2000

An Intelligent Interactive Knowledge Model for Decision Support in Real Time Traffic Management

Josefa Z. Hernandez
Technical University of Madrid

Juan M. Serrano
Universidad Rey Juan Carlos

search

An Intelligent Interactive Knowledge Model for Decision Support in Real-Time Traffic Management

Josefa Z. Hernández
Department of Artificial Intelligence
Technical University of Madrid
Campus de Montegancedo s/n
Boadilla del Monte, 28660 Madrid, Spain

Juan M. Serrano
ESCET
Rey Juan Carlos University
Independencia 12
Móstoles, 28931 Madrid, Spain

Abstract- **This paper proposes the use of advanced knowledge models to support real time decision for management problems as an adequate response to the current needs and technology. The new conditions for human operation created by the telematics technology are discussed and a general architecture using knowledge modelling techniques is proposed. Then, the application of the approach to support real time management of the private traffic in the city of Turin is described.**

I. Introduction

The development of telematics systems in the last decade has made feasible to build up applications integrating sensors, communications and real time data bases to provide raw information about the state of a natural or artificial installation such as a chemical plant, a traffic network or a watershed. In this way, a great amount of information is available that should be used to improve the management of the system, which generally means to take the adequate decisions in the right moment. In complex installations, evaluating decisions requires in many cases to use complex models integrated with human reasoning. So, a new type of support has to be enabled to ensure that both theoretical and common sense knowledge are adequately available to the human operators responsible for the installation management.

In this context, the European Commission research project FLUIDS (Future Lines of User Interface Decision Support) proposes a design methodology for the development of advanced decision support systems (DSS's). The methodology makes a strong commitment towards the role that human-computer interaction features play in the final specification of such kind of systems, and defines a knowledge-based architecture aiming to support DSS's with higher interaction capabilities. In addition, three sample demonstrators were implemented to show the feasibility of the design approach in the traffic management domain (though the methodology is not constrained to this particular domain).

First, general human-computer interaction requirements for DSS's are identified. In particular, a domain-independent set of questions relevant in the management process followed by the responsible people is defined. Next, the proposed knowledge-based architecture for a DSS capable to support the required user-system dialogue is described whose main components are a Conversation Manager, a Problem Solving Environment and a Presentation Manager. This paper is focused on the first two components, stating the need to consider a collection of alternative problem-solving methods capable to flexibly support the interaction requirements. The adequate problem solving methods for generating an answer are dynamically selected by a metalevel reflective model included in the Problem Solving Environment. Relevant aspects of the design process of these architecture components are considered. In addition, the KSM tool [1] is proposed as an adequate framework to implement those elements. Finally, the application of this approach is illustrated with the FLUIDS/CRITIC application, a DSS for private traffic management in the city centre of Turin.

II. On Human-Computer Interaction in Decision Support Systems

This section describes the main characteristics of the interaction between a DSS and its potential users from two perspectives: from a strategic view, the main goal and principles to be satisfied by the interaction are sketched; and from a more specific point of view, some common generic questions for any kind of DSS are proposed. In this way, some guidelines, concerning the interaction capabilities of a DSS system, to be supported by any development methodology of such kind of systems are established.

With regard to the strategic level, the supporting role of the application should be emphasised. It means that its main goal is not to fully achieve the management of the system by itself, but to support the operators and engineers to get their management goals more efficiently. This abstract principle can be further elaborated on the following points:

- From the perspective of a co-operative working model between the system and the operator, the system should act as an *assistant* that keeps the operator informed about the behaviour of the monitored system and suggests the more suitable control actions to be taken according to the observed situation, but leaving the final responsibility on the management to the operators.
- In order to achieve the highest possible level of *mutual intelligibility*, the system should try to emulate as much as possible the reasoning processes and conceptual terminology of the control operators providing in this way a common decision language. In addition, due to the dynamic character of the domains where DSS are usually installed, this model of the operator's expertise should be easily accessible and modifiable.
- Since the operators are the final responsibles for the impact produced by the control actions, the system must

provide *justifications* of the suggested actions explaining the reasoning line followed to reach such conclusions. In addition, in order to take into account the inherent uncertainty pervading the management process, several alternative answers should be provided when possible. Similarly, the user should be given the possibility to obtain new alternative answers by providing different values for the underlying default hypothesis. In this way the flexibility and confidence of the operators in the system conclusions may be improved.

- Furthermore, due to the usual time constraints to be taken into account when making the decisions, it is necessary to attain a *fluid* communication between the operators and the system understood as the transmission of the right information in the right moment and mode of presentation. In this sense, the system should be capable of *adapting* its answers to the characteristics of the user and the evolution of the dialogue.

More specifically, some common features of the interaction with a DSS at the semantic level (i.e. concerning the kind of information which should be communicated), can also be considered by analysing the reasoning process followed by the expert people in the problem domain. Basically, this process is oriented towards the elaboration of action plans to control the system, so the users may want to ask the application what kind of control actions are recommended. In addition, the plans are elaborated on the basis of the problems currently happening, together with their diagnosis. The foreseeable problems may be relevant as well, together with the variables affecting the state of the system: environmental and control actions. Thus, it should be possible for the user to ask for the current and foreseeable problems detected by the application. So, the following classes of questions should be considered:

- *What is happening* questions: i.e. detection of significant current and recent events or problems. If the knowledge model is on-line connected with a telematics system providing basic state information abnormal or undesired situations can be detected and presented to the operator, who may ask for additional derived questions and explanations, e.g. with a *why did the problems happen* question.
- *What may happen* questions. This type of questions may be answered using the behaviour models assuming the current scenarios of external actions. These questions may be also conditionally formulated in the form *What may happen if* <*hypotheses*>, assuming different scenarios of external actions and management decisions of the system, that may inform the operator about the potential short-term evolution of the situation.
- *What should be done* questions, to be answered using knowledge models for planning control actions oriented to solve or alleviate the problems previously detected. These questions may also have a conditional expression: *What to do if* <*hypotheses*>, implicitly assuming in that way different short-term scenarios of the monitored installation.

III. An Architecture for Interactive Decision-Support Systems

The design of an generic architecture for DSS's capable to support a dialogue-based user-system interaction in the previous terms can be faced using two different kind of technologies taken from the Knowledge Engineering and Artificial Intelligence fields:

- On the one hand, the recent knowledge modelling techniques provide the facilities to build highly structured knowledge architectures where the different problem solving methods applied by the expert people can be explicitly expressed in a declarative fashion allowing to incrementally improve the expertise model by means of an open architecture [1], [2]. In this way, the requirement for a common decision language would be achieved, ensuring that both the domain experts and the application reason in similar and understandable ways. In addition, given the high level of abstraction of the problem-solving approach and its non-deterministic nature, meaningful explanations and alternative answers can be provided as well. This last characteristic can be also supported by the possibility of providing several alternative problem-solving methods for some given task. Thus, the people responsible for the management of the system can assume the DSS's proposals, as they are able to fully understand the reasoning line followed by the application.
- On the other hand, user interface's development techniques have evolved to intelligent planners of co-ordinated multimedia presentations supported by knowledge-based architectures using specialised knowledge on different aspects of the presentations design [3], [4], [5]. In this way, the application can significantly improve its expressive capabilities, dynamically configuring the presentation of the information to be conveyed according to the user's preferences and characteristics.

In our view, a reasonable approach for a general decision support architecture is to combine both kind of technologies (mainly due to the fact that sophisticated presentations can only be efficient if the content of the information they present is relevant enough). However, although both solutions give support to advanced user-system interaction models, in this paper we focus our interest in the second one, to support the kind of conversation required in decision support scenarios in line with the interaction requirements outlined before.

Starting from the classes of questions previously mentioned a relationship between these queries and a knowledge-based component of the target architecture may be defined as follows. Every question is directly related to a class of problem or task that needs to be solved in order to

obtain an answer. The solution of any of these tasks may be in principle obtained by different problem solving methods, offering in this way the possibility of getting different answers for the same question. In its turn, the performance of the selected method may require the solution of several subtasks that again may be realised by different methods and so on, defining the problem solving structure that describes the reasoning process required to generate the answer. In this way, every relevant question for a DSS is related to a particular hierarchical structure of tasks-methods-subtasks. The major tasks associated to questions that the designer of the application may consider would be problem detection and problem diagnosis (for *what is happening* questions), prediction (for *what may happen* questions), and evaluation of control actions or suggestion of plans to solve problems (for *what should be done* questions). Thus, this kind of interaction requirements acts as a heuristic to guide the knowledge engineer in the design of a DSS in a specific domain.

Given that several methods may be used to solve the same task (e.g. heuristic vs. model-based methods for the classification task), the dynamic selection of the most adequate method to perform a task should be the result of a *reflective* process on the problem solving capabilities available. This reflection has to consider the information context when the query is made. However, in order to achieve higher levels of user-system interaction, the selection of the appropriate problem solving methods should be also guided by criteria based on conversational aspects [6]. These aspects refer interaction features such as the level of abstraction and/or precision required in the answer, the type of user, time constraints in the answer generation, etc.. Actually, several kind of users can usually be considered in decision support scenarios ranging from low-level operators specialised in concrete aspects of the system (e.g., the operator of reservoir gates), which require information on a high level of detail, to high-level managers that are interested in global evaluation and strategic aspects. Consequently, different methods for classification or planning would be selected to generate the proper answer depending on the user needs and the evolution of the dialogue. As this reflective knowledge consists on knowledge about the problem solving knowledge/methods available it defines a metalevel knowledge model.

From the above description, an architecture supported by three main components may be defined (see Fig. 1):

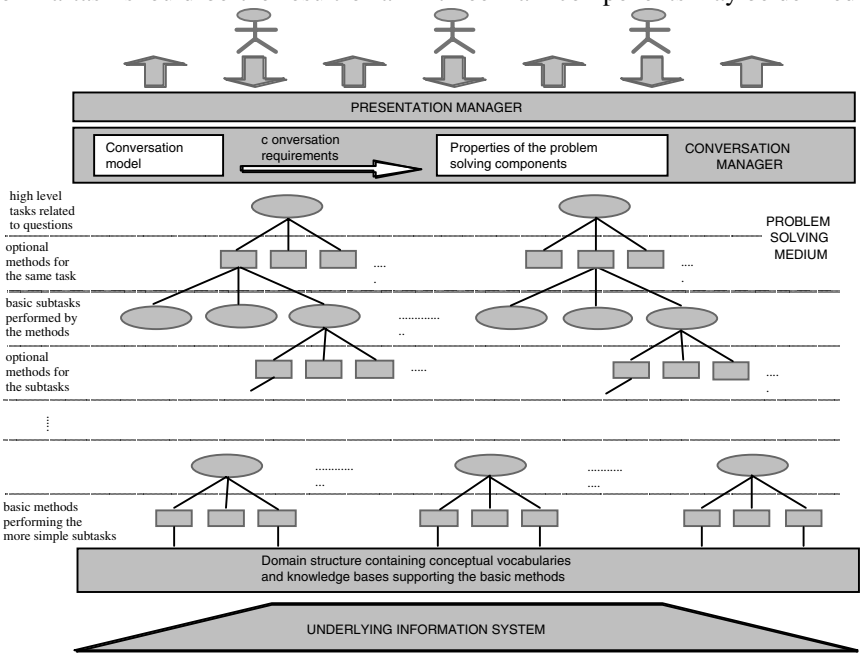

Figure 1: General architecture of the interactive knowledge model for decision support

- a *Presentation Manager* specialised in the input-output activities of the system,
- a *Conversation Manager* which plays the role of an intelligent interpreter between the user needs and the system capabilities. It infers the interaction features, relevant for the user and the current state of the dialogue, from the metalevel model deciding on the fly the elements of the Problem Solving Medium that should be applied to obtain the required answers.
- a *Problem Solving Medium* that contains a structured collection of automatic tools implementing the problem solving functionalities of the system usable to generate the answers.

This architecture and the framework for DSS proposed in [7] share some common features: a presentation component is regarded in both cases, and the *Language* system could be related to the *Conversation Manager*. On the other hand, the functionality of the *Knowledge* and *Problem Processing* systems may be covered by the *Problem Solving Medium.* The differences are related to the emphasis given in the

proposed architecture on the metalevel model, and its impat on human-interaction features.

In summary, the reasoning line followed to support a user-system conversation may be the following (see Fig. 2, below):

- the *Presentation Manager* translates the question formulated by the user to the problem solving language, that means, to the generic task that needs to be solved in order to get the answer.
- the *Conversation Manager* makes an analysis of the recent dialogue with the user and defines a set of conversation requirements that should guide the selection of the methods in the Problem Solving Medium.
- attending to these requirements, a top-down selection process is applied in the *Problem Solving Medium* to define an inference structure whose execution generates the answer.
- finally, the *Presentation Manager* takes this answer and elaborates an adequate presentation, using a combination of different modalities (texts, graphics, etc.)

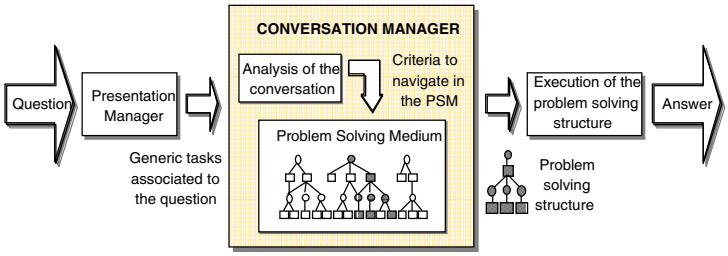

Figure 2: Reasoning line followed to generate the answers to the user questions

IV. Design Methodology

Some considerations on the design process and tools for the development of a DSS supported by the previous architecture are now outlined. The first step is the design of the conversation model in terms of the main questions and interaction features that the DSS should take into account. In this way, human-computer interaction requirements establish the basis for the design of the next components. The second step is the design of the problem-solving environment, which can be divided into two different parts: the design of the metalevel reasoning model and the Problem Solving Medium. The third step, which can be actually run in parallel with the second one and is not described, is the design of the presentation model. Lastly, the KSM tool is proposed as an adequate means to support the implementation of the designed components.

A. *Design of the Conversation Model*

First, the specific instances of the generic classes of questions considered for decision-support need to be identified, according to the characteristics of the problem domain and the requirements of the potential users. In order to get a full description of the possible questions and answers, the conceptual vocabularies of basic variables and their corresponding domain values should be defined. These include: (i) the basic or raw variables characterising the state of the system; (ii) the complex evaluation variables describing the possible situations of the system; (iii) the set of possible control decisions, from those that specify general strategies or control policies to the detailed ones defined as the aggregation of different steps characterised by simple control actions; and (iv) the set of hypothesis that define alternative formulations of the questions and characterise different features of the environment affecting the state of the system.

Then, a conversation may be structured as a sequence of questions, answers and explanations expressed using the previous definitions. However, in order to adequately characterise the communicative goals of the user, the relevant interaction features concerning the context of the interaction should be defined as well. As commented in the last section, these characteristics of the interaction will affect the selection of the problem-solving methods that will provide the required output. In this sense, some domain-independent features can be identified that are potentially useful in any kind of DSS, thus providing again some heuristics for the design of this component of the application. These include, for instance, the required level of abstraction and level of detail for the answer, the requirement for further explanations, the type of user, and the possible time constraints imposed in the answer generation. Some of these features could be inferred from the context of the conversation, in such a way that the user may not need to explicitly declare his/her communicative intentions.

B. *Design of the Problem Solving Medium*

This part of the design applies conventional techniques of knowledge acquisition that take advantage of the knowledge level structuring paradigms. The design of this environment may require the following steps:

1) Identification of the Problem-Solving Medium components

First, the key element in the specification of this stage of analysis is the identification, for every type of question, of the main tasks to be performed in order to obtain an answer. This relationship between questions and tasks may be deduced from the parsing of the questions structure. According to the classes of questions previously identified for any kind of DSS, classification, predictive and planning tasks may be considered, respectively.

Even though in the initial version of any system usually only one problem solver may be provided for every task, the experience in the use of the model and a deeper understanding of the problem domain may lead to introduce additional methods to solve the same kind of problems. Thus, for every major task, the alternative problem solving

approaches that could be applied should be defined, obtaining in this way several supporting subtasks for each problem-solving method. For every subtask, the previous step should be applied, until a tree-like structure of alternative problem solvers for every task and subtask is identified. For instance, several versions of problem solvers may be designed for efficiency reasons with different conceptual granularity providing in this way different levels of accuracy and detail in the answers.

Finally, the basic inference methods supporting the bottom level subtasks should be described on the basis of their corresponding domain model, i.e. in terms of the conceptual vocabulary and knowledge bases. This knowledge and the task-method-subtask structure in general, may be obtained from different knowledge sources, whose availability depends on the nature of the problem domain, such as the following ones:

- The expertise of the people responsible of performing the activities in the problem domain used to support the decision making.
- Data mining processes applied to databases with registered information about the behaviour of the system that provide more abstract knowledge.
- Mathematical or physical theories that support the dynamics of the system.
- Common sense knowledge derived from the experience with previous versions of the application that makes possible to update the knowledge model.

2) Specification of the metalevel model

Given that the use of the last components is oriented to configure reasoning structures capable to satisfy users queries, it is necessary to characterise the peculiarities of the methods that would make them eligible for being part of these structures. The metalevel model includes the criteria to dynamically select the problem solving methods that will participate in the generation of an answer. These criteria may be defined according to the interaction-related properties of the methods and the state of the dialogue, making possible the selection of the methods that satisfy as many requirements as possible.

For instance, if a *what should be done?* question needs to be answered with a high level of abstraction, a low level of assistance and with the possibility of providing explanations, then the Conversation Manager may decide that the most appropriate way to answer this question is to use a heuristic planner that uses a forward chaining procedure to select general strategies for solving problems.

C. Operationalisation of the Conversation Model and Problem Solving Medium

The two main components of the proposed architecture were implemented using the KSM tool [1]. The KSM software environment supports the definition and implementation of complex structured knowledge models by means of three main elements: the *CONCEL* language (that supports the specification of the conceptual vocabularies of the model), the *LINK* language [8] (used in the specification of the problem-solving methods), and a set of *primitives of representation* (that provide a symbolic representation language and related inference mechanisms, such as frames, rules, bayesian networks, etc., to define and use basic knowledge bases). Besides the task-method-subtask structure, the knowledge model can be structured on the basis of the more aggregated concept of *knowledge area*, which groups tasks related to some particular body of knowledge. A *primary knowledge area* is one that can not be further decomposed in new subareas, and so groups tasks implemented with the basic inference mechanisms of a primitive of representation, which operate on a knowledge base implemented with the representation language of the primitive. An important characteristic of the KSM framework in the context of the DSS development is that provides the experts in the DSS domain the possibility to inspect and modify the knowledge stored in the model by their own. This is possible since the languages used to express this knowledge (CONCEL, LINK, those offered by different types of primitives) allow expressing the domain concepts and knowledge in a more declarative way.

Concerning the Conversation Model, the specific terminology and interaction features of the questions identified may be declared in the conceptual vocabularies of the model. Those interaction features that can be inferred from the context of the dialogue can be obtained using the knowledge, expressed in frames for instance, specified in (possibly several) primary knowledge areas. The task(s) necessary to be executed to answer a given question could be simply specified with the LINK method of the corresponding *answer question* task.

With regard to the metalevel component of the Conversation Model, the main goal is the selection of the more suitable methods to satisfy a given collection of interaction features. In this context, the reflective capabilities of the LINK language are relevant since the selection of the right method requires to reason on the problem-solving capabilities available. The relationship between desirable interaction features and method's characteristics can be represented in primary knowledge areas. A possible strategy is to dynamically configure the specific problem-solving structure, as the tasks are needed to be launched. In this way, each time a task is fired the outcome of the reflective method associated to the task provides the problem-solving method that best fit the current interaction requirements. This process may be repeated for the execution of every subtask of the selected problem-solving method.

V. Application in the Traffic Management Domain

In order to illustrate the proposal described in this paper, this section presents an example of DSS in the domain of private traffic management in the city centre of Turin. This system was developed within the European Commission financed research project FLUIDS. First, the particular needs

of human-computer interaction in this domain are outlined and, then the solution provided by the proposed architecture is described focusing the attention in the problem-solving model, i.e. the Conversation Manager and the Problem Solving Medium. Information regarding the Presentation Manager component may be found in the FLUIDS project web site [9].

At present, the management of private traffic in the city centre of Turin is performed by means of the UTOPIA control system [10]. UTOPIA provides general control directives to a collection of subsystems, called SPOT units, responsible of automatically and dynamically regulate the control devices in the intersections of the traffic network. The role of the FLUIDS system in this context was to help an operator in supervising not only the state of the traffic but also the performance of these SPOT units. Currently, the control system operator monitors the functioning of the SPOT units and, when it is required, manually adjusts some of its internal parameters in order to ensure an adequate performance.

The FLUIDS tool may automatically supervise the state of the traffic network and the SPOT units and alert the operator about the presence of current or foreseeable problems, the possible causes of these problems, and adequate control action proposals oriented to solve these problems. Therefore, the FLUIDS tool for private traffic management plays a role of an automatic assistant for supervising the state of the traffic network and the operation of the SPOT units, working as a critic that evaluates its performance and suggests modifications to improve their operation. For this reason it was named the FLUIDS/CRITIC system.

A. *Conversation Manager*

According to the general goals of the FLUIDS system for the Private Traffic System previously described, and the generic types of questions described above, the operator-system conversation model needs to support the following specific classes of questions:

- *What is happening* questions related to the state of the traffic network at different levels: areas, links and intersections. These questions ask for the presence of congestion problems, incidents and/or excess demand situations. In addition, this kind of question can refer to the evaluation of the current performance of the control system as well. These evaluations can be also useful for the diagnosis of the congestion or excess demand problems. For instance, an unbalanced control policy may not take into account the high incoming flow of an incoming link in the green split of the control cycle.
- *Why did it happen* questions asking for the diagnosis of causes which explain the problems currently happening. The goal in this case is to find the causes of abnormal situations. In general, causes can be internal to the SPOT application (a low-level performance of the control system, which usually means that there are some internal parameters that are not well tuned) or external (e.g., problems of communications, detectors, etc.). Other possible causes may be independent of the UTOPIA control system, such as incidents or excess demand traffic problems.
- *What may happen* questions looking for estimations of the potential short term evolution of the traffic network under specific environment conditions (traffic demand and turning ratios) or as an impact of the application of particular a control action proposed by the SPOT units in the past (modification of the nominal flow value for a specific carriageway).
- *What to do* questions asking for suggestions of control actions that improve the current or foreseeable situation under different scenarios of resources availability. For instance, if a congestion problem is caused by a control problem the application may provide recommendations oriented to change or to reset particular numerical parameters of SPOT, or actions that isolate a particular malfunctioning device (e.g., a detector).

Furthermore, the previous questions may be followed by *Why* type questions, asking for explanations which support the results supplied by the application.

B. *The Problem-Solving Components*

In the domain of Private Traffic Management the following main basic tasks and methods may be considered:

- *Classification task*. The input to this task are raw data obtained from the telematics network. The output consists of the problems detected in the traffic network. Two methods can be considered: (i) heuristic classification and (ii) lookahead classification. For the heuristic classification method three subtasks can be defined: *abstract data*, which receives the real time data sent by the information system, mainly consisting of vehicle counts, speeds, queues, saturation flow, turning percentages, and signal timing values. From these data, it generates abstract and aggregated measures such as: the estimated delay, the capacity of the links, etc. *Validate data* that is described next. *Match problems*: The pattern of congestion of the links, intersections and the whole area network are matched against the current traffic conditions. For the *lookahead classification* method, the same subtasks of the heuristic method are included but also a new subtask for the detection of potential short-term problems.

- *Validation task*. The real time data supplied by the information system, and the values returned by the abstract data task, are analysed in order to identify their validity. Two different methods may be considered: (i) (heuristic) local validation and (ii) model-based validation. The *local validation* method uses a set of heuristic rules to identify those variables with different values for the same concept, and hence possibly incorrect. These rules are mainly based on the difference between the estimated and the historical or optimum values. For the *model-based validation* method a local validation task is initially solved. Then, a consistency validation task looks for inconsistencies between the values of the variables describing the state of the traffic situation.
- *Diagnose traffic problems task*. Given a congested link, and the past evolution of the system, this task returns as output the possible causes of the current congestion. In general, the causes can be local to the system (e.g. an incident in some downstream link) or external (an excess demand). Two methods can be considered: (i) classificative diagnosis, and (ii) model-based diagnosis. Both methods differ in the depth of the knowledge used: a set of prototypical evolutions of the traffic state leading to congestion problems, for the first one; and a traffic behaviour model, for the second one.
- *Diagnose control problems task*. This task diagnoses the causes of the problems in the active control applied by the SPOT units: an unbalanced signal plan, or a significant difference between the estimated and expected delay. The output is the quality level of the input data used by the automatic control system: estimated data that represent the current state of the intersection, nominal data that model the topology and default traffic behaviour, or control parameters specified by the area control module. The task is solved by a heuristic method that reflects the expertise of the operators in the control centre of Turin.
- *Manage traffic signals task*. The goal of this task is to modify the unbalanced traffic signal control of the intersections. The input to the task is the description, diagnosis and forecast of the current situation. A *generate & test* method is applied. The generate traffic signal plan returns a modification of the signal timing that distributes the green time of the control cycle proportionally to the index of saturation of the critical links. A knowledge base of heuristic rules is used. The next test stage of the method perform two subtasks: predict evolution and accept signal proposal, which return an assessment of the control proposal in terms of its impact in the short-term evolution of the traffic conditions.
- *Manage SPOT control units task*. The diagnosis and description of the low performance of the control units at intersections is given as input. A planning method is used. The method applies a forward reasoning method to a knowledge base composed of heuristic rules. The method proposes different modifications to the nominal parameters of the SPOT unit (saturation flow, turning percentage, flow, etc.) and different warnings to the maintenance groups (check loop detectors, communication line, etc.).

```
TASK SPECIALIST: validate data
METHODS:   heuristic validation &
                model-based validation
SELECTION KNOWLEDGE:
time constraint = no AND flexibility = no AND
possibility of additional questions = yes AND
recency constraint = yes
          -> model-based validation

time constraint = yes AND flexibility = yes AND
abstraction level = high
          -> heuristic validation
METHODS DECOMPOSITION:
heuristic validation:      local validation
model-based validation:  local validation,
                              global validation
```

Figure 3: Metalevel knowledge of the validation task

C. *Metalevel Issues*

The metalevel knowledge consists of frames associated to each task that define the interaction parameters to be considered for each of the alternative methods associated to the task. For instance, the frame in Fig. 3 is used to select the more convenient method for the validation task.

The local validation method is a heuristic method that only uses the difference with respect to the average or expected value to conclude that a given measure is incorrect. So, in case that the answer is quickly required, with no matter on the quality of the input data (since the analysis is not very strict) nor the level of detail of the answer, this method may be preferred. But, if there is no problem with the time of response and an in-depth analysis is required, the model-based method should be selected. In addition, more detailed explanations could be given with the answer.

D. *Implementation Issues*

This section presents a brief overview on the main implementation issues concerning the problem-solving model of the FLUIDS/CRITIC system. Figure 4 shows the knowledge model developed with the KSM tool.

The entire knowledge for decision support is represented by one knowledge area called *UTC* (urban traffic control knowledge). The tasks for answering *what is happening*, *what to do*, etc., questions are associated with it. This knowledge area is in turn decomposed into four major knowledge areas: classification, validation, diagnosis and control. Each knowledge area includes a set of tasks that can be solved using the corresponding type of knowledge. In some cases these tasks can be performed by several problem-solving methods, using knowledge of the sub-areas in which are decomposed. For instance, the *validation knowledge* area is decomposed into the *local validation* and *consistency knowledge* areas, which support the two different methods available to perform the validation task. The *reflective validation knowledge* area provides the metalevel knowledge

required to select the right method according to the interaction features established.

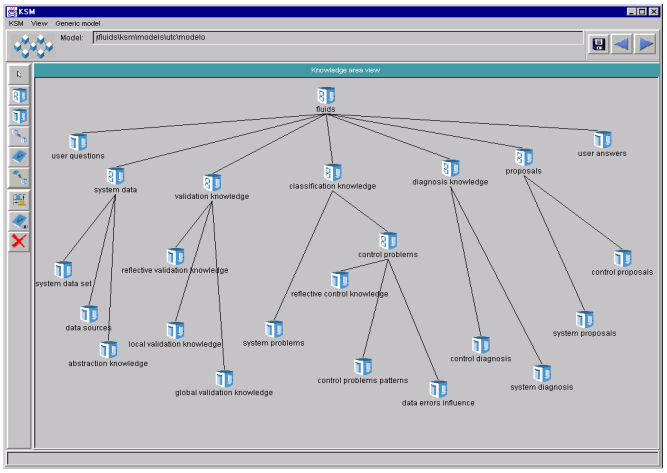

Figure 4: Problem-solving model of the FLUIDS/CRITIC

Regarding the basic knowledge areas, the problematic situations like congestion, incidents, and so on, are represented with frames, and a pattern-matching procedure is used as the basic inference mechanism. Other primary knowledge areas such as the *control proposals* use rules instead. In addition, this type of knowledge areas has conceptual vocabularies associated, defining the components of the topology (e.g. intersections, carriageways, links, etc.) and their basic attributes (e.g. delay, number of stops for links). A primary knowledge unit does not have to be a knowledge-based module. It may be a neural network, a conventional database or even a conventional program with an algorithmic approach. This is the case of the *utopia system data* area, which is used to obtain the raw data collected by the telematics network installed in the traffic network.

Last, figure 5 shows the main window of the FLUIDS/CRITIC system, with the presentation manager's output to a question over the current situation of a particular area.

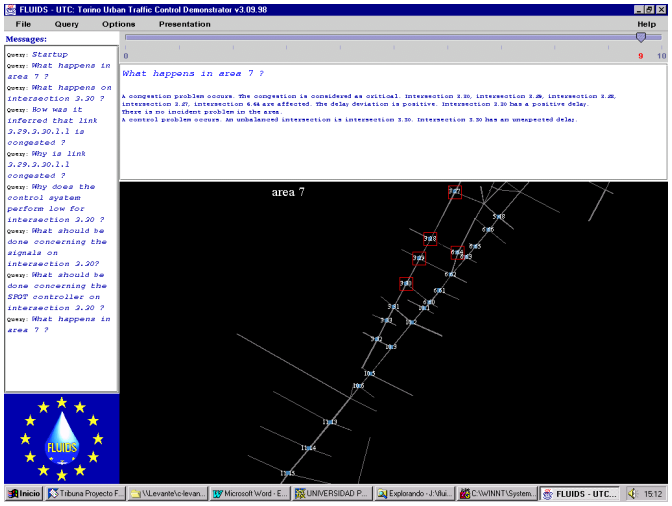

Figure 5: Main window of the FLUIDS/CRITIC system

VI. Conclusions

In summary, the current generation of telematics applications for on-line traffic management requires an advanced operation scenario that may be supported by intelligent tools capable to provide a flexible and adaptive communication between systems and operators. This type of communication may be based on human-computer dialogues where the system acts as an assistant to support the operator decisions. The complexity required by these dialogues suggests the use of recent results in the field of intelligent user interfaces. According to this, this paper proposes a knowledge model that supports this type of interaction whose basic idea lies in the use of a reflective architecture. Two main components are distinguished: a *meta-level component* whose goal is to decide the appropriate way to solve problems according to the needs of the conversation, included in the Conversation Manager, and an *object-level component* composed of a structured collection of problem solvers responsible of producing the corresponding answers, included in the Problem Solving Medium. The experience with this approach in the FLUIDS project allows considering the proposal presented in this paper an appropriate technical answer to the needs of control rooms on real time transport management.

References

[1] J. Cuena and M. Molina, "KSM: An environment for design of structured models", *Knowledge Based Systems: Advanced Concepts, Techniques & Applications*, pp. 497-524, World Scientific Publishing, 1997.

[2] B.J. Wielinga, A.T. Schreiber and J.A. Breuker, "KADS: A modeling approach to knowledge engineering", *Knowledge Acquisition*, 4 (1), 1992.

[3] E. André, W. Finkler, W. Graf, T. Rist, A. Schauder, W. Wahlster, "WIP: The automatic synthesis of multimodal presentations", M.T. Maybury (ed.), *Intelligent Multimedia Presentations*, MIT Press, 1993.

[4] M. Bordegoni, G. Faconti, S. Feiner, M.T. Maybury, T. Rist, S. Ruggiery, P. Trahanias, M. Wilson, "A standard reference model for intelligent multimedia presentation systems", *Journal of Computer Interfaces & Standards*, Special Issue on Intelligent Multimedia Presentation Systems, 1997.

[5] M.T.Maybury, "Planning multimedia explanations using communicative acts", M.T. Maybury (ed.), *Intelligent Multimedia Presentations*, MIT Press, 1993.

[6] J. Hernández and M. Molina, "Advanced human-computer interaction for decision support systems using knowledge modelling techniques", *15th International Federation on Information Processing (IFIP) World Computer Congress, IFIP'98. IT&KNOWS Conference, Information Technology and Knowledge Systems*, pp. 400-414, Austrian Computer Society, 1998.

[7] C. W. Holsapple and A. B. Whinston, *Decision Support Systems: A Knowledge-Based Approach*, West, St. Paul, 1996.

[8] M. Molina, J.L. Sierra and J.M. Serrano, "A language to formalise and to operationalise problem solving Strategies of structured knowledge models", *8th Workshop on Knowledge Engineering: Methods & Languages* (KEML'98), Karlsruhe, 1998.

[9] FLUIDS, Future Lines of User Interface Decision Support Systems, Telematics Applications Programme, IV Framework Programme, Commission of the European Communities, 1996. URL: http://www.dfki.de/fluids.

[10] V. Mauro and C. Di Taranto, "UTOPIA", IFAC/IFIP/IFORS *Conference on Control, Computers and Communications in Transport*, Paris, 1989.