values. The taxonomy of objects specified in the OSM ontology can support various tasks: visualization, searching for POIs (Points of Interest) and route planning.

Static map ontology: The ontology specifies additional data structures included into the static map: lanes, crossroads and turns, as well as the taxonomy of their properties expressing physical characteristics (width, maximum height, turn radius, damaged surface), limits or obligations imposed by the traffic organization (speed limit, forbidden turn, etc.) and the properties originating from environment usually expressed by warning signs (wild animals, icy surface in winter, intense pedestrian traffic).

During the system development the OSM and Static map ontologies were used to define the structure and fill dictionary tables of the database referred as *Static Map* in the architectural diagrams (Fig. 5 and Fig. 6).

Monitoring parameters: The ontology formalizes the model of monitoring parameters stored in the *Traffic Repository* as composed of three basic concepts:

- Monitoring parameter *type* defining various quantitative properties pertaining to traffic and weather conditions: average speed, waiting time at the traffic lights, length of the vehicles queue, temperature, wind speed, rainfall, etc. Each type is assigned with a unit, a range of values and constraints specifying map objects to which a type can be linked.
- *Parameter instance* that bind a parameter with a location where a parameter is measured, an object (node,lane, turn or area) to which the measurement applies, frequency indicating how often the data are updated and the instance state (active, suspended, failure).
- Current parameter *value* assigned to an instance with accompanying timestamp and validity period.

The ontology is a formal basis for discovery and query services delivered by the *Traffic Repository* that allow client software (in particular route planning algorithms) to find monitored parameters in the indicated area and then access their current values.

Events: The ontology defines various events influencing the traffic: accidents, demonstrations, traffic jams, weather conditions, seasons. Events have spatiotemporal characteristic, i.e. they have the occurrence time, duration and they are attached to a certain location (point, road, area). The ontology also classify *Threats* understood as possible events that may result in damage (accident) or negative influence on traffic (jam). There are casual relations between events and threats, e.g. severe weather conditions may cause accidents.

Users (traffic participants): The ontology classifies vehicles based on such physical attributes as number of axles and dimensions. It specifies also their roles in the traffic: normal users, police, fire or medical services and users with special privileges, as handicapped persons or vehicles allowed to enter a restricted traffic zone.

Areas: The map is divided into areas (sometimes overlapping), to which the following attributes are assigned:

- the type of the road connections (density, road layout regular or irregular, numerous one way roads, road width, presence of multilane roads, intense pedestrian or bicycle traffic),
- traffic characteristics (presence of monitored parameters, average speed ranges, their variations in time, high probability of jams occurrences),
- safety characteristics (probability of car accidents),
- accessibility for users.

These attributes are inherited by all road segments belonging to a given area. They determine the selection of the algorithm that would cope with the area specificity.

Task parameters: The ontology formalizes the concepts and relations used while specifying the route planning tasks: route type (directly influencing the goal function), user profile (the type and the role of the vehicle), route points (start, end, intermediate) and planned time of starting or finishing the travel.

Algorithms: The ontology of algorithms classifies their types (exact, approximate, greedy), role (construction and improvement), enumerates criteria for selecting next nodes in heuristics. Optimization algorithm use the different criterion functions: minimization of road length, minimization of time, maximizing safety, maximizing comfort etc.

Algorithm implementations: The ontology contains addresses of web services implementing optimization procedures, describes their parameters and classifies them as instances of particular algorithm type.

Rules: The ontology stores rules for algorithm selection encoded in [25] and a narrow set of classes and properties supporting algorithm selection and configuration. It imports, however, the algorithm ontology, as parameters of algorithms are mentioned in rules.

VI. Architecture of the Route Planning Subsystem

The architecture of the Route Planning Subsystem is outlined in Fig. 5. Its basic elements are: *Client Interface* (implemented as a web service), a set of procedures implementing optimization algorithms and the *Decision Module*, that acts as a broker: it directs route planning requests to appropriate optimization procedures (also published as a web service).

The typical interaction scenario in the system is the following:

1) *Client* sends a request for route planning (indicating route points and a user profile). The *Client Interface* module creates a task and returns it URI to the client. The task is then passed to the *Decision Module*.
2) *Decision Module* checks whether analogous route is stored in the *Data Warehouse*; if so, it is added to the *Calculated Routes*.
3) *Decision Module* builds a list of algorithms A (or more precisely algorithm configurations) that can be applied for a given optimization problem on base of predefined rules (see section 6). The set of applicable algorithms A is decomposed into two disjoint subsets: A_c construction algorithms and A_i improvement algorithms.
4) *Decision Module* starts one or a few of construction algorithms A_c with highest priorities passing to them the task parameters.
5) Depending on the obtained quality metrics for the calculated solution, the *Decision Module* can start population

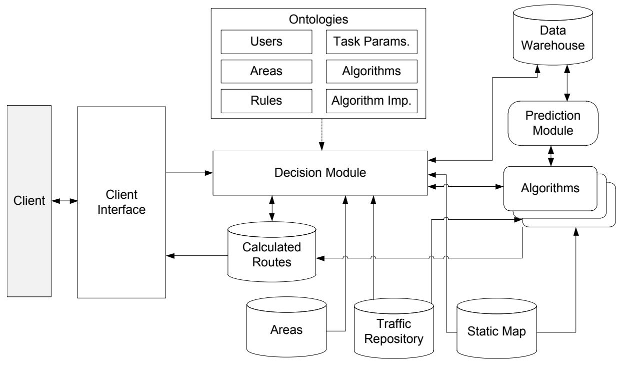

Fig. 5. Architecture of the route planning subsystem

improvement A_i algorithms passing to them the set of solutions obtained in the construction phase.

6) Information about the calculated solutions (including start and end area, user profile, criterion, timeslot, applied algorithm and value of the goal function) are stored in the *Data Warehouse*.
7) *Client* periodically polls the service in order to check if a calculated route is available. It can chose between getting the best solution or a list of alternative routes (if available).
8) *Mobile client* can repeat the request passing an updated location and previously assigned task URI. In this case the scenario is continued at the step (4).

Algorithm implementations shown as overlapping blocks in Fig. 5 consists of four components: *Structure Adapter*, *Weights Adapter*, *Optimization Model*, and *Optimization Procedure* (Fig. 6)

Structure Adapter builds the graph structure and stores it in *Optimization Model* from the data originating from the static map and the user profile. The structure adapter uses the OSM, static map and user ontologies to determine which roads and turns (manoeuvres on the crossroads) are accessible for a given user type. The structure adapter also introduces some graph weights that are stable over time, e.g. speed limits. The obtained graph can be shared by multiple algorithms. Moreover, to facilitate reusability it takes the form of the multigraph [26]. Depending on the algorithm implementation, the model can be stored in the database, what enables constructing optimization procedures based on built in relatively simple path calculating capabilities (e.g. PostGIS extension to PostgreSQL database [27] or in the memory. The memory based storage is preferred in case of improvement algorithms.

Weights Adapter calculates additional graph weights from time-varying parameters. It uses ontologies of users and monitoring parameters. It also relies on the services of the *Prediction Module* that, on the basis of information stored in the *Data Warehouse* predicts values of the relevant parameters in the optimization time horizon.

Optimization Procedure takes as input the requested route points, calculates the optimum solution and stores it in the database of calculated routes.

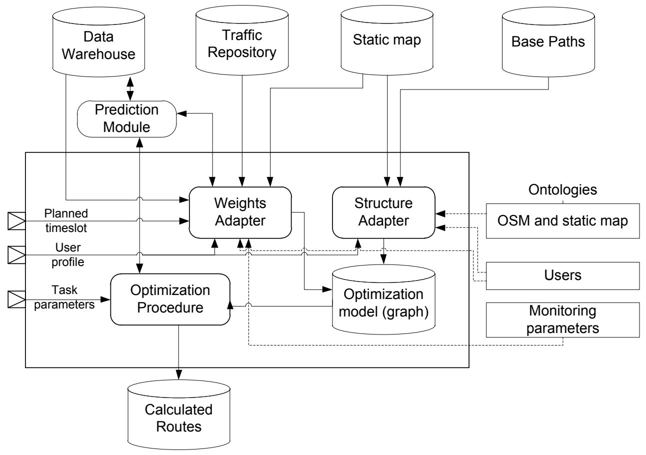

Fig. 6. Internal structure of an algorithm implementation

VII. Rules for selection of optimization algorithms

As indicated in the section VI, the *Decision Module* builds a list of optimization algorithms and their configurations that can be applied in a particular route planning problem instance. The algorithms selection is based on set of rules encoded in SWRL language [25] stored in the dedicated Rules ontology. This ontology imports several compound ontologies enumerated in the section V, because the rules premises and conclusions are expressed in terms of various concepts defined in the underlying ontologies. The high level concepts appearing in the rules definitions are:

- Task characteristics including defined route points (start, end, intermediary), areas to which the route points belong and their properties (street layout, density, road infrastructure, road classes, number of lanes, length of segments, intersections, traffic characteristic), distance to the base paths, travel time and distance.
- User profile, including vehicle type, traffic role, privileges and preferences.
- General context (including season and weather conditions).
- Algorithms (a class of algorithm and configuration parameters, usually flags and boolean switches).

Below, in Table I examples of two rules are presented:

- *SnowfallAndTruck* specifying, that: ”In case of snow fall and a truck vehicle, use a greedy algorithm, which prefers selection of main roads.”
- *VelocityDisparity* that can be formulated as: ”If any route point belongs to an area with observed speed disparity between roads, an algorithm should attempt to calculate routes alternative to base paths.”

The procedure of algorithm selection realized within the decision module consists of the following steps:

1) For the delivered route planning request, a temporary *ABox* is constructed by adding an individual p_{root} belonging to the auxiliary class *Premise* (the root element)

Rule SnowFallAndTruck:
Premise(?p),
context(?p, ?c),Context(?c),event(?c, ?e),SnowFall(?e),
user_profile(?p,?u),UserProfile(?u),vehicle_type(?u, ?vt),Truck(?vt)
$\rightarrow$ entails(?p, algorithm_sf),
GreedyAlgorithm(algorithm_sf),
alg.preferred_main_roads(algorithm_sf, true),
alg.priority(algorithm_sf, 0.7)

Rule VelocityDisparity:
Premise(?p),
task(?p, ?t),Task(?t),task.point(?t, ?p),RoutePoint(?p),
point.in_area(?p, ?a), Area(?a), area.velocity_disparity(?a, true)
$\rightarrow$ entails(?p, algorithm_vd),
alg.calculate_alternative_routes(algorithm_vd, true),
alg.priority(algorithm_vd, 0.65)

TABLE I
SAMPLE RULES SNOWFALLANDTRUCK AND VELOCITYDISPARITY

linked with subgraphs describing the task, the user profile and the context. The graph content is extended by various *decorators* interpreting the request parameters and, where needed, adding individuals and asserting relations (e.g. indicating a snow fall, setting vehicle type to a truck or the area.velocity_disparity flag).

2) For each rule an individual belonging to class *Algorithm* is added to the *ABox*. If n rules are defined, then n individuals are created. Conclusions of rules reference different individuals, e.g. algorithm_sf for the rule SnowfallAndTruck and algorithm_vd for VelocityDisparity.
3) In the next step the Pellet [28] reasoning engine is invoked to execute the set of rules and to infer assertions about algorithm type, its configuration parameters and a priority. As rules configure independent individuals, there is no dependency on order in which they are fired.
4) Finally, the list of resulting algorithms and their configurations is merged (algorithms and switches with the highest priorities are taken).

Fig. 7 illustrates the layout of the *ABox* resulting from rules executions. Continuous lines mark individuals and properties there were input before firing the rules, dashed lines are used to show properties and literals introduced to the *ABox* by rules' consequences. Several individuals belonging to the *Algorithm* class are linked by entails property with p_{root} element and assertions about their parameters are made. Each algorithm configuration is attributed with the priority parameter representing the belief that they are suitable for the given input conditions.

There are at least two approaches to constructing rules and interpreting the results of inferences made. The first approach consists in writing rule consequents in such a manner, that they define full, ready to use algorithm configurations, i.e. a web service address and a set of switches. In this case selection of an algorithm is straightforward: a configuration with the highest priority is taken.

The second approach is more fuzzy: the rules make less assertions and set only some switches (rules in Table I represent

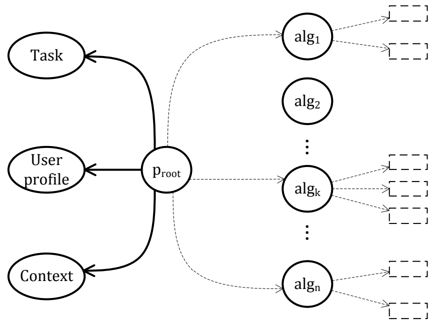

Fig. 7. ABox after applying the set of rules

this approach). As the result a sequence of algorithm configurations ordered by priority is obtained: $\alpha = (a_0, a_1, ..., a_r)$. We can not merge them all, as some configurations may be conflicting, e.g one configuration may set *preferred_main_roads* switch set to *on*, and an other to *off*. Such conflicts can be statically identified by an analysis of rule consequences and formally expressed as the $conflict$ relation between algorithm configurations. We calculate the highest priority set of non-conflicting configurations to be merged using the relatively simple heuristic procedure, that starts with an empty set A_0 and adds subsequent non-conflicting configurations from $a_0 \dots a_r$ according to the priority based ordering. If needed, analogously can be calculated other sets A_j (with the **for** loop starting from $i = j$).

```
$A_0 = \emptyset$
for $i = 0 \dots r$ do:
if $\forall a \in A_0 : (a, a_i) \notin conflict$ then add $a_i$ to $A_0$
merge all graphs from the set $A_0$
```

Certainly, an interesting issue is the source of knowledge encoded in rules. As the route planning subsystem is still in the development phase, the rules for algorithm selection and configuration originate from three sources: comparison of performance of implemented algorithms for a various test cases, experts assessment and architectural decisions.

Each implemented algorithm (understood here as an optimization procedure coupled with a road model) is tested on a few thousands of randomly generated route planning tasks. The tests allow to establish the performance and accuracy for specific groups of tasks. To give an example, algorithms from the A* family behave very well for urban environment: typically, an optimal route between points lying at a distance of 10 km is obtained within 100 ms. However their performance for longer routes is low, e.g. a 600 km route between two cities in Poland: Kraków and Gdańsk is calculated in 68 s. On the other hand, tests shown that greedy algorithms are generally inferior in urban environment, whereas they are capable to calculate long non-optimal routes in a few milliseconds.

The rule *SnowFallAndTruck* (see Table I) represents a typical rule introduced by an expert. It attempts to mimic qualitative user decision in case of severe weather conditions.

The same effect can be probably obtained in a quantitative model that would increase costs assigned to secondary or tertiary roads. However, validation of such model would have been extremely difficult.

The last source of rules is related to architectural decisions. The architecture of the Route Planning Subsystem can be considered as multi-instance (in opposition to multi-tenant). In particular, for selected privileged users, e.g. local police services, dedicated instances are developed, which uses much more detailed models of road network in an urban area including tracks and roads reserved for pedestrian traffic. Such models are not available for normal users.

We also expect, that during the system exploitation, we will be able to collect a representative set of historical data and perform in depth analysis aiming at optimizing and learning rules.

VIII. Conclusion

This paper discusses application of ontologies in the route planning subsystem for highly dynamic urban environment realized within the INSIGMA project. An innovative feature of the described approach is using semantic description of the route planning context including both static and dynamic properties: type of the road network, historical data, dynamically collected information about the traffic. The services of the system are adapted to user profile taking into account its preferences, and various constraints as physical dimensions of vehicles and traffic organization. A central role in the developed system plays the decision module, which based on set of rules and the knowledge stored in ontologies selects the most efficient algorithms and configures them to perform the route planning tasks. At the lower level of algorithms implementation ontologies are also used by structure and weights adapters that are responsible for construction of graph models constituting problem models for optimization procedures.

References

[1] E. W. Dijkstra, "A note on two problems in connexion with graphs." *Numerische Mathematik*, vol. 1, pp. 269–271, 1959.

[2] P. E. Hart, N. J. Nilsson, and B. Raphael, "A formal basis for the heuristic determination of minimum cost paths," *IEEE Trans. Systems Science and Cybernetics*, vol. 4, no. 2, pp. 100–107, 1968.

[3] D. Delling, P. Sanders, D. Schultes, and D. Wagner, "Engineering route planning algorithms," *Algorithmics of large and complex networks*, vol. 2, pp. 117–139, 2009. [Online]. Available: http://www.springerlink.com/index/J446P618705P150J.pdf

[4] F. Schulz, D. Wagner, and K. Weihe, "Dijkstra s algorithm on line : An empirical case study from public railroad transport 1," *Algorithm Engineering*, vol. 1668, pp. 110–123, 1999. [Online]. Available: http://www.springerlink.com/index/79LVR2MR29Q38JMJ.pdf

[5] P. Sanders and D. Schultes, "Engineering highway hierarchies," *Engineering*, vol. 4168, pp. 804–816, 2006. [Online]. Available: http://www.springerlink.com/index/u2k30471700g747m.pdf

[6] D. Schultes and P. Sanders, "Dynamic highway-node routing," *Proceedings of the 6th international conference on Experimental algorithms*, pp. 66–79, 2007. [Online]. Available: http://portal.acm.org/citation.cfm?id=1768578

[7] A. Orda and R. Rom, "Distributed shortest-path protocols for time-dependent networks," *Distributed Computing*, vol. 10, no. 1, pp. 49–62, 1996.

[8] S. Subramanian, "Routing algorithms for dynamic, intelligent transportation networks," Ph.D. dissertation, Virginia Polytechnic Institute and State University, 1999, aAI9923378.

[9] L. Fu, "An adaptive routing algorithm for in-vehicle route guidance systems with real-time information," *Transportation Research Part B: Methodological*, vol. 35, no. 8, pp. 749 – 765, 2001. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0191261500000199

[10] H. Kanoh, "Dynamic route planning for car navigation systems using virus genetic algorithms," *International Journal of KnowledgeBased and Intelligent Engineering Systems*, vol. 11, no. 1, p. 6578, 2007. [Online]. Available: http://iospress.metapress.com/index/f2jyhy81u1jae5ma.pdf

[11] A. S. Niaraki and K. Kim, "Ontology based personalized route planning system using a multi-criteria decision making approach," *Expert Systems with Applications*, vol. 36, no. 2, pp. 2250–2259, 2009. [Online]. Available: http://linkinghub.elsevier.com/retrieve/pii/S0957417407006902

[12] Z. Shen, F. Hu, and M. Kawakami, "Personalized route planning based on the semantic network: A case study of Kanazawa city, Japan," in *Geoinformatics, 2011 19th International Conference*, June 2011, pp. 1 –6.

[13] OpenStreetMap, "OpenStreetMap Wiki." [Online]. Available: http://wiki.openstreetmap.org/wiki/Main_Page

[14] B. V. Cherkassky, A. V. Goldberg, and T. Radzik, "Shortest paths algorithms: Theory and experimental evaluation," *Mathematical Programming*, vol. 73, no. 2, pp. 129–174, 1996. [Online]. Available: http://www.springerlink.com/index/10.1007/BF02592101

[15] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein, "Greedy algorithms," *Introduction to Algorithms*, vol. 56, no. 2, pp. 370–405, 2001.

[16] Z. Michalewicz, *Genetic Algorithms + Data Structures = Evolution Programs*, D. W. Loveland, Ed. Springer-Verlag, 1992, vol. 19, no. 3. [Online]. Available: http://portal.acm.org/citation.cfm?id=229930

[17] Z. Michalewicz and D. B. Fogel, *How to Solve It : Modern Heuristics*. Springer, 2000, vol. 32, no. 1.

[18] C. Liao and P. Luarn, "A discrete version of particle swarm optimization for flowshop scheduling problems," *Computers & Operations Research*, vol. 34, no. 10, pp. 3099–3111, 2007. [Online]. Available: http://linkinghub.elsevier.com/retrieve/pii/S0305054805003643

[19] W. Chmiel, P. Kadłuczka, and G. Packanik, "Implementations of pso algorithm for permutation problems," *Automatyka*, vol. 15, pp. 117–126, 2011.

[20] W. Chmiel and P. Kadłuczka, *Implementation of conditional expectation value of criterion function in implementation of approximation algorithms*. Wydawnictwa Naukowo-Techniczne, 2004, pp. 27–33.

[21] B. Motik, P. F. Patel-Schneider, and B. Parsia, "OWL 2 web ontology language structural specification and functional-style syntax," *Direct*, vol. 27, no. October, pp. 1–133, 2009. [Online]. Available: http://www.w3.org/TR/owl2-syntax/

[22] J. Śliwa, K. Gleba, W. Chmiel, P. Szwed, and A. Glowacz, "IOEM - ontology engineering methodology for large systems," in *ICCCI (1)*, ser. Lecture Notes in Computer Science, P. Jedrzejowicz, N. T. Nguyen, and K. Hoang, Eds., vol. 6922. Springer, 2011, pp. 602–611.

[23] N. Guarino, "Formal ontology and information systems," *Proceedings of FOIS98*, vol. 46, no. June, pp. 3–15, 1998. [Online]. Available: http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.29.1776&rep=rep1&type=pdf

[24] OpenStreetMap, "Map features - OpenStreetMap Wiki." [Online]. Available: http://wiki.openstreetmap.org/wiki/Map_Features

[25] I. Horrocks, P. F. Patel-Schneider, H. Boley, S. Tabet, B. Grosof, and M. Dean, "SWRL: A semantic web rule language combining OWL and RuleML," pp. 1–22, 21 May 2004. [Online]. Available: http://www.w3.org/Submission/SWRL/

[26] W. Chmiel, P. Kadłuczka, and S. Ernst, *A Multicriteria Model for Dynamic Route Planning*. Springer Berlin / Heidelberg, 2011, vol. 149, pp. 174–182. [Online]. Available: http://www.springerlink.com/content/g34u1612366x4285/

[27] R. O. Obe and L. S. Hsu, *PostGIS in Action*, S. SterlingEditors, Ed. Manning, 2011, no. 3/15/2010.

[28] Clark&Parsia, "Pellet: OWL 2 Reasoner for Java." [Online]. Available: http://clarkparsia.com/pellet/