# A Structure of Problem-solving Methods for Real-time Decision Support in Traffic Control

Martin Molina, Josefa Hernandez, Jose Cuena

Department of Artificial Intelligence, Universidad Politécnica de Madrid

Campus de Montegancedo S/N, 28660 Boadilla del Monte, Madrid, SPAIN

{mmolina,phernan,jcuena}@dia.fi.upm.es

http://dia.fi.upm.es

This version corresponds to a preprint
of the actual paper published in:
*International Journal of Human and Computer Studies,*
*(1998) 49, 577-600*

# A Structure of Problem-solving Methods for Real-time Decision Support in Traffic Control

Martin Molina, Josefa Hernandez, Jose Cuena

Department of Artificial Intelligence, Technical University of Madrid

Campus de Montegancedo s/n, 28660 Boadilla del Monte, Madrid, SPAIN

{mmolina,phernan,jcuena}@dia.fi.upm.es

http://dia.fi.upm.es

**Summary**. This article describes a knowledge-based application in the domain of road traffic management that we have developed following a knowledge modeling approach and the notion of problem-solving method. The article presents first a domain-independent model for real-time decision support as a structured collection of problem solving methods. Then, it is described how this general model is used to develop an operational version for the domain of traffic management. For this purpose, a particular knowledge modeling tool, called KSM (Knowledge Structure Manager), was applied. Finally, the article shows an application developed for a traffic network of the city of Madrid and it is compared with a second application developed for a different traffic area of the city of Barcelona.

## 1 Introduction

During the last 15 years, the knowledge engineering community has proposed different solutions to improve the development of knowledge-based systems considering this process as a knowledge modeling activity where high level conceptual entities are used to configure the final system. Within this context, the notion of problem-solving method (PSM) was proposed in different knowledge engineering frameworks such as the role-limiting method of McDermott (1988), the generic task of Chandrasekaran et al. (1992), and KADS, Schreiber et al. (1993). Problem-solving methods establish general strategies of reasoning defining types of knowledge that describe the role they play during the inference process. In large and complex applications that require knowledge-based solutions, the use of this type of components can significantly decrease the knowledge acquisition effort because they suggest possible knowledge types to be considered during the analysis of the problem. In addition to that, they can make better structured final applications, improving the understandability of the final system and, as a consequence, improving its reliability and flexibility for maintenance.

In this article, we describe how we have applied this approach to build a knowledge-based system in the domain of urban motorway traffic control. This system corresponds to a real-world application that was developed to be used in real-time as a decision-support tool in a control center on an urban motorway network area. The article presents first the domain of traffic management, identifying the main tasks in the decision making process. Then, a generic conceptual model for real-time decision support is described as a structured collection of problem-solving methods. This model is a general solution that we formulated to support the functional specifications established for the application. Then, it is described how we built the

operational version of such a model. For this purpose we used a particular tool, the KSM (Knowledge Structure Manager) environment, that helps developers in both the development of an operational version of a model and the maintenance of the final application. Finally, the article shows and compares two applications developed using this model for two different traffic networks of the cities of Madrid and Barcelona.

## 2 The Domain of Road Traffic Management

The road transport management has recently developed a significant demand of advanced information technology. Control centers for traffic management are connected on-line to devices (such as detectors on roads, cameras, traffic lights, etc.) in such a way that operators can supervise the state of the road by consulting data bases with recent information from detectors and can modify the state of control devices. The effective use of these traffic monitoring and management facilities requires sophisticated support tools for on-line operators, to help them in dealing with the information complexity and the diversity of sensors and control devices. In particular, expert systems for decision support have recently been successfully introduced in this field, Cuena et al., (1992), Deeter and Ritchie (1993), Molina et al. (1995).

Figure 1 shows a typical infrastructure for real-time traffic control that can be found in different cities. There are detectors on major roads recording several traffic measures such as speed (km/h), flow (vehicle/h) and occupancy (percentage of time the sensor is occupied by a vehicle). The distance between successive sensors on a freeway is usually about 500 meters. The information arrives periodically to the control center (e.g., every minute). The control center receives also information about the current state of control devices. Control devices include

traffic signals at intersections, traffic signals at on-ramps, variable message signs (VMS) that can present different messages to motorists (e.g., warning about an existing congestion or alternative path recommendation), radio advisory systems to broadcast messages to motorists, and reversible lanes (i.e., freeway lanes whose direction can be selected according to the current and expected traffic demand). In the control center, operators interpret sensor data and detect the presence of problems and their possible causes. Problems are congested areas at certain locations caused by lack of capacity due to accidents, excess of demand (like rush hours), etc. In addition, operators determine control actions to solve or reduce the severity of existing problems. For instance, they can increase the duration of a phase (e.g., green time) at a traffic signal, or they may decide to show certain messages on some VMSs to divert traffic.

According to this scenario, a real-time decision support tool that helps operators in detecting traffic problems and choosing appropriate control actions requires to provide the answers to three types of questions, Cuena and Hernández (1997): (1) *what is happening?*, i.e., what problem of a predefined set of problem classes is happening in the current state of the system, (2) *what may happen if?*, i.e., what relevant events may happen in the near future if some external conditions change according to certain hypotheses, and (3) *what to do if?*, i.e., what should be done in order to improve the current state considering hypotheses about external conditions.

## 3 A Generic Knowledge Model for Real-time Decision Support

This section describes the domain-independent model for real-time decision support that was designed to be applied to the traffic management domain in order to support the functionality described in the previous section. The generic knowledge model presented in this section is

described in abstract terms and could be used to develop applications in other domains different from traffic control, although here some general concepts are associated to examples from this domain to illustrate the description.

The model presented in this section is the result of a *phase of analysis* of the problem, where we used similar types of description entities that are present in CommonKADS, Schreiber et al. (1994), and other modeling frameworks based on the *task-method-domain* approach. According to this, this section describes first the task-method structure of the model, then the primitive inferences and, finally, the domain abstraction, following the three knowledge categories established by the CommonKADS methodology: task knowledge, inference knowledge and domain knowledge. In addition to this, the operational version of this model requires developing a *phase of design*, where it is necessary to use additional description entities to synthesize and extend the previous model, together with certain computational constructs. For this purpose, we used the KSM software environment, as it will be described in section 4.

The model uses the following basic concepts to describe the problem. There is an abstract *dynamic system* that evolves over time and that might present certain functional problems (e.g., the traffic on an urban motorway network). The dynamic system is divided into *components* (e.g., one way of a particular highway). A component includes a set of *resources* (e.g., cross sections, links, etc.) that impose certain limitations of capacity. The system's state is monitored periodically (e.g., every minute) by using *detectors* that record basic magnitudes, *observables* (e.g., traffic speed and flow) at significant locations. There is an external *environment* that establishes certain *demand* of resources (e.g., the demand in the domain of traffic is expressed in

number of vehicles going from certain origins to certain destinations). In addition to that, an operator can *regulate* the demand by using certain control devices or *effectors* (e.g., changeable message signs that recommend alternative paths). Both, the environment demand and the operator regulation using effectors, are *external actions* on the system. Components may present *problems* at certain *critical resources* (e.g. an accident at an off-ramp) that are cause of *symptoms* (e.g., a traffic queue).

### 3.1 The Task-Method Structure

Several tasks-methods-subtasks trees (figure 2) can answer the three types of questions defined in section 2. The *what is happening* type of questions may be answered by a *diagnosis* task that analyzes the current state of the dynamic system to detect and to diagnose existing problems. For instance, in the context of traffic control, this task detects the presence of a traffic problem (e.g., a congested area) and it finds out its causes (e.g., the set of paths that significantly contribute to a high traffic density). Given the complexity of real systems, usually the surveillance cannot be done for the whole system at once but it must be done for different separate components (e.g., in the traffic domain, each individual road is supervised separately instead of monitoring directly the whole network). The result of the total diagnosis is the union of local diagnosis for the different components. Thus, the diagnosis task is carried out by a method that divides the whole area into simpler components (using the *select component* task) and then it applies local diagnosis for each one (using the *local diagnosis* task). This second task is done by a method that first identifies the presence of a problem and, then, finds out the final contributors that explain such a problem. The identification of problems is done following the *heuristic classification* method that decomposes the diagnosis task into three subtasks: *abstract state* (e.g., in the traffic

domain, data such as the speed recorded by detectors is abstracted into significant qualitative values), *identify problem* (e.g., an accident or a saturated off-ramp) and *refine problem*.

Actually, what we call problem identification in the domain of traffic control is a task that not only identifies symptoms, such as the presence of a queue, but also identifies a first level of causes, for instance, an accident at a certain location. However, in order to have a complete comprehension of the problem to propose appropriate solutions, it is necessary to determine the final contributors of the problem. In the context of traffic control, this means that it is necessary to determine and quantify which paths contribute with a significant amount of traffic flow to the presence of the problem. For this purpose, traffic experts estimate the current demand according to historical information and use a simplified behavior model of the road network to determine the paths that significantly contribute to the problem. This reasoning is modeled by a task called *determine contributors* and is carried out by a method called *model-based analysis* that first analyzes the behavior of the dynamic system, according to the estimated current demand and, then, selects the specific elements that contribute to the presence of the problem. The behavior analysis of the dynamic system is done in two steps: estimating the external actions and simulating the behavior. The estimation of the external actions is done by three subtasks: *estimate the global demand* using historical information, *estimate the regulation effect* (e.g., the estimated effect of a path recommendation or a warning about the presence of traffic queues), and *distribute the demand* on resources taking into account such a regulation effect.

The second type of question (*what may happen if*) can be answered by a *prediction* task that forecasts the short term future behavior of the system in order to determine the severity of the

existing problems, accepting optionally as input different hypotheses of external actions (e.g., in the traffic domain, hypotheses of different traffic demand or hypotheses of different control actions). As the previous diagnosis task, this task separates the whole system into components, considering two different subtasks: the *select component* task and the *local prediction* task. The second task is carried out by a method that uses a model of the component to simulate its behavior, which is the same method used for diagnosis. It (1) predicts the external demand by considering future situations (when they are not received as input), and (2) simulates the behavior to estimate the severity of problems at critical locations. Note that the use of different hypotheses allows the user to study the problem with different external conditions following a dialogue where the user proposes different regulation actions and the system answers with their estimated impacts.

Finally, the third type of question (*what to do if*) can be answered by a *configuration* task that finds out solutions that can eliminate or alleviate current problems. This task, like the others, separates the whole system in components. However, in this case, the global solution cannot be directly the addition of local solutions but they must be combined considering their interactions (this is why here the *system decomposition* method includes a last step to combine proposals). In the case of the traffic domain, for instance, a local solution (e.g., a path recommendation) to decrease a traffic queue in a particular area can increase an existing queue in another area. Thus, the main task is divided into three subtasks: the *select component* subtask, that selects a component each time, the *local regulation* subtask, that finds out a local solution for a given component, and the *global regulation* subtask, that integrates local solutions to define global proposals. In its turn, the *local regulation* task is carried out by a *generate-and-test* method that,

first, proposes local solutions for current problems according to local heuristics and, second, it tests (by simulation) the effect of the proposed control actions in order to select the best solutions. Then, the *global regulation* task is carried out by a *propose-and-revise* method that, first, generates a combination of local solutions in the propose step and, then, the combination is analyzed to verify whether it satisfies compatibility constraints. When an incompatibility is detected, revision steps modify the global proposal using knowledge about priorities among components.

In summary, this task structure integrates several tasks and subtasks that offer the three main functions of the model for real-time decision support. This model includes a total of 8 problem-solving methods: system decomposition, identify & explain, heuristic classification, model-based analysis, model-based simulation, demand estimation, generate & test and propose & revise. The following sections refine this model describing in more detail the primitive inferences (elementary tasks) and the domain abstraction in which this model is based.

### 3.2 The Primitive Inferences

The total set of primitive inferences corresponding to the previous task model for decision support is shown in figure 3. This table shows primitive inferences, following the CommonKADS terminology, with their respective input and output roles and domain knowledge. For instance, the first primitive inference, called *select component*, is used to separate the whole system model into smaller components. It receives as input the whole system where the decision support model is going to reason and generates as output an individual component, using as domain knowledge a model of the dynamic system.

The next three primitive inferences are used to identify problems. The primitive inference called *abstract state* is an abstraction step that interprets sensor data. It receives as input the component and the observables (e.g., traffic speed and traffic flow at certain locations), and generates qualitative information about the state (e.g., the presence of a queue) that plays the role of symptoms. This task uses abstraction functions that relate raw data with higher level parameters. The primitive inference called *match problem* receives the component, symptoms and observables, and identifies types of problems in the physical system (such as accidents or saturated sections) that may condition the system's behavior. This task uses a set of generic problem scenarios that cover the set of problem types that the system may present. The next primitive inference, called *refine problem*, is used to determine details about the detected type of problem (e.g., the specific location of an accident). It receives as input the component, the type of problem, symptoms and observables, and generates as output specific problems using a system model as domain knowledge.

The next five primitives are used to analyze and simulate the behavior of the dynamic system. The primitive inference called *estimate global demand* determines the current demand. For instance, in the context of traffic, this task estimates the demand of the traffic network using historical information. This primitive inference receives as input the component, the observables and the time, and generates hypotheses of global demand, using a demand model that includes historical behavior. The primitive inference called *estimate regulation effect* determines the effect of current (or hypothesized) control actions (e.g., the effect on capacity due to changes in the green time of certain traffic lights or the diversion effect of path recommendation messages). The

next primitive inference, called *distribute demand*, combines the global demand with the regulation effect to produce the final demand. The primitive inference called *simulate behavior* performs a simulation using as input the component, the observables, the problems and the demand. This primitive inference produces as output the use of resources (e.g., the estimated flow in the different paths of the network) and the problem severity, formulated as the unbalance between demand and capacity. Finally, the primitive called *select contributors* produces as output the contributors associated to a problem (e.g., critical paths that contribute to the presence of a traffic queue).

Finally, the last five primitive inferences are used to produce regulation proposals. The primitive inference called *generate proposal of regulation* includes heuristic knowledge to propose candidate local control actions for existing problems (e.g., when there is a problem of capacity, a solution is to increase the green time of certain traffic lights). These proposals are tested by simulation using the *local prediction* task and, then, the primitive inference *select local solution* is used to choose the best proposals according to their impacts. The last three primitive inferences, *propose global regulation* that integrates local solutions proposed for individual components, *verify regulation* that analyzes the previous global proposal to detect incompatibilities and *remedy proposal* that fixes incompatibilities, correspond to the three subtasks associated to the propose-&-revise strategy.

### 3.3 The Domain Abstraction

The previous set of primitive inferences makes use of certain domain knowledge as figure 3 shows. This section describes this domain knowledge in abstract terms, i.e. it is considered for a

generic dynamic system. The domain description is divided into modules that are associated to the corresponding primitive inferences:

- *System model*. The system model includes basic concepts modeling the physical structure of the dynamic system. This model is mainly used by the primitive inference called *simulate behavior*, but it is also used by other two tasks: *select component* to choose each time a particular component for applying a task, *refine problem* to find out about details for an existing type of problem, and *select contributors* to select the elements that contribute to the presence of a problem. For instance, in the traffic domain this model includes concepts defining the network structure such as nodes (entry and exit points), sections, links connecting sections (with several types of links such as on-ramp link and off-ramp link), origin-destination pairs, paths that establish itineraries between origin-destination pairs and road areas integrating the previous elements.

- *Abstraction functions*. These functions abstract numerical information into higher level qualitative parameters. In general, this knowledge includes: (1) numerical functions to compute new parameters using lower level parameters, and (2) the qualitative interpretation of numerical parameters considering noise and uncertainty in the input data. In particular, an example of abstraction knowledge in the traffic domain could be the qualitative speed (which may be low, medium or high) that is abstracted from the numerical parameter speed according to a one-dimensional possibility function.

- *Problem scenarios*. These scenarios represent patterns of problems that the system may present. Each pattern includes observable conditions to conclude the presence of a particular

type of problem. Thus, each pattern can be viewed as a frame-like representation that includes slots corresponding to observables (e.g., speed, occupancy and other measures at different locations) and symptoms abstracted from observables (e.g., circulation regime, saturation level, etc.).

- *Demand model*. This model includes historical knowledge about the environment (e.g., in the case of traffic, there are scenarios of traffic demand, i.e., the amount of vehicles going from certain origins to certain destinations). The model includes temporal references in order to be able of making predictions.

- *Control actions model*. This model includes knowledge about control actions and associates to them (1) system state conditions that are compatible with the control action, and (2) the estimated effect of control actions. The state conditions can be viewed as the context in which the action has the estimated effect.

- *Compatibility model*. A model including compatibility constraints is required in order to analyze the correctness of the synthesis of local proposals for solving problems. An example of incompatible situation in the traffic domain is the presence of the following two actions: (i) the first one recommends to follow path *P* to go to a certain destination by displaying messages to the drivers in changeable message panels and (ii) the second control action reduces the capacity of a section belonging to path *P* by decreasing the duration of the green time of a particular traffic light. These two control actions are incompatible given that one of them increases the demand of the path, whereas the other decreases its capacity.

- *Priority model*. Finally, the priority model is used to solve violations among local proposals. The priority is used by a *propose-&-revise* method that modifies proposals of global solutions to successively eliminate detected violations by applying the corresponding solutions. Thus, when an incompatibility is detected the component with the lowest priority is asked to propose a new control action. The level of priority can be dynamically deduced from the current situation (e.g., considering the severity of current problems).

## 4 The Operational Model for Real-time Decision Support in Traffic Control

The previous section shows a generic knowledge model for real-time decision support which was the result of a phase of *analysis* of the problem using a methodology similar to CommonKADS. However, its translation into an operational version executable on the computer still presents some technical difficulties that have to be solved by applying certain *design* decisions. For this purpose, we used the KSM (Knowledge Structure Manager) tool, a software environment for knowledge modeling that is especially useful for the development and maintenance of the final operational version of a knowledge model, Cuena and Molina (1994), Cuena and Molina (1997) (`http://www.isys.dia.fi.upm.es/ksm`). KSM like other knowledge modeling tools for building operational models, such as Protégé-II, (Musen et al. 1995) or Krest (Macintyre 1993) includes the problem-solving method approach to formulate knowledge models but also introduces new modeling entities that are useful to facilitate the construction and maintenance of an efficient final operational version. KSM also provides reusable pre-programmed components, representation languages and graphical tools to produce the operational system.

In particular, KSM introduces an additional knowledge modeling entity, called *knowledge area*, which is used as a basic module for structuring tasks, methods and domains. This entity helps to provide more synthetic views of the knowledge model and facilitates the construction of the operational version. In general, a knowledge area follows the intuition of a body of expertise that explains a certain problem-solving competence. A knowledge area integrates in a module a set of tasks with their corresponding problem-solving methods. A cognitive architecture can be viewed as a hierarchically structured collection of knowledge areas at different levels of detail, where each knowledge area represents a particular qualification that supports specific problem-solving actions. According to this, the model is viewed with a top-level area that is divided (using the part-of relation) into other more detailed subareas that, in their turn, are divided into other simpler areas and so on, developing the whole hierarchy (where some areas may belong to more than one higher level area). A bottom level area is called *primary knowledge area* and defines, together with a set of primary tasks, a local view of the domain knowledge that may be operationalized by using a local knowledge representation formalism supported by a basic software building block. Primary areas also encapsulate a set of basic concepts, in the so-called *conceptual vocabulary*, which may be shared with other primary areas.

Following the knowledge-area organization, the original task-method structure for decision support (figure 2) which was useful to analyze the expertise, can be reorganized as it is described in figure 4 in order to facilitate the development of the operational version. This structure includes a top-level area representing the whole knowledge about the dynamic system management. This area offers the three main tasks of the model: *diagnosis* task, *prediction* task and *configuration* task. The area is decomposed into two subareas following a *part-of* relation:

*component knowledge*, which includes knowledge about a particular component of the system, and *coordination knowledge*, which includes knowledge about the coordination of local proposals. The component knowledge area is decomposed into three simpler areas: (1) *problems knowledge*, to understand about problems that can occur within the component, (2) *behavior knowledge*, that includes knowledge about the component to be used for abstraction and simulation of its behavior, and (3) *regulation knowledge*, to propose and evaluate regulation proposals in order to solve existing problems. In their turn, these areas are refined and decomposed as it is shown in figure 4.

Note that this structure basically includes at the bottom level the same areas identified in the domain abstraction (*problem scenarios*, *demand model*, *control action plans*, *compatibility model* and *priority model*) with the associated elementary tasks. As an exception, the *physical structure* integrates two areas: the *component model* and the *abstraction functions* (this integration was done mainly to improve the efficiency of the final implementation). There is also a new area compared to the domain abstraction, the *combination model*. This area is included to support the *propose regulation* task, and given that it is based on an algorithmic approach, it does not require a domain model. Thus, in KSM the developer establishes a partition of the domain abstraction, considering each module as a primary area. The higher level areas are aggregations of these elements. Therefore, the knowledge-area view can be considered as a way to organize and structure the domain abstraction. But, on the other hand, knowledge areas are modules including tasks, so they serve also to integrate subsets of tasks in a more synthetic organization.

With this organization, tasks are part of knowledge areas, and task-trees can be automatically derived from the knowledge area structure. Thus, for instance, the *prediction* task (one of the three main tasks) is associated to the *system knowledge* area. Likewise, the task *match problem* is associated to the primary area *problem scenarios*. One of the advantages of this explicit association between tasks and knowledge areas is that it is useful to avoid certain types of tasks. In general, tasks selecting parts of the knowledge model can be removed given that there will be explicit areas for these parts. For instance, the task *select-component* is necessary in principle to successively choose a component to diagnose, predict and configure. However in the KSM approach, given that there is a knowledge area for each type of component, this task is not necessary.

Methods in KSM are associated to tasks and they are formulated using a particular language called *Link*, Molina et al. (1998*)*. This language allows developers to specify the control-knowledge of their problem-solving methods. For instance, figure 5 shows the particular version of the propose and revise strategy to configure global proposals of regulation. In the example, data flow section defines how tasks are connected (where tasks are associated to knowledge areas). The control flow section uses rules to establish the execution order of the tasks. In this case there are three rules that establish a loop until the proposal is compatible.

Finally, there are also common concepts, i.e. a conceptual vocabulary, shared by different primary areas. In KSM, these vocabularies are formulated using a particular language called *Concel* that uses a concept-attribute-facet representation together with an organization in classes-subclasses-instances. In this model, there is a vocabulary that includes the set of basic concepts for each component of the system (e.g. sensor, effector, component, system, etc.). Note, however,

that a vocabulary is not a description of the whole domain knowledge. It only includes basic terms shared between different knowledge areas. Figure 6 shows a partial vocabulary written using the Concel language.

In summary, this structure of knowledge-areas, tasks, methods and vocabularies constitutes a pattern that is general and reusable. It is an abstract description of the types of knowledge and strategies of reasoning that are present in the real-time decision support scenario. However, it still needs to be refined by selecting the appropriate basic knowledge representations and inference methods for each primary area. For this purpose, as it will be presented below, it is necessary to choose appropriate primitives of representation that impose particular domain description languages and provide local knowledge-acquisition facilities. The resulting structure of knowledge areas will be very appropriate to develop the final operational model by following a process of duplication and specification of abstract knowledge areas, sharing by inheritance their relations and structure established a generic level.

### 4.1 The Operational Support by Primitives of Representation

The previous generic model needs to be refined by selecting appropriate symbolic representations and inference procedures. For this purpose, we used the concept of *primitive of representation* provided by KSM, Molina et al. (1997). A primitive of representation is a reusable pre-programmed software component that implements a generic technique to solve certain classes of problems. The primitive defines a particular representation language together knowledge acquisition facilities and several inference procedures that provide a problem-solving competence. In order to produce the operational version of the knowledge model, the developer

associates a primitive to each primary knowledge area. The selection of appropriate primitives of representation to support a knowledge model is an important step within the development of the application where the developer establishes a connection between the conceptual description at knowledge level and the symbolic level support. KSM has a library of primitives of representation that can be extended with other more specific primitives. Note that the use of a library provides a multi-representation environment where developers can select the most appropriate technique for each module of the architecture. This is particularly important in real systems where efficiency must be taken into account. This library is open, i.e., new additional primitives can be included in the library according to the particular needs of each application.

In particular, for the case of the knowledge model described in this article, figure 7 shows the primitives associated to each primary area. Usually, the developer tries to use domain-independent primitives in order to keep the generality of the knowledge model. Here, knowledge-based techniques for knowledge representation such as rules, frames or constraints are very appropriate given that they provide a good level of flexibility to be applied to different domains. For instance, in our model, rule-based and frame-based primitives were used to support four knowledge-areas. However, this generality is not always possible. Sometimes it is necessary to use more specific domain dependent primitives to keep the required efficiency and level of explanation for the final implementation. In the case of the decision support model for traffic control it was necessary to use a traffic network simulator as a primitive that supports the knowledge about the physical structure knowledge. This primitive is still quite general (because it does not depend on the particular traffic network) but it is specific for the domain of traffic management. Likewise, the knowledge area for control action plans was supported by a more

specific primitive (although non domain-dependent) that provides a particular representation based on triplets of control actions, state conditions and effects. The model includes also a non knowledge-based primitive to support the combination model that follows an algorithmic approach to successively produce combinations of proposals.

In more detail, the problem scenarios primary area is formulated as a frame base where each frame represents a type of problem. The reasoning method evaluates the matching degree of the real situation and the collection of problem frames using a particular uncertainty model. Figure 8 shows an example of a frame describing a traffic problem in a motorway corresponding to a traffic jam caused by an on-ramp. The frame includes at the beginning a section (called variables definition) to define the set of variables that will be used within the frame. These variables are dynamically associated to particular values during the inference process by using the conceptual description of particular roads. In this example, several variables are used to define a generic part of the road where there is an on-ramp (the three variables called previous, next and ramp identify the relevant points of the road area). Then, the next section of the frame (called description) is used to formulate the traffic state by associating values to attributes (slots). For instance, the speed of the previous section must be low, the occupancy of the previous section must be high, and so on. The value of attributes can be: a single qualitative value, a list (this means that the actual value must belong to that list), a number, a variable or a numerical expression (this is a kind of if-needed procedure that dynamically computes the value of the attribute). Each attribute has a label (e.g., labels a, b, c, d, etc. in this example). Labels are used in the last section of the frame (called relevance of characteristics) to indicate when (and to what extend) the whole frame can be deduced if some of the slots match the actual situation. Here, a rule representation is used.

Each rule includes on the left-hand side a logical expression of labels of slots, where *and* is represented as a colon and *or* is represented as a semicolon. On the right hand side, a percentage indicates the matching degree of the frame when the slots whose labels are on the left-hand side of the rule match the actual situation. For instance, in the example the third rule means that if slot *a* or slot *b* match, and slot *h* matches, then the whole frame matches with a degree of 80%. Since slots may partially match the current state of the traffic, and there are several rules, an uncertainty model is used to compute the matching degree for the frame.

Another frame-based but simpler representation is used to support the demand model knowledge area. This knowledge is represented by using hierarchies of patterns of demand associated to temporal intervals, in such a way that it is possible to determine demand scenarios for a given present or future state. Figure 9 shows a partial example of such a frame. The first set of slots is used to define the temporal range where this demand is active, in this example, a weekday from 6:30 am to 9:30 am. Then, there are slots that represent origin-destination pairs with an approximate estimation of the traffic flow. For instance, from the origin Mataro to the destination Badalona Norte there will be an approximate flow of 1000 veh/h. This type of frames are not general, i. e. they must be given for each particular traffic area.

The physical structure is formulated by a primitive that uses a declarative description of a network as a graph with nodes and links together with abstraction functions associated to components. The description of the network is written using the Concel language provided by KSM (with concepts organized in classes and instances, with attributes and facets). On the other hand, abstraction functions are represented as possibility functions typically used in fuzzy logic

to abstract numerical data. Figure 10 shows the general definition of these functions to abstract speed, saturation level and occupancy. Although they are generally defined, it is possible to add particular versions of these functions for specific parts of the roads. Using this information the primitive makes abstractions about the traffic state for each section of the road. In addition to that, the primitive uses the model of the traffic network to simulate the traffic behavior. This simulation is based on the assignment of a traffic demand on the traffic network, and generates as output the expected flow at certain locations and the unbalance between demand and capacity at critical points.

The knowledge about control action plans is formulated as a collection of triplets <control-actions, state-conditions, estimated-effect>. Figure 11 shows an example of such a representation for a control action plan that warns about the presence of an incident (e.g., a traffic accident) at a particular location. The first section of this example shows a set of messages about the existence of an incident to be presented to the drivers using certain variable message panels. The second section (state conditions) includes a logical expression about the state of the road that must be true in order to apply this plan. Finally, the third section shows the expected effect of the control action on the traffic behavior. This is represented as the expected distribution of the traffic along different paths. For instance, the example says that to go from Meridiana to A18, between the 80% and 90% of traffic will take the path *by A17 road*, and between 10% and 20% will take the path *by secondary road*. This representation can be used to support two different types of inference methods. The first one determines the expected effect of current control actions and it is useful to interpret the current traffic situation. The second one works on the opposite direction.

It proposes control actions that are consistent with the current traffic state and achieve a set of required effects.

Concerning the compatibility model, a rule-based primitive is used. Here, rules represent no-good situations including on the left hand side a description of the conditions that make two control actions incompatible. For instance, there are rules to detect a situation when two different control actions want to display a different message in the same panel. Other sets of similar rules can be written to detect when a particular plan (that was selected to solve a problem in a certain problem area) can make worse another problem. Finally, a rule-based representation is also used to formulate knowledge to solve conflicts about incompatible control actions. The knowledge base includes facts and rules about levels of priority and rules that determine which problem area must change its proposal. A set of facts represent couples of traffic areas describing which has more priority. There can be also rules that deduce this priority from the severity of existing problems.

## 5 Applications for the cities of Madrid and Barcelona

The structure of knowledge-areas, tasks, vocabularies and primitives of representation for real-time decision support described above is a generic structure supported by general software components (together with a few components that are traffic domain dependent). This operational model is called TRYS (Tráfico: Razonamiento y Simulación) and constitutes a *specialized software environment* on real-time decision support that can be used as a tool for building particular applications for different sites. This section describes how this model was applied to produce two final operational applications for two different cities in Spain, Madrid, Cuena et al. (1996) and Barcelona, Cuena et al. (1995).

To develop a final application for a particular domain a developer creates a quasi-isomorphic structure of knowledge areas specialized in this domain as an instantiation of the general description. The basic idea is that for each *generic* component of the model there will be one or more *domain* components following the same relations established by the generic model, and whose abstract formulation is completed with domain knowledge. In more detail, the development of the final application follows three main steps: (1) *creation of the domain structure*, for each vocabulary of the generic model one or more domain vocabularies are created and for each generic knowledge area, one or more domain areas are defined (these domain components inherit the structure of the generic components) (2*) domain knowledge acquisition*, for each vocabulary a set of domain concepts are written as subclasses and instances of concepts included in generic vocabularies, and for each primary area domain knowledge is written by using the corresponding language provided by primitives, and (3) *refinement of problem-solving knowledge*, optionally it is also possible to refine at domain level the strategies of reasoning defined at generic level.

For instance, the generic model has been applied to develop a decision-support system for an extensive traffic network in Madrid. This network includes part of one ring road, the M30, and two motorway accesses to the city, the NIII and the NIV. The dimension of the area draws an approximate extension of 10 per 15 kilometers. The traffic control infrastructure includes 80 loop detectors and 30 VMS panels. These devices are connected with the traffic control center through fiber optic communications, which makes possible to receive data from sensors every minute and to display messages on VMS panels in real time. According to this, the knowledge-based system

must receive as input 1660 variables each time, four variables per detector in temporal series of five minutes plus three variables per VMS. The domain model for this application includes 8 local control regions together with the coordination knowledge. This number of regions was obtained considering both directions of traffic for every road and a division of the main ring road in two parts. The network areas assigned to the regions are control dependent, it means that some regions share control devices, usually those VMS panels that are close to the intersections between the areas.

In order to develop the domain model for Madrid, first, a set of 8 conceptual vocabularies was created, one for each region, as instances of the conceptual vocabulary defined for the generic model. Then, the generic structure of knowledge-areas (figure 4) was used as a pattern to create the domain structure of knowledge-areas for the application of Madrid. Figure 12 shows this structure where the top-level area has been instanced on the Madrid Traffic Network area, and the *component* knowledge area has been instanced (with its corresponding structure of simpler knowledge areas) on 8 different traffic areas (N-III inbound, N-III outbound, ..., M-30 East Southbound and M-30 East Northbound). Once this domain structure was created, vocabularies and knowledge bases for primary areas were written as a result of the domain knowledge acquisition. Domain conceptual vocabularies were written in Concel language and include subclasses and instances of classes defined in generic vocabularies. In particular, the set of 8 vocabularies includes a total of 427 concepts (about 40 origin and/or destination nodes, 79 links where problems may appear, 20 sections and 120 paths). Each primary area (except one, the combination model) contains a knowledge base. The set of knowledge acquisition facilities, provided by each primitive associated to each primary area, is used to build particular knowledge

bases. For example, the knowledge model includes 8 instances of the problem scenarios knowledge area (one for each traffic region), therefore 8 knowledge bases were written using the language provided by the frame-based primitive associated to this knowledge area. The same operation was done for the rest of knowledge bases, which produced a total of 34 knowledge bases for the model. For example, the model includes a total of 84 problem frames that include 2940 logical expressions distributed in different slots, and 104 frames for control actions plans with a total of 2038 logical expressions in slots. Finally, it was also necessary to refine the problem solving knowledge (using the Link language) associated to the top-level knowledge area in order to consider the 8 regions that are include in this model.

This model was developed along three years (including the definition and development of the generic model with the corresponding software components). It was installed at the traffic control center of the Directorate of Traffic Management and connected with a real time database in June 1995. The model detects problems and suggests panel messages to control the problems in the M30, N-III and N-IV roads. The application works on a Unix workstation connected to the rest of the computers in the control center using a local area network. Figure 13 shows a screen of the system (translated into English). Presently, a current project is being developed to add four more traffic areas to this model corresponding to the roads NV and M607.

The same generic model was reused to build another system of a larger size, dealing with several traffic areas of the city of Barcelona (an urban ring that includes Ronda de Dalt and Ronda Litoral and five accesses: the roads A2, A17, A18, A19 and C246). In this case, the network was distributed in 22 local control regions (14 more regions than the model of Madrid). The traffic

control infrastructure available in Barcelona includes 52 VMS panels, ramp metering at 7 entrances to the ring road, and more than 300 loop detectors (this means that more than 5150 variables are received each time period by the knowledge-based system). The same strategy applied to develop the domain knowledge model for traffic control in Madrid was used for building the model of traffic control in Barcelona. This produced a total of 22 vocabularies and a similar but larger structure of knowledge areas (181 areas). Vocabularies contain a total of 3680 traffic concepts, and the model has 90 knowledge bases (22 problem scenarios, 22 demand models, 22 physical structures, 22 sets of control actions plans, plus one with the compatibility model and another one with the priority model). For instance, the model includes a total of 126 problem frames (with 3780 logical expressions) and 188 control action plans (with about 4700 logical expressions). The model was installed at the Traffic Control Center of Barcelona (Jefatura Provincial de Trafico) in July 1996. In contrast to the application of Madrid, that works on a single workstation connected on-line to the rest of the information system, the application of Barcelona was fully integrated with the rest of the information system. It was distributed in a set of Unix processes and integrated in the global architecture established for the information system of the control center. This produced a better efficiency and a good integration with the global user interface.

The application for Barcelona was developed during one year (four persons) which meant a significant decrease in the required effort compared to the case of Madrid. This was done thanks to the possibility of reusing the generic model applied to Madrid and the operational support based on software components. The main activities to develop the application of Barcelona were focused on knowledge elicitation following the knowledge areas established for the generic

model, together with activities related to software development in order to integrate the knowledge-model in the control center. As a result, this experience showed that the use of knowledge models at generic level based on high level structuring concepts (tasks, problem-solving methods, knowledge areas, etc.) was extremely useful in order to increase the productivity and the quality of the final system in the development of complex and large knowledge systems as it is the case of decision-support applications for traffic management.

## 6 Conclusions

The work presented in this article corresponds to a real-world experience with an extensive and complex problem, traffic management, where the solution based on the use of structures of problem-solving methods has been shown to be successful. The use of the conceptual level provided by the problem-solving method approach was very useful since it provided an adequate solution to analyze and structure the expertise. The resulting model was designed to be generic, i.e., applicable either partially or completely to other domains different from traffic control where there is a dynamic system that needs to be supervised and controlled by human operators (e.g., a chemical plant, a river basin where floods may produce emergencies, a complex computer network, etc.).

Once the analysis was done at the conceptual level following the problem-solving method approach, it was operationalized by using the corresponding computational resources. For this purpose, it was very useful to use the KSM environment that provided the required methodology to produce the operational version. KSM helped to integrate tasks and methods in efficient and intuitive global modules (the knowledge areas) and provided pre-programmed building blocks

(the primitives of representation) and operational languages (Concel and Link) to build the final version.

We used this decision-support knowledge model to develop two different final applications. First, we developed an application for a traffic network in Madrid. For this goal, the use of a knowledge modeling approach was very useful to facilitate the development of the system and to increase the quality of the final system. Then, we developed a second application for a traffic network in Barcelona (of a larger size). In this case, reusing the generic structured knowledge model was very helpful to guide the knowledge acquisition activity and, consequently, it significantly reduced the required effort for this process. In addition to that, we reused software components making easier the implementation.

Both knowledge-based systems were installed on-line, connected to conventional information systems in the corresponding control centers (in June 1995 in Madrid and July 1996 in Barcelona). However, the recent experience with these systems in the control centers has shown certain difficulties in the integration with the previously established traditional operation methods based on conventional information systems. This has suggested that further work needs to be devoted to develop an adequate and ergonomic human-computer interaction in this type of systems. With this aim, we are currently involved in a research project (FLUIDS, Future Lines on User Interface Decision Support) where an improved human-computer communication is supported by a knowledge architecture in the line of the system described in this article.

## References

Breuker, J. & Van de Velde, W. (1994). CommonKADS Library for Expertise Modelling, Reusable Problem Solving Components. IOS Press, Amsterdam, Oxford, Washington DC.

Chandrasekaran, B. & Johnson, T.R & Smith, J.W. (1992). Task Structure Analysis for Knowledge Modeling. Communications of the ACM, 35 (9), 124-137.

Cuena, J. & Ambrosino, G. & Boero, M. (1992). A General Knowledge-Based Architecture for Traffic Control: The KITS Approach. Proc. International Conference on Artificial Intelligence Applications in Transportation Engineering. San Buenaventura, California.

Cuena, J. & Molina, M. (1994). KSM: An Environment for Knowledge Oriented Design of Applications Using Structured Knowledge Architectures. In "Applications and Impacts. Information Processing'94", Volume 2. K. Brunnstein y E. Raubold (eds.) Elsevier Science B.V. (North-Holland), IFIP.

Cuena, J. & Hernández, J. & Molina, M. (1995). Knowledge-based Models for Adaptive Traffic Management Systems. Transportation Research (Elsevier Science). Part-C. Vol 3. No 5. pp 311-337.

Cuena, J. & Hernández, J. & Molina, M. (1996). Knowledge Oriented Design of an Application for Real Time Management: The TRYS System. Proc. European Conference on Artificial Intelligence (ECAI'96).

Cuena, J. & Molina, M. (1997). KSM: An Environment for Design of Structured Knowledge Models. Chapter of the book "Knowledge-based Systems: Advanced Concepts, Techniques and Applications". S. G. Tzafestas (Ed.). Publisher World Scientific Publishing Company.

Cuena, J. & Hernández, J. (1997). An Exercise of Knowledge Oriented Design: Architecture for Real Time Decision Support Systems. Chapter of the book "Knowledge-based Systems: Advanced Concepts, Techniques and Applications". S. G. Tzafestas (Ed.). Publisher World Scientific Publishing Company.

Deeter, D.L. and Ritchie, S.G. (1993). A Prototype Real-Time Expert System for Surface Street Traffic Management and Control. ASCE, Third International Conference on Applications of Advanced Technologies in Transportation Engineering. Seattle, Washington, USA.

Macintyre, A. (1993). KREST User Manual 2.5. Vrije Universiteit Brussel, AI-lab. Brussels.

McDermott, J. (1988). Preliminary Steps Toward a Taxonomy of Problem Solving Methods. Automating Knowledge Acquisition for Expert Systems, S.Marcus (ed.), Kluwer Academic, Boston.

Molina, M. & Logi, F. & Ritchie, S. & Cuena, J. (1995). An Architecture Integrating Symbolic and Connectionist Models for Traffic Management Decision Support. Proc. VI International Conference on Applications of Advanced Technologies in Transportation Engineering.

Molina, M. & Shahar, Y. & Cuena, J. & Musen, M. A. (1996). A Structure of Problem-solving Methods for Real-time Decision Support: Modeling Approaches Using Protégé-II and KSM. Proc. 10th Knowledge Acquisition for Knowledge-based Systems Workshop. Banff (Canada).

Molina, M. & Gómez, A. & Sierra, J.L. (1997). Reusable Components for Building Conventional and Knowledge-based Systems: The KSM Approach. Proc. 9th International Conference on Software Engineering and Knowledge Engineering SEKE 97. Madrid.

Molina, M., & Sierra, J.L., & Serrano, J.M. (1998). A Language to Formalize and to Operationalize Problem Solving Strategies of Structured Knowledge Models. 8th Workshop on Knowledge Engineering: Methods & Languages, KEML 98. Karlsruhe, Germany.

Musen, M.A. & Gennari, J.H. & Eriksson, H. & Tu, S.W. & Puerta, A. R. (1995). PROTÉGÉ-II: Computer Support For Development Of Intelligent Systems From Libraries of Components. In Proceedings of MEDINFO '95, Eighth World Congress on Medical Informatics, 766–770, Vancouver BC.

Schreiber, G. & Wielinga, B.J & Breuker, J.A. (eds) (1993): KADS: A Principled Approach to Knowledge-based System Development. Knowledge-Based Systems, vol 11. Academic Press. London.

Schreiber, G. & Wielinga, B.J & de Hoog, R., & Akkermans, H. & van de Velde, W. (1994): CommonKADS: A Comprehensive Methodology for KBS Development. IEEE Expert, December, 28-37..

FIGURES

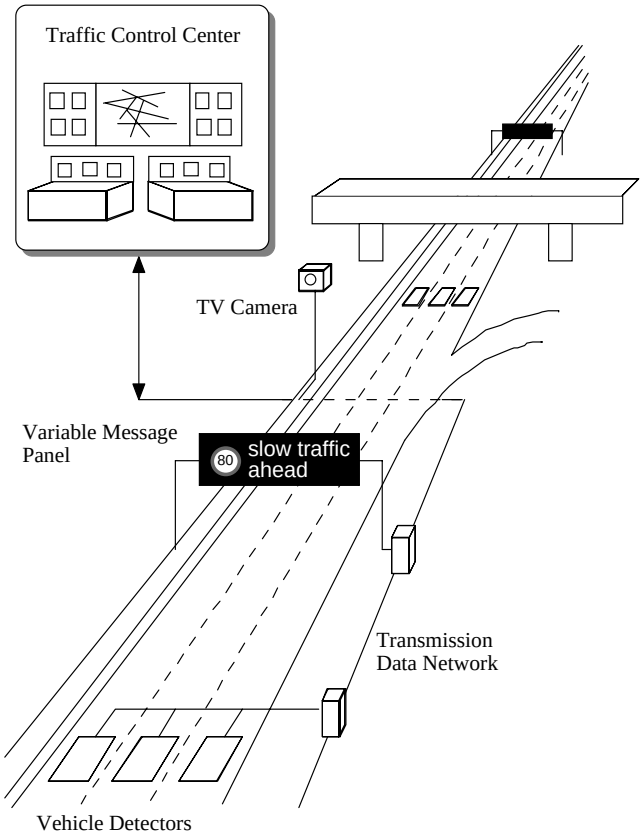


**Fig. 1.** Typical information infrastructure for real-time traffic control.

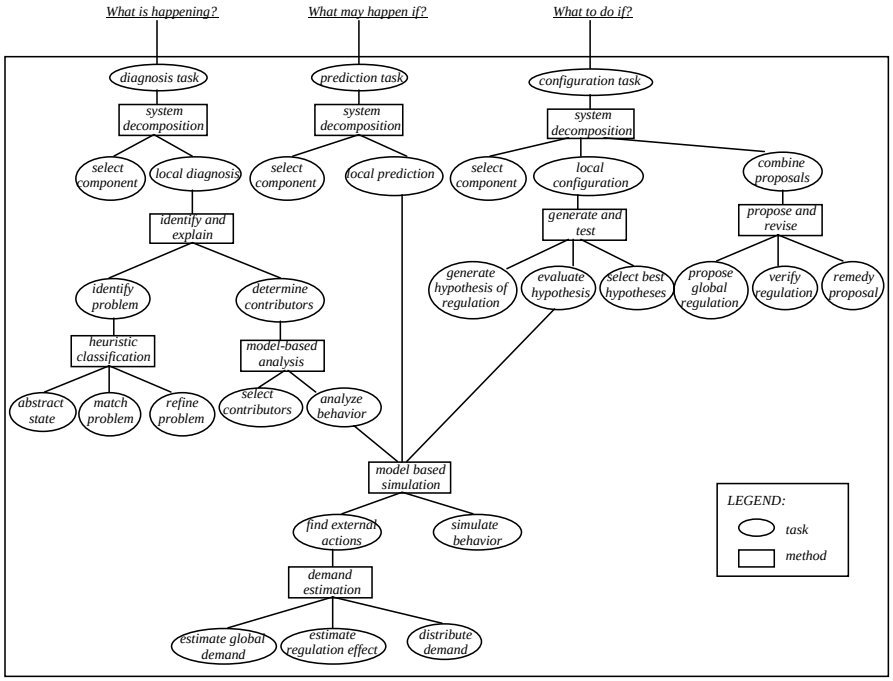


**Fig. 2.** Task-method structure of the knowledge model for real-time decision support.

| Primitive Inferences | Input Roles | Output Roles | Domain Knowledge |
|---|---|---|---|
| select component | system | component | system model |
| abstract state | component<br>observables | symptoms | abstraction functions |
| match problem | component<br>symptoms<br>observables | type of problems | problem scenarios |
| refine problem | component<br>type of problems<br>symptoms<br>observables | problems | system model |
| estimate global demand | component<br>observables<br>time | global demand | demand model |
| estimate regulation effect | component<br>control actions | regulation effects | control actions model |
| distribute demand | component<br>global demand<br>regulation effects | demand | - |
| simulate behavior | component<br>observables<br>problems<br>demand | use of resources<br>problem severity | system model |
| select contributors | component<br>problems<br>use or resources | contributors | system model |
| generate hypothesis of regulation | component<br>symptoms<br>problems<br>current control actions<br>critical contributors | local reg. proposal | control actions model |
| select best hypotheses | component<br>local reg. proposals<br>problem severities | local regulation | - |
| propose global regulation | local regulations | global reg. proposal | - |
| verify regulation | global reg. proposal | violations | compatib. model |
| remedy proposal | global reg. proposal<br>violations | global solution | fixes model |

**Fig.3.** Primitive inferences of the knowledge model

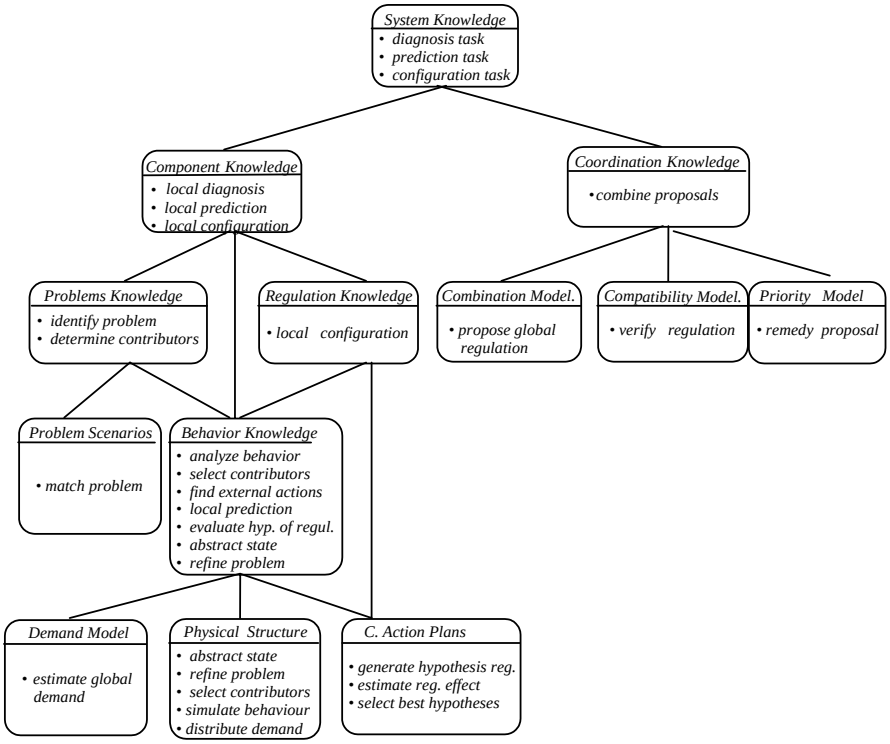


**Fig. 4.** The knowledge-area structure of the decision support knowledge model.

```
METHOD  propose-and-revise regulation
        INPUT  local regulations
        OUTPUT global regulation
DATA FLOW
        (combination model) propose global regulation
               INPUT local regulations, global regulation
               OUTPUT global regulation
        (compatibility model) verify regulation
               INPUT global regulation
               OUTPUT violations
        (priority model) remedy proposal
               INPUT global regulation, violations
               OUTPUT global regulation
CONTROL FLOW
        START
        ->  (combination model) propose global regulation,
            (compatibility model) verify regulation.
        (compatibility model) verify regulation
        IS incompatible
        ->  (priority model) remedy proposal,
            (combination model) propose global regulation,
            (compatibility model) verify regulation.
        (compatibility model) verify regulation IS compatible
        -> END.
```

**Fig. 5.** Example of problem-solving strategy formulated using the Link language

```
Concept system subclass of object.
Attributes:
      components (instance of component).

Concept component subclass of object.
Attributes:
      resource (instance of element).

Concept resource subclass of object.
Attributes:
      state (symbol),
      capacity (number),
      detectors (instance of detector).

Concept environment subclass of object.
Attributes:
      external actions (instance of action).

Concept control action subclass of action.
Attributes:
      value (symbol),
      effector (instance of effector).
...
```

**Fig. 6.** Partial example of conceptual vocabulary formulated using the Concel language

.

| *Primary Knowledge Areas* | Problem Scenarios | Demand Model | Physical Structure | Control Action Plans | Combination Model | Compatib. Model | Priority Model |
|---|---|---|---|---|---|---|---|
| *Primitives of Representation* | Frame-based primitive | Frame-based primitive | Traffic network simulator | Triplets <A,C,E> | Combin. Procedure | Rule-based primitive | Rule-based primitive |

**Fig. 7.** Computational support of primary knowledge areas by using primitives of representation.

```
FRAME traffic jam caused by on-ramp
VARIABLES DEFINITION
        (<link>)
          type =             on-ramp link,
          previous section = <previous>,
          next section =     <next>,
          ramp section =     <ramp>,
DESCRIPTION
        (<previous>)
          speed =            low             [a],
          occupancy =        high            [b],
          saturation =       low             [c],
        (<ramp>)
          occupancy =        [medium, high]  [d],
          saturation =       low             [e],
        (<next>)
          speed =            medium          [f],
          occupancy =        [low, medium]   [g],
          saturation =       high            [h],
        (critical section)
          location =         <next>          [i],
          state =            overflow[j],
          category =         problem         [k],
          excess =
           EXP(dem(<next>) - capac(<next>)) [l],

RELEVANCE OF CHARACTERISTICS
        (a; b), d, h  -> 100%.
        i, j          -> 100%.
        (a; b), h     -> 80%.
        d, h          -> 70%.
        (a;b)         -> 50%.
```

**Fig. 8.** Example of frame representing a type of traffic problem.

```
FRAME morning peak TYPE week day pattern
   type of day:                        week day,
   temporal interval:                  6:30 - 9:30,
   ....
   Mataro -> Badalona Norte:           1000 veh/h,
   Mataro -> Badalona Centro Sur:      1000 veh/h,
   Mataro -> Zona Urbana:              2000 veh/h,
   Mataro -> Glories:                  2000 veh/h,
   NII -> Badalona Norte:              500 veh/h,
   NII -> Badalona Centro Sur:         500 veh/h,
   ....
   Badalona Centro Sur -> Glories      1000 veh/h.
```

**Fig. 9.** Partial example of frame representing a pattern of traffic demand.

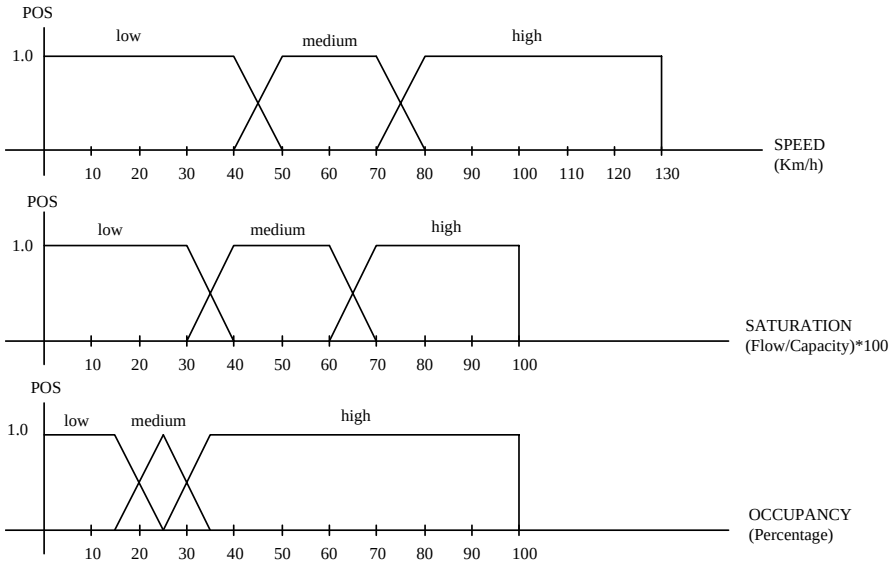


**Fig. 10.** Possibility functions to abstract the traffic state from numerical data.

```
CONTROL ACTION PLAN warning about incident at A17
    CONTROL ACTIONS
        (B20/3-PIV-1) message = INCIDENT AT A17
        (B20/6-PIV-1) message = INCIDENT AT A17
        (A17/8-PIV-1) message = INCIDENT AHEAD
    STATE CONDITIONS
        (A17 Girona PK5300) state = free AND
        (A17 Girona in Montcada) state = incident AND
        (A17 Girona at Montmelo exit) state = free
    EXPECTED EFFECT
        (Meridiana -> A18)
          [80, 90] by A17 road
          [10, 20] by secondary road
        (Trinidad -> A18)
          [60, 80] by A17 road
          [20, 40] by secondary road
```

**Fig. 11.** Example of knowledge representation for a control action plan.

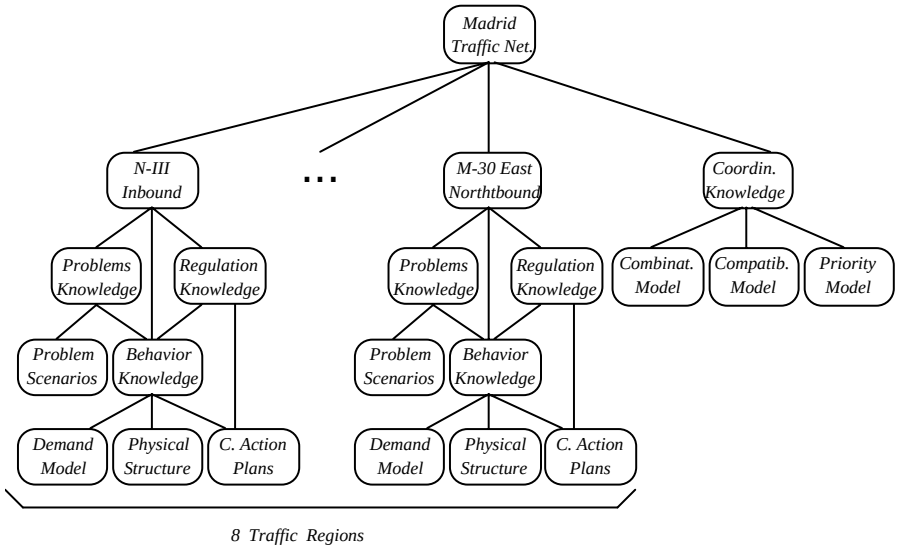


**Fig. 12.** A domain knowledge-area perspective for the particular application of Madrid.

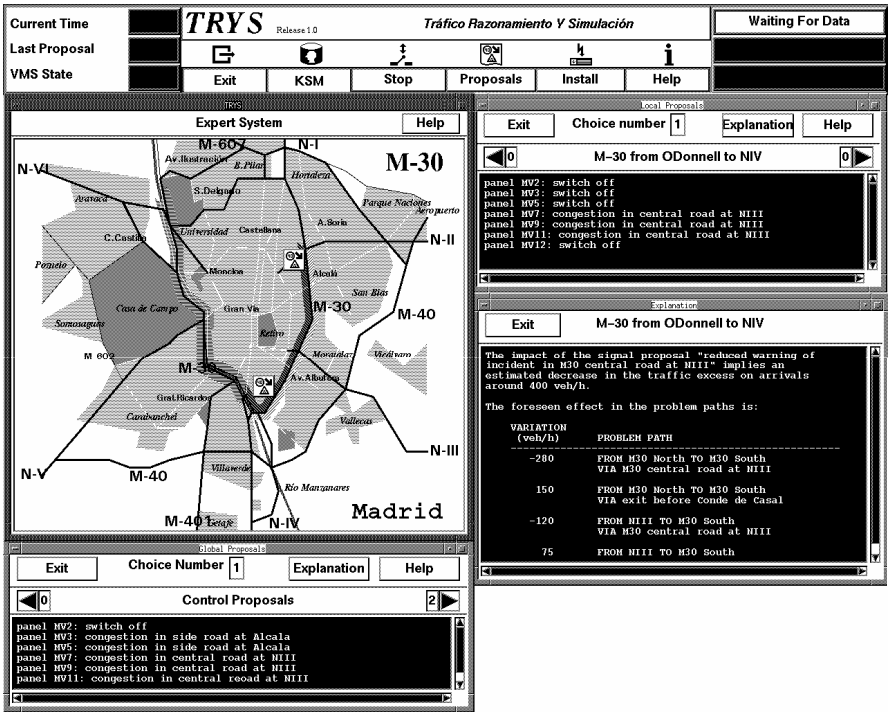


**Fig. 13.** Screen example of the user interface of the application for Madrid.

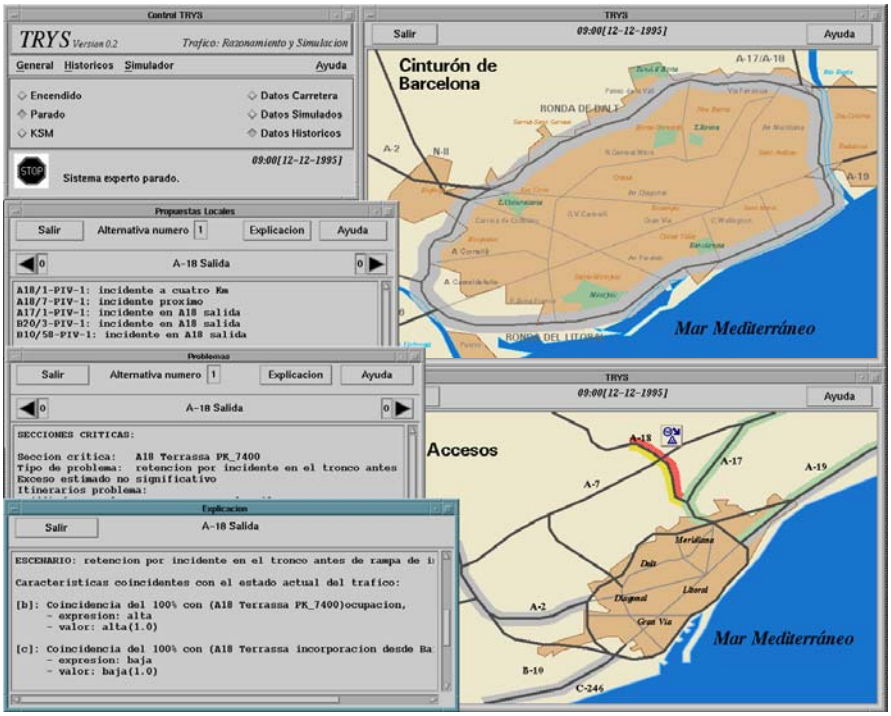


**Fig. 14.** Screen example of the user interface of the application for Barcelona.